University of Zurich[UZH]

Faculty of Science
Institute for Computational Science

Master Thesis

Extreme Scale Particle Hydro Methods

Thomas Meier

supervised by

Prof. Dr. Joachim Stadel
Dr. Douglas Potter

December 18, 2021

Abstract

Fluid dynamics simulations are a widely used tool in astrophysics and cosmology to study various phenomena from the formation of cosmological structure, galaxies, stars and planets to planetary collisions and supernovae. One of the methods used to describe the fluid is Smoothed Particle Hydrodynamic (SPH) which can be coupled to N-Body gravity codes in a straightforward way. In this Master thesis, SPH is implemented into the N-Body code PKDGRAV3 using a novel approach to the neighbor search which has been a performance problem in existing implementations. To demonstrate the capabilities of the new implementation, example simulations for planetary collisions and cosmological structure formation were performed. We also demonstrate the scaling capabilities in simulations up to 2×10^9 particles and compare performance with GASOLINE which is currently used to run planetary collision simulations for publications. We show that the new implementation is able to run much larger simulations in less time than GASOLINE.

Acknowledgments

I would like to thank my supervisor Prof. Dr. Joachim Stadel who inspired me to do this work. Furthermore, I want to thank Dr. Douglas Potter for his help in understanding the existing code and his effort to make the PKDGRAV3 gravity code scale on Eiger. I also want to thank the CSCS where many of the collision simulations were run on Eiger. Lastly, I would like to thank Dr. Christian Reinhardt for stimulating discussions and his perusal of this work.

Contents

1 Introduction

Understanding phenomena in astrophysics and cosmology is not as straightforward as in other disciplines of physics. Observations with telescopes can only deliver a snapshot of processes that last decades or millennia or even longer and often do not have the necessary resolution to resolve the interesting features of an object. Due to the large spatial and temporal scale of these objects, studying them in the laboratory is not possible. Thus, much of scientific work in this area has been focused on running computer simulations of the system of interest and comparing the results to observations. Many such systems (stars, galaxies, baryons in cosmological structure formation) consist of some gaseous material, which can be described by fluid dynamics (in the context of planetary collisions even rock and iron can be treated as a fluid). As gravity is usually strong enough to significantly influence or dominate the behavior, the set of equations used to describe the fluid are the Euler equations with self-gravity (see Section 2.1.1). The Euler equations are continuum equations, and in order to solve them numerically, they have to be discretized spatially and temporally. The spatial discretization can be done in many different ways, using differential or integral, as well as Eulerian or Lagrangian approaches. One of the often used methods is Smoothed Particle Hydrodynamics (SPH) which discretizes the fluid into packets called particles which move with the flow (Lagrangian) described by the differential form of the Euler equations.
Until now, the ICS uses a highly modified version of the SPH code Gasoline [1] to run planetary collision simulations [2, 3, 4, 5, 6]. As the code is dating back to the early 2000s, it has difficulties scaling on modern hardware. It is thus infeasible to run simulations with the high resolutions necessary to resolve features like fluid mixing vortices, subsurface structure, oceans or atmospheres of rocky planets. Earth's hydrosphere weighs around $2.3 \times 10^{-4}\,\mathrm{M}_\oplus$ while its atmosphere weighs $8.3 \times 10^{-7}\,\mathrm{M}_\oplus$. So at our go-to resolution for collision simulations of 2×10^5 particles, this would mean 46 particles for the hydrosphere and fewer than 1 for the atmosphere, assuming equal mass particles, since all particles are required to have similar mass to avoid numerical instabilities [7]. Furthermore, a higher number of particles allows the resolution of details in the flow and shock structure that can not be resolved by lower resolution simulations. This can be seen in Figure 1 which shows a result of a collision simulation between two $1\,\mathrm{M}_\mathrm{J}$ ideal gas balls for particle numbers of 2×10^5 and 2×10^9.

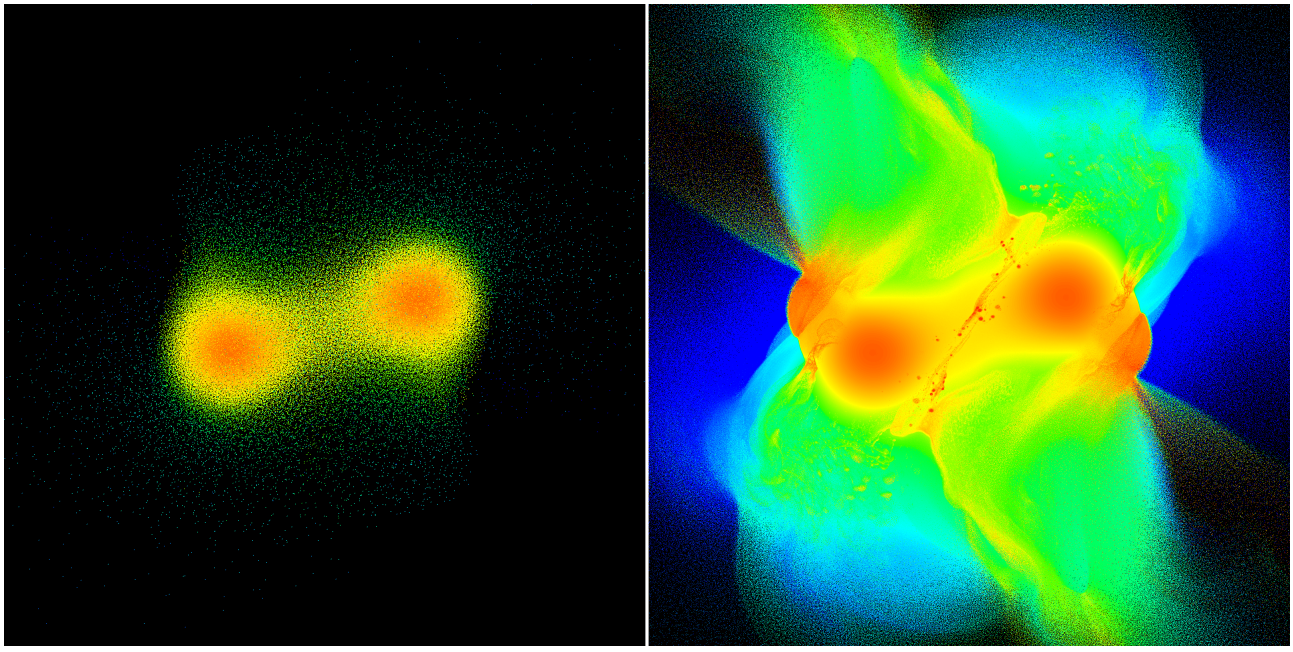

Figure 1: Comparison of the result of a collision simulation between two $1\,\mathrm{M}_\mathrm{J}$ ideal gas balls with 1×10^5 particles (left) and 1×10^9 particles each (right). Clearly, the low particle number of 2×10^5 particles is not enough to resolve the details in the structure of the shocks and ejecta.

So, the need for a new higher performance code is clear. As PKDGRAV3 (the successor to PKDGRAV on which GASOLINE is based) is state of the art in resolution, scalability and performance for gravity calculation, it is a logical step to extend it with an SPH module.
In this work, we implement the SPH hydro method into the pure gravity code PKDGRAV3 and we show that it vastly outperforms and outscales GASOLINE on our main work machine Eiger at CSCS.
In Section 2, a short overview over the theory of fluid dynamics with self-gravity is given, together with a short introduction on how to solve the equations numerically. This is followed by an in-depth discussion of SPH and time integration methods. The implementation of SPH into PKDGRAV3 is then described in Section 3. In Section 4 performance results, comparisons with GASOLINE and example simulations are shown. Finally, in Section 5 we conclude the work and give an outlook on future efforts.

2 Theory

In this section, a short overview of the general concept of computational hydrodynamics with self-gravity is given. First, the general equations of hydrodynamics with self-gravity and some methods to discretize them are described. Next, some methods to calculate the acceleration due to self-gravity are shown. Then, the Smoothed Particle Hydrodynamics (SPH) method is described in detail. Lastly, some time integration schemes and special aspects of the application in cosmology are discussed.

2.1 Hydrodynamics with self-gravity

Many phenomena in astrophysics and cosmology can be studied with the use of hydrodynamics with self-gravity. These applications range from cosmological structure formation simulations over galaxy formation, protoplanetary disks, supernovae and stellar dynamics to planetary collisions. Many of these applications involve only gaseous material, but even in the cases where more complex materials (like rock and iron) are involved, hydrodynamics is able to describe the properties, as gravitational forces dominate over the material strength (see [8] for a detailed explanation).

2.1.1 Euler equations

The behavior of gas is described by the kinetic gas theory using microscopic conservation laws. This leads to the Boltzmann equation

$$\frac{\partial f}{\partial t} + \boldsymbol{u} \cdot \frac{\partial f}{\partial \boldsymbol{x}} + \boldsymbol{a} \cdot \frac{\partial f}{\partial \boldsymbol{u}} = \int_{4\pi} \int_{\mathbb{R}^3} (f_1' f_2' - f_1 f_2) \sigma v \mathrm{d}\Omega \mathrm{d}^3 u_2 \tag{2.1}$$

where f is the phase space distribution function of the gas particles, u is the velocity and the right side is called the collision integral. By taking the moments (integrating over the velocity, weighted by a factor m, mu_i or $\frac{1}{2}mu^2$) we get the macroscopic conservation or fluid equations (for a derivation see [9])

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0 \tag{2.2}$$

$$\frac{\partial}{\partial t}(\rho \boldsymbol{v}) + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + \mathbb{P}) = \rho \boldsymbol{a} \tag{2.3}$$

$$\frac{\partial E}{\partial t} + \boldsymbol{\nabla} \cdot (E\boldsymbol{v} + \mathbb{P} \cdot \boldsymbol{v} + \boldsymbol{Q}) = \rho \boldsymbol{v} \cdot \boldsymbol{a} \tag{2.4}$$

By assuming local thermodynamical equilibrium, the distribution functions take the form of the Maxwell-Boltzmann distribution and the off-diagonal elements of the pressure tensor $\mathbb{P}$ and the heat flux $\boldsymbol{Q}$ vanish. This simplifies the macroscopic conservation equations to the Euler equations in conservative or Eulerian formulation (where we replace the external acceleration $\boldsymbol{a}$ by the gravitational acceleration $\boldsymbol{g}$):

$$\text{Conservation equation:} \quad \frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0 \tag{2.5}$$

$$\text{Momentum equation:} \quad \frac{\partial}{\partial t}(\rho \boldsymbol{v}) + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + P\mathbb{I}) = \rho \boldsymbol{g} \tag{2.6}$$

$$\text{Energy equation:} \quad \frac{\partial E}{\partial t} + \boldsymbol{\nabla} \cdot ((E + P)\boldsymbol{v}) = \rho \boldsymbol{v} \cdot \boldsymbol{g} \tag{2.7}$$

By replacing the total energy with

$$E = \frac{1}{2}\rho v^2 + e = \frac{1}{2}\rho v^2 + \rho\epsilon \tag{2.8}$$

where ϵ is the specific internal (or thermal) energy (the specific internal energy is also often denoted as u) and using the Lagrangian time derivative

$$\frac{\mathrm{D}}{\mathrm{D}t} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \tag{2.9}$$

we can write the Euler equations in Lagrangian form (or the time derivative following the fluid motion):

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} = -\rho \boldsymbol{\nabla} \cdot \boldsymbol{v} \tag{2.10}$$

$$\rho \frac{\mathrm{D}\boldsymbol{v}}{\mathrm{D}t} = -\boldsymbol{\nabla} P + \rho \boldsymbol{g} \tag{2.11}$$

$$\rho \frac{\mathrm{D}\epsilon}{\mathrm{D}t} = -P \boldsymbol{\nabla} \cdot \boldsymbol{v} \tag{2.12}$$

2.1.2 Discretization of the Euler equations

In order to solve the Euler equations numerically with a computer, we need to discretize them. There are many different ways to approach this, and we can differentiate them into four groups by the way the frame is chosen (Eulerian static or Lagrangian moving) and the form of the equation that is solved (differential or integral). In Table 1, the four groups are shown.

	Eulerian	Lagrangian
Differential	Finite Difference The Pencil Code [10]	SPH Gasoline [1], gadget-4 [11]
Integral	Finite Volume PLUTO [12], Athena [13] RAMSES [16], ENZO [17]	Moving Mesh/Meshless Arepo [14], TESS [15] MFM/MFV [18]

Table 1: Four different ways to discretize the Euler equations with example codes.

Each of these approaches has its own advantages and disadvantages that make it suitable for certain applications. In general, Eulerian formulations share a common disadvantage in that they are not Galilean invariant by construction. This means that adding a global velocity increases advection errors and can require prohibitively small time steps to be stable. Lagrangian formulations do not have these flaws, as the discretization elements follow the fluid motion. The main differences between the differential and integral formulation is the mixing behavior and the form of numerical dissipation. While differential methods underestimate, integral formulations overestimate mixing between different fluids [19]. Differential methods use artificial viscosity to treat discontinuities, while integral methods use a Riemann solver with a slope limiter. Both ways introduce excess unphysical dissipation. The different methods also differ in the way an adaptive dynamic resolution can be achieved. Eulerian methods have to rely on adaptive mesh refinement (AMR), whereas in Lagrangian methods, the comoving discretization leads to an increase in discretization element density (particles, sampling points or mesh elements) in high density regions. While particle-based methods have excellent conservation properties, mesh-based methods usually do not conserve angular momentum and finite difference methods not even conserve mass and linear momentum [18]. There also exist methods that combine multiple aspects, or do not fit in these categories, like for example Phurbas [20, 21] which uses interpolation on moving sample points that do not represent fluid packets.

2.1.3 Including additional physics

Various codes include other physics either on resolution scale (e.g. physical viscosity [22], diffusion of species [23], heat conduction [24], magnetohydrodynamics (MHD) [25]) or on sub-resolution scale (e.g. turbulence modeling [26], feedback in cosmological structure formation by galaxy and black hole

formation [27], supernovae [28] and AGN [29], galaxy formation sub-grid models [30], more accurate equations of state [6]).

2.2 Gravity

In many applications in theoretical astrophysics and cosmology, self-gravity is crucial and has to be included in the calculation. The gravitational acceleration $\boldsymbol{g}$ that enters the Euler equations via the momentum equations (2.6) and (2.11) is given as

$$\boldsymbol{g} = -\boldsymbol{\nabla}\phi \tag{2.13}$$

The gravitational potential ϕ is the solution of Poisson's equation

$$\Delta\phi = 4\pi G\rho \tag{2.14}$$

where ρ is the density and G is the gravitational constant. Some commonly used methods to calculate the self-gravity are described below.

2.2.1 Direct method

In the case of N point masses, the gravitational potential given by the Poisson equation (2.14) and the gravitational acceleration are given by

$$\phi(\boldsymbol{x}) = -\sum_{i=1}^{N} \frac{Gm_i}{|\boldsymbol{x}-\boldsymbol{x}_i|} \qquad \boldsymbol{g}(\boldsymbol{x}) = -\sum_{i=1}^{N} \frac{Gm_i(\boldsymbol{x}-\boldsymbol{x}_i)}{|\boldsymbol{x}-\boldsymbol{x}_i|^3} \tag{2.15}$$

The direct gravity method calculates this sum for all particles. This involves $\mathcal{O}(N^2)$ calculations, which is infeasible for large N and it is thus rarely used except for special cases where N is not too large. One method to improve performance is to use special purpose hardware such as the GRAPE [31], where the gravitational interaction is calculated using FPGA hardware.

2.2.2 Tree methods

The potential due to distant groups of particles can be approximated by multipole expansions, usually about the groups center of mass but other points can be chosen. Only the closest particles have to be treated directly (usually the direct force is also softened to lower the impact of close encounters or model continua like fluids [32]). The multipole expansion introduces errors in the force, which are controlled by a single parameter (the opening angle) that determines how small and distant a group must be to use the multipole expansion. Codes that use this method expand the potentials to quadrupole or higher order and construct a tree hierarchy of the particles. The classic Barnes-Hutt tree-code [33] uses an oct-tree, but other trees are also possible, like binary trees. Tree methods can reduce the computational complexity to $\mathcal{O}(N \log N)$. The Fast Multipole Method (FMM) [34] uses the idea that nearby particles are subject to a similar acceleration due to distant groups of particles. The potential generated by the source cell at the position of the sink cell is multipole expanded to give the potential for all particles in the sink cell. This reduces the computational complexity to nearly $\mathcal{O}(N)$ [35]. PKDGRAV3 uses this method [36].

2.2.3 Particle-mesh methods

In the classic P-M method, the gravitational potential is calculated on a grid, by solving the Poisson equation

$$\Delta\phi = 4\pi G\rho \tag{2.16}$$

With this, gravitational interactions at small scales, below the cell length, are smoothed. The density field ρ is constructed from the particles by using a kernel to split the mass of the particles to the grid

cells around the particle position. The simplest choice is to assign the full mass to the cell in which the particle is, but this leads to large force fluctuations. Better results can be obtained by using a cloud-in-cell (CIC) or triangular shape cloud (TSC) kernel. The Poisson equation is typically solved using the Fast Fourier Transform, but standard grid methods can also be used. The force derived from the potential on the grid is then assigned back to the particles using the same kernel as in the density field construction. The method is of order $\mathcal{O}(N_p)$ for the particle number N_p and $\mathcal{O}(N_g \log N_g)$ for the number of grid cells N_g when using the FFT method. The dynamic range of the particle-mesh method can be increased by using an adaptive instead of a fixed grid to solve the Poisson equation. Similar to the AMR method for the hydrodynamic equations, the grid cells are concentrated in the regions where higher resolution is necessary, for example in high-density regions. Codes that use this method are RAMSES [16] and ENZO [17]. Another way to increase the force resolution of particle mesh codes is to couple a mean field description on large scales with a direct, softened, treatment of the gravitational interactions on distances of order of a few grid spacings [37]. This method is called P^3M (particle-particle-particle-mesh) and can efficiently increase the resolution of PM methods, but in the presence of strong clustering, large numbers of particles interact directly with each other, increasing the computational complexity to $\mathcal{O}(N^2)$. This can be resolved by using adaptive meshes, refining the grid in high-density regions. Adaptive P^3M codes can bring the computational cost to $\mathcal{O}(N \log N)$, like a tree code. Another possibility is to use a tree code for the short range force calculation, leading to a hybrid PM-Tree method. An example for a code that uses this method is GADGET-2 [38] (the newest version in the GADGET series, GADGET-4 can also use the fast multipole method).

2.3 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian method that discretizes the fluid into particles and uses a kernel function to smooth their distribution and calculate the differential operators in the differential form of the Euler equations. The particle positions are evolved with the fluid motion, such that we can rewrite the Lagrangian time derivative for the derivatives of the quantities on the sampling points as the normal time derivative:

$$\frac{\mathrm{D}}{\mathrm{D}t} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla} \qquad \rightarrow \qquad \frac{\mathrm{d}}{\mathrm{d}t} \tag{2.17}$$

This makes SPH intrinsically Galilean invariant which means that it does not suffer from one of the main disadvantages of Eulerian methods, the velocity dependent advection errors and timestep conditions only depend on local velocity differences, not the velocity relative to the reference frame. SPH provides exact mass, energy, linear and angular momentum conservation. When handled correctly (see Section 2.3.2.1) SPH also provides continuous adaptive resolution. In Section 2.3.1, traditional SPH is derived and some more detailed aspects are described. In Section 2.3.2 the SPH method is derived directly from the Lagrangian and some of the drawbacks of SPH are addressed with suggested solutions for them.

2.3.1 Traditional SPH

Smoothed Particle Hydrodynamics (SPH) was developed in the late 1970s for astrophysical applications simultaneously by Gingold and Monaghan [39] and Lucy [40]. The following description of traditional SPH is similar to the one in my Bachelor thesis [41].

2.3.1.1 The Kernel In SPH, the fluid is discretized using particles that interact via a kernel function $W(\boldsymbol{d}, h)$ with a characteristic radius called the smoothing length h. Physical properties (e.g. ρ, P, u) at a given position (e.g. a particle position) can be determined by integration of the respective quantity over the kernel volume, weighted by the kernel function:

$$f(\boldsymbol{r}) = \int f(\boldsymbol{r}') W(\boldsymbol{r} - \boldsymbol{r}', h) \mathrm{d}\boldsymbol{r}' \tag{2.18}$$

The integral is then discretized by a Riemann summation over the particles in the kernel

$$f(\boldsymbol{r}_i) = \sum_j m_j \frac{f_j}{\rho_j} W(\boldsymbol{r}_i - \boldsymbol{r}_j, h_i) \tag{2.19}$$

The kernel can be any function that satisfies the following properties:

$$\lim_{h\to 0} W(\boldsymbol{r} - \boldsymbol{r}', h) = \delta(\boldsymbol{r} - \boldsymbol{r}') \tag{2.20}$$

$$\int W(\boldsymbol{r} - \boldsymbol{r}', h) \mathrm{d}\boldsymbol{r}' = 1 \tag{2.21}$$

Commonly used kernel functions are the Gaussian, B-splines [42] or Wendland functions [43]. The latter two are compact kernels, which means that they have a finite interaction length, while the Gaussian kernel includes contributions of all particles. Compact kernels greatly improve computational performance, as the Riemann sums have to be performed only over a small set of particles and not over all. To handle the large range of changing length scales in cosmological and astrophysical simulations, the kernel smoothing length is usually variable and determined on a particle-by-particle basis such that the kernel always contains a certain number of particles [44]. This allows SPH to adapt its resolution to changing local conditions and makes it ideal for simulating problems that cover large spatial dimensions and have large density contrasts. Typically, the kernel is symmetrized to conserve momentum and energy in the equations of motion

$$W_{ij} = \frac{1}{2} W(|\boldsymbol{r}_i - \boldsymbol{r}_j|, h_i) + \frac{1}{2} W(|\boldsymbol{r}_i - \boldsymbol{r}_j|, h_j) \tag{2.22}$$

As SPH consists of two approximations, the kernel weighted integral and the discretization with Riemann sums, true convergence needs not only $h \to 0$ and $N \to \infty$, but also $N_{Kernel} \to \infty$.

2.3.1.2 Density The continuity Equation (2.10) that describes the rate of change of density can be discretized to

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = \sum_j m_j \boldsymbol{v}_{ij} \cdot \boldsymbol{\nabla}_i W_{ij} \tag{2.23}$$

but in SPH, the density of a particle is usually calculated by the interpolant

$$\rho_i = \sum_j m_j W_i \tag{2.24}$$

If the masses of the particles are fixed, which is usually the case, mass is exactly conserved.

2.3.1.3 The momentum equation The positions of the particles that represent the fluid are evolved by integrating their velocity. The straightforward discretization of the momentum equation (2.11) yields

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\frac{1}{\rho_i} \sum_j \frac{P_j}{\rho_j} \boldsymbol{\nabla}_i W_{ij} \tag{2.25}$$

but this does not conserve linear and angular momentum because the forces acting between two particles are not symmetric. By writing

$$\frac{\boldsymbol{\nabla} P}{\rho} = \boldsymbol{\nabla}\left(\frac{P}{\rho}\right) + \frac{P}{\rho^2} \boldsymbol{\nabla}\rho \tag{2.26}$$

we get a discretization that conserves linear and angular momentum [42]

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \boldsymbol{\nabla}_i W_{ij} \tag{2.27}$$

The term $\frac{\boldsymbol{\nabla} P}{\rho}$ can be written in different ways, some of which are described in Section 2.3.2.7.

2.3.1.4 The energy equation For compressible flow, the internal energy u of the particles has to be considered. The non-dissipative rate of change of the internal energy (Equation (2.12)) can be expressed as

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{P}{\rho^2}\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\frac{P}{\rho}\boldsymbol{\nabla}\cdot\boldsymbol{v} \tag{2.28}$$

where Equation (2.10) was used. This can then be discretized into

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{P_i}{\rho_i^2}\sum_j m_j \boldsymbol{v}_{ij}\cdot\boldsymbol{\nabla}_i W_{ij} \tag{2.29}$$

2.3.1.5 Equation of state The Euler equations now contain 6 unknowns for each particle

$$\boldsymbol{v} = (v_x, v_y, v_z) \qquad \rho \qquad P \qquad u \tag{2.30}$$

but only 5 equations. Therefore, an equation of state (EOS) is needed to complete the description of the system by relating ρ, u and P. One commonly used EOS to describe gas is the ideal gas equation of state

$$P = (\gamma - 1)\rho u \qquad c = \sqrt{\gamma(\gamma-1)u} \tag{2.31}$$

with the adiabatic index γ. The ideal gas equation of state is the simplest EOS that considers temperature changes, but ignores interactions between gas molecules. It is thus a good approximation for very low density fluids like in cosmology simulations. The isothermal gas is an even simpler EOS, where the temperature is fixed $T = const$ and we get for the pressure

$$P = \rho a^2 \qquad a = \sqrt{\frac{k_B T_0}{m}} \tag{2.32}$$

with a the isothermal sound speed. This EOS removes the need for the energy equation.

2.3.1.6 Artificial viscosity Dissipative processes occur in flows due to viscosity as described by the Navier-Stokes equation or in the form of high Mach number flows in shocks. To deal with discontinuities like shocks, an artificial viscosity Π_{ij} is introduced and added to Equation (2.27) [42]

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij} \right) \boldsymbol{\nabla}_i W_{ij} \tag{2.33}$$

The artificial viscosity has the form

$$\Pi_{ij} = \begin{cases} \frac{-\alpha \bar{c}_{ij}\mu_{ij} + \beta\mu_{ij}^2}{\bar{\rho}_{ij}} & \text{for } \boldsymbol{v}_{ij}\cdot\boldsymbol{r}_{ij} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.34}$$

where

$$\bar{c}_{ij} = \frac{c_i + c_j}{2} \qquad \bar{\rho}_{ij} = \frac{\rho_i + \rho_j}{2} \qquad h_{ij} = \frac{h_i + h_j}{2} \tag{2.35}$$

and μ_{ij} is given as

$$\mu_{ij} = \frac{h_{ij}(\boldsymbol{v}_{ij} \cdot \boldsymbol{r}_{ij})}{r_{ij}^2 + \epsilon h_{ij}^2} \tag{2.36}$$

α and β are the shear and Von Neumann-Richtmyer viscosities and are usually set to $\alpha = 1.0 - 1.5$ and $\beta = 2\alpha = 2.0 - 3.0$, and the softening is $\epsilon = 0.01$ [45]. The artificial viscosity also adds a term to the internal energy

$$\frac{\mathrm{d}u_\Pi}{\mathrm{d}t} = \frac{1}{2}\sum_j m_j \Pi_{ij} \boldsymbol{v}_{ij} \cdot \boldsymbol{\nabla}_i W_{ij} \tag{2.37}$$

to account for the increase in entropy due to shock heating.

2.3.2 Modern SPH

Traditional SPH has many shortcomings. One of the main arguments agains the use of SPH is that it suppresses fluid-mixing instabilities like the Kelvin-Hemholtz instability [19] and subsonic turbulence [46]. The artificial viscosity needed to capture discontinuities, both in the initial conditions and in shocks, leads to unwanted artificial angular momentum transfer and suppression of subsonic turbulence. Compared to the excellent shock capturing capabilities of methods with Riemann solver, SPH smooths shocks over multiple smoothing lengths. Many modifications, improvements and tweaks were proposed over time to solve some of these drawbacks, but the dust has not yet settled to reveal a consensus on a new standard method. In Section 2.3.2.1 we first derive the SPH equations directly from the Lagrangian, and show how to add correction terms due to a spatially varying smoothing length to the equations of motion. In Section 2.3.2.3 we prove that the derived equations actually conserve all advertised quantities. In Sections 2.3.2.4 - 2.3.2.7 some of the solutions for the shortcomings are discussed and in Section 2.3.2.8 a way to make SPH also conserve entropy for the ideal gas is described.

2.3.2.1 Derivation from the Lagrangian The SPH formulation described in Section 2.3.1 is derived by discretizing integrals. But the SPH equations can also be derived from the Lagrangian of an ideal fluid (closely following [25])

$$L = \int \left[\rho v^2 - \rho u(\rho, s)\right] \mathrm{d}V \tag{2.38}$$

where the internal energy per unit mass u is a function of the thermodynamic variables ρ (density) and s (entropy). We can now discretize this expression

$$L = \sum_i m_i \left[\frac{1}{2} v_i^2 - u_i(\rho_i, s_i)\right] \tag{2.39}$$

and we can consider the system as a discrete Hamiltonian system, without explicit reference to the continuum system. So all associated symmetries and equations of motion apply. We can thus use the least action principle and the Euler-Lagrange equations to derive the equations of motion. By minimizing the action

$$S = \int L \mathrm{d}t \qquad \delta S = \int \delta L \mathrm{d}t = 0 \tag{2.40}$$

we get

$$\delta S = \int \left(\frac{\partial L}{\partial \boldsymbol{v}} \cdot \delta \boldsymbol{v} + \frac{\partial L}{\partial \boldsymbol{r}} \cdot \delta \boldsymbol{r}\right) \mathrm{d}t = 0 \tag{2.41}$$

We integrate by parts and use that $\delta \boldsymbol{v} = \frac{\mathrm{d}(\delta \boldsymbol{r})}{\mathrm{d}t}$ with $\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\partial}{\partial t} + \boldsymbol{v} \cdot \boldsymbol{\nabla}$ and we get

$$\delta S = \int \left\{\left[-\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \boldsymbol{v}}\right) + \frac{\partial L}{\partial \boldsymbol{r}}\right] \cdot \delta \boldsymbol{r}\right\} \mathrm{d}t + \left[\frac{\partial L}{\partial \boldsymbol{v}} \cdot \delta \boldsymbol{r}\right]_{t_0}^{t} = 0 \tag{2.42}$$

We assume that the variation $\delta \boldsymbol{r}$ vanishes at the start and the end and is arbitrary elsewhere. Thus, we get that the equations of motion are given by the Euler-Lagrange equations (here for particle i):

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \boldsymbol{v}_i}\right) - \frac{\partial L}{\partial \boldsymbol{r}_i} = 0 \tag{2.43}$$

Until here, we assumed that the time integral is not discrete, so when we later discuss exact conservation, it is always meant to be solely governed by errors in the time integration scheme. We also assumed that the Lagrangian is differentiable (this has meaning in reference to discontinuities for example in the initial conditions which have to be smoothed using dissipative terms). To calculate the equations of motion we need

$$\frac{\partial L}{\partial \boldsymbol{v}_i} = m_i \boldsymbol{v}_i \qquad \frac{\partial L}{\partial \boldsymbol{r}_i} = -\sum_j m_j \left.\frac{\partial u_j}{\partial \rho_j}\right|_s \frac{\partial \rho_j}{\partial \boldsymbol{r}_i} \tag{2.44}$$

since u is a function of ρ and s but we assume s constant. From the first law of thermodynamics, we have

$$\mathrm{d}U = T\mathrm{d}S - P\mathrm{d}V \tag{2.45}$$

and per unit mass, where we use $\mathrm{d}V = -\frac{m}{\rho^2}\mathrm{d}\rho$, we get

$$\mathrm{d}u = T\mathrm{d}s + \frac{P}{\rho^2}\mathrm{d}\rho \tag{2.46}$$

such that at constant entropy, the change in thermal energy is given as

$$\left.\frac{\partial u_j}{\partial \rho_j}\right|_s = \frac{P_j}{\rho_j^2} \tag{2.47}$$

If we differentiate the density estimate (Equation (2.24)) we get

$$\frac{\partial \rho_j}{\partial \boldsymbol{r}_i} = \frac{1}{\Omega_j}\sum_k m_k \frac{\partial W_{jk}(h_j)}{\partial \boldsymbol{r}_i}(\delta_{ji} - \delta_{ki}) \tag{2.48}$$

where $W_{jk}(h_j) = W(|\boldsymbol{r}_j - \boldsymbol{r}_k|, h_j)$ and δ_{ji} is a Dirac delta function. Ω_j is a correction factor that accounts for the fact that h is itself a function of ρ and it is given by

$$\Omega_i = \left[1 - \frac{\partial h_i}{\partial \rho_i}\sum_j m_j \frac{\partial W_{ij}(h_i)}{\partial h_i}\right] = 1 + \frac{h_i}{3\rho_i}\sum_j m_j \frac{\partial W_i(h_i)}{\partial h_i} \tag{2.49}$$

where we used

$$\frac{\partial h}{\partial \rho} = -\frac{h}{\rho d} \tag{2.50}$$

with the dimensionality $d = 3$. Ω_i is also called the grad-h or ∇h term. We can now calculate the equation of motion for the velocity from

$$\frac{\partial L}{\partial \boldsymbol{r}_i} = -\sum_j m_j \frac{P_j}{\Omega_j \rho_j^2}\sum_k m_k \frac{\partial W_{jk}(h_j)}{\partial \boldsymbol{r}_i}(\delta_{ji} - \delta_{ki}) \tag{2.51}$$

which we can simplify to give the equation of motion

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j m_j \left[\frac{P_i}{\Omega_i \rho_i^2}\frac{\partial W_{ij}(h_i)}{\partial \boldsymbol{r}_i} + \frac{P_j}{\Omega_j \rho_j^2}\frac{\partial W_{ij}(h_j)}{\partial \boldsymbol{r}_i}\right] \tag{2.52}$$

which simplifies for a constant smoothing length to Equation (2.27). For the internal energy equation we start from

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{P_i}{\rho_i^2}\frac{\mathrm{d}\rho_i}{\mathrm{d}t} \tag{2.53}$$

Using the density derivative, we get

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{P_i}{\Omega_i\rho_i^2}\sum_j m_j(\boldsymbol{v}_i - \boldsymbol{v}_j)\cdot\boldsymbol{\nabla}_i W_{ij}(h_i) \tag{2.54}$$

2.3.2.2 Calculating the smoothing length In classic SPH, the smoothing length h is defined such that the number of particles inside the kernel is constant. But this leads to a discontinuous h-field, which makes it impossible to correct for a variable smoothing length in the SPH equations. Thus, when one wants to include these so-called grad-h terms (see Equation (2.49)), the smoothing length has to be determined in a way such that the resulting h-field is continuous. For this, one has to iteratively solve the relation

$$\rho_i = \sum_j m_j W_i(\boldsymbol{r}_i - \boldsymbol{r}_j, h_i) \qquad M_{tot}^i = \int_{V_i}\rho\mathrm{d}V = \frac{4}{3}\pi R_{kern}^3(h_i)\rho_i = const. \tag{2.55}$$

simultaneously for h_i and ρ_i. If the mass m is not the same for all particles, this relation has to be modified to use the particle density instead of the mass density:

$$n_i = \sum_j W_i(\boldsymbol{r}_i - \boldsymbol{r}_j, h_i) \qquad N_{tot}^i = \int_{V_i}\rho\mathrm{d}V = \frac{4}{3}\pi R_{kern}^3(h_i)n_i = const. \tag{2.56}$$

Solving this set of equations is usually done using Newton-Raphson iterations.

2.3.2.3 Conservation properties As the particles represent fluid packets, mass is exactly conserved, if the particle masses are constant. By deriving the equations of motion directly from the Lagrangian, we already know that they should conserve linear and angular momentum as well as total energy. But we can also show these properties. From Equation (2.52), we can directly derive the conservation of linear momentum:

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_i m_i\boldsymbol{v}_i = \sum_i m_i\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_i\sum_j m_i m_j\left(\frac{P_i}{\Omega_i\rho_i^2}\boldsymbol{\nabla}_i W_{ij}(h_i) + \frac{P_j}{\Omega_j\rho_j^2}\boldsymbol{\nabla}_i W_{ij}(h_j)\right) = 0 \tag{2.57}$$

where the double sum is zero due to the antisymmetry of the kernel gradient. The total angular momentum is also conserved:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\sum_i \boldsymbol{r}_i\times m_i\boldsymbol{v}_i &= \sum_i m_i\left(\boldsymbol{r}_i\times\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t}\right) &(2.58)\\ &= -\sum_i\sum_j m_i m_j\left(\frac{P_i}{\Omega_i\rho_i^2}F_{ij} + \frac{P_j}{\Omega_j\rho_j^2}\tilde{F}_{ij}\right)\boldsymbol{r}_i\times(\boldsymbol{r}_i - \boldsymbol{r}_j) = 0 &(2.59)\end{aligned}$$

where we wrote $\boldsymbol{\nabla}_i W_{ij}(h_i) = \boldsymbol{r}_{ij}F_{ij}$ and $\boldsymbol{\nabla}_i W_{ij}(h_j) = \boldsymbol{r}_{ij}\tilde{F}_{ij}$ and the last term in the double summation is zero due to $\boldsymbol{r}_i\times\boldsymbol{r}_j = -\boldsymbol{r}_j\times\boldsymbol{r}_i$. We can now consider the conservation of total energy from the Hamiltonian

$$H = \sum_i \boldsymbol{v}_i\cdot\frac{\partial L}{\partial\boldsymbol{v}_i} - L = \sum_i m_i\left(\frac{1}{2}v_i^2 + u_i\right) \tag{2.60}$$

which is the total energy of the SPH particles E since the Lagrangian has no explicit time dependence. We get the time derivative

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \sum_i m_i \left(\boldsymbol{v}_i \cdot \frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} + \frac{\mathrm{d}u_i}{\mathrm{d}t} \right) \tag{2.61}$$

Substituting in the Equations (2.52) and (2.54), we get

$$\begin{aligned}
\frac{\mathrm{d}E}{\mathrm{d}t} &= \sum_i m_i \frac{\mathrm{d}e_i}{\mathrm{d}t} && (2.62)\\
&= \sum_i m_i \left(\boldsymbol{v}_i \cdot \left(-\sum_j m_j \left[\frac{P_i}{\Omega_i \rho_i^2} \boldsymbol{\nabla}_i W_{ij}(h_i) + \frac{P_j}{\Omega_j \rho_j^2} \boldsymbol{\nabla}_i W_{ij}(h_j) \right] \right) \right. && (2.63)\\
&\quad \left. + \frac{P_i}{\Omega_i \rho_i^2} \sum_j m_j (\boldsymbol{v}_i - \boldsymbol{v}_j) \cdot \boldsymbol{\nabla}_i W_{ij}(h_i) \right) && (2.64)\\
&= -\sum_i \sum_j m_i m_j \left[\frac{P_i}{\Omega_i \rho_i^2} \boldsymbol{v}_j \cdot \boldsymbol{\nabla}_i W_{ij}(h_i) + \frac{P_j}{\Omega_j \rho_j^2} \boldsymbol{v}_i \cdot \boldsymbol{\nabla}_i W_{ij}(h_i) \right] = 0 && (2.65)
\end{aligned}$$

where we get again zero due to the antisymmetry of the kernel derivative. So, we get exact conservation (up to the precision of the time integration) for linear and angular momentum and total energy.

2.3.2.4 Viscosity switches To avoid the effects of excess artificial viscosity in converging or shearing flows, different viscosity switches can be applied to limit the artificial viscosity outside of shocks. Standard SPH has a very simple switch, that applies the artificial viscosity only on particle pairs that approach each other (see Equation (2.34)). To reduce the artificial viscosity in differentially rotating systems, the Balsara switch [47] can be applied, which reduces the artificial viscosity by a factor of

$$f_{ij} = \frac{|f_i + f_j|}{2} \tag{2.66}$$

where

$$f_i = \frac{|\boldsymbol{\nabla} \cdot \boldsymbol{v}_i|}{|\boldsymbol{\nabla} \cdot \boldsymbol{v}_i| + |\boldsymbol{\nabla} \times \boldsymbol{v}_i| + 0.0001 \frac{c_i}{h_i}} \tag{2.67}$$

A different way to reduce artificial viscosity was proposed by Morris and Monaghan [48], in the form of an evolving viscosity α:

$$\frac{\mathrm{d}\alpha}{\mathrm{d}t} = -\frac{\alpha - \alpha^\star}{\tau} + S \tag{2.68}$$

where τ is an e-folding timescale, S is a source term and $\alpha^\star$ is the minimum viscosity. For the timescale they propose

$$\tau = \frac{h}{lc} \tag{2.69}$$

with $l = 0.1 - 0.2$ and for the source term they use

$$S = \max(-\boldsymbol{\nabla} \cdot \boldsymbol{v}, 0) \tag{2.70}$$

Cullen and Dehnen [49] find that even though this is better than the Balsara switch, it still inadequately models low-viscosity flows. They propose the following modification:

$$A_i = \xi_i \max(-\dot{\boldsymbol{\nabla}} \cdot \boldsymbol{v}_i, 0) \tag{2.71}$$

with the limiter

$$\xi_i = \frac{\left|2(1-R_i)^4 \boldsymbol{\nabla}\cdot\boldsymbol{v}_i\right|^2}{\left|2(1-R_i)^4 \boldsymbol{\nabla}\cdot\boldsymbol{v}_i\right|^2 + \mathrm{tr}(\boldsymbol{S}_i\cdot\boldsymbol{S}_i^t)} \tag{2.72}$$

where $\boldsymbol{S}$ is the traceless symmetric part of the velocity gradient matrix and R_i is the shock detector

$$-1 \approx R_i = \frac{1}{\rho_i}\sum_j \mathrm{sign}(\boldsymbol{\nabla}\cdot\boldsymbol{v}_j) m_j W \tag{2.73}$$

With this, the viscosity is set to

$$\alpha_{loc,i} = \alpha_{max}\frac{h_i^2 A_i}{v_{sig,i}^2 + h_i^2 A_i} \tag{2.74}$$

with the signal velocity

$$v_{sig,i} = \max(c_{ij} - \min(0, \boldsymbol{v}_{ij}\cdot\hat{\boldsymbol{r}}_{ij})) \tag{2.75}$$

2.3.2.5 Pressure-SPH As said in the beginning of Section 2.3.2, SPH has problems with fluid mixing and instabilities like the Kelvin-Helmholtz instability [19]. One method to improve SPH in this regard was proposed by Hopkins [50]. Generally, the fluid equations do not necessarily have to be formulated in the variables ρ and u, but other combinations can be used, like (P, u) or (P, s). By smoothing the pressure instead of the density, the pressure 'blip' at the discontinuity which acts as a 'surface tension' that suppresses the instability can be removed [19, 51, 52]. So instead of Equation (2.24), one would use

$$P_i = \sum_j (\gamma-1) m_j u_j W_{ij}(h_i) \tag{2.76}$$

to smooth the pressure. This results in the correction factors

$$\Omega_i = 1 + \frac{h_i}{3P_i}\frac{\partial P_i}{\partial h_i} \tag{2.77}$$

and the equation of motion for the velocity is then

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j (\gamma-1)^2 m_j u_i u_j \left[\frac{1}{\Omega_i P_i}\boldsymbol{\nabla}_i W_{ij}(h_i) + \frac{1}{\Omega_j P_j}\boldsymbol{\nabla}_i W_{ij}(h_j)\right] \tag{2.78}$$

and for the internal energy we get

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \sum_j (\gamma-1)^2 m_j u_i u_j \frac{1}{\Omega_i P_i}(\boldsymbol{v}_i - \boldsymbol{v}_j)\cdot\boldsymbol{\nabla}_i W_{ij}(h_i) \tag{2.79}$$

This form of pressure-SPH was derived using a very general formulation of SPH derived directly from the Lagrangian by constructing a weighted volume element

$$V_i = \frac{X_i}{\sum_j X_j W_{ij}} \tag{2.80}$$

for any choice of X_i. The density can then be calculated as

$$\rho_i = \frac{m_i}{V_i} \tag{2.81}$$

The choice $X_i = (\gamma-1)m_i u_i$ leads to the pressure-SPH implemented in SPHYNX [53], while $X_i = m_i$ leads to the formulation from Section 2.3.2.1.

2.3.2.6 Artificial thermal conductivity A different approach to improve the handling of instabilities was proposed by Price [54]. By adding a term to the internal energy derivative, representing an artificial thermal conductivity, the discontinuity is smoothed and this enables the formulation to resolve discontinuities.

$$\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right)_{cond} = -\sum_j \frac{m}{\rho_{ij}} \alpha_u \sqrt{\frac{|P_i - P_j|}{\rho_{ij}}} (u_i - u_j) \hat{\boldsymbol{r}}_{ij} \cdot \boldsymbol{\nabla}_i W_{ij} \tag{2.82}$$

The parameter α_u is handled in the same way as the viscosity in the Morris and Monaghan viscosity switch:

$$\frac{\mathrm{d}\alpha_{i,u}}{\mathrm{d}t} = -\frac{\alpha_{i,u} - \alpha_{min,u}}{\tau_i} + S_{i,u} \tag{2.83}$$

with the source term

$$S_{i,u} = \frac{h_i |\boldsymbol{\nabla}^2 u|_i}{\sqrt{u_i + \epsilon}} \tag{2.84}$$

and $\alpha_{min,u} = 0$.

2.3.2.7 Pressure symmetrization The pressure symmetrization used in Equation (2.26) is not the only possibility. In general, we can write [42]

$$\frac{\boldsymbol{\nabla} P}{\rho} = \frac{P}{\rho^\sigma} \boldsymbol{\nabla}\left(\frac{1}{\rho^{1-\sigma}}\right) + \frac{1}{\rho^{2-\sigma}} \boldsymbol{\nabla}\left(\frac{P}{\rho^{\sigma-1}}\right) \tag{2.85}$$

which results in the following equation of motion

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_j}{\rho_i^{2-\sigma}\rho_j^\sigma} + \frac{P_i}{\rho_i^\sigma \rho_j^{2-\sigma}}\right) \boldsymbol{\nabla}_i W_{ij} \tag{2.86}$$

Setting $\sigma = 2$ yields the expression in Equation (2.27), while $\sigma = 1$ gives the geometric density average used in GASOLINE2 [55]

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(\frac{P_i + P_j}{\rho_i \rho_j}\right) \boldsymbol{\nabla}_i W_{ij} \tag{2.87}$$

and shows very promising results in the box test, which is usually extremely challenging for SPH implementations. Another symmetric combination is

$$\boldsymbol{\nabla} P = 2\sqrt{P}\boldsymbol{\nabla}\sqrt{P} \tag{2.88}$$

which results in the equation of motion

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_j m_j \left(2\frac{\sqrt{P_i P_j}}{\rho_i \rho_j}\right) \boldsymbol{\nabla}_i W_{ij} \tag{2.89}$$

described in [56].

2.3.2.8 Entropic formulation For the ideal gas equation of state, instead of using the internal energy, a different formulation can be used where the internal energy is replaced by an entropic function $A(s)$ which is evolved. In this way, not only energy (and linear and angular momentum) but also entropy is conserved when adaptive smoothing lengths are used and properly accounted for [56]. The entropic function is defined as

$$P = A(s)\rho^\gamma \qquad u = \frac{A(s)}{\gamma - 1}\rho^{\gamma - 1} \tag{2.90}$$

In adiabatic flows, the value of A is conserved, and so the time derivative of A_i is given by

$$\frac{\mathrm{d}A_i}{\mathrm{d}t} = \frac{1}{2}\frac{\gamma-1}{\rho_i^{\gamma-1}} \sum_j m_j \Pi_{ij} \boldsymbol{v}_{ij} \cdot \boldsymbol{\nabla}_i W_{ij} \tag{2.91}$$

and contains only the dissipative terms from the artificial viscosity. This is much more difficult for complex EOS, as not every EOS is thermodynamically complete and provides entropy information (see [2] for an entropy conserving SPH formulation for the Tillotson EOS that does not need entropy information). But if the EOS provides entropy, this can be used directly to implement an entropy conserving formulation.

2.4 Time integration

Generally, the hydrodynamics and gravity methods discussed above transform the continuous Euler equations from a system of coupled partial differential equations to a system of coupled ordinary differential equations (ODE). This system of ODEs then has to be integrated over time, using a time integration scheme. To numerically solve the initial value problem

$$y'(t) = f(t, y(t)) \qquad y(t_0) = y_0 \tag{2.92}$$

different schemes can be used. We define the time step as $h = t_{i+1} - t_i$.

2.4.1 Euler and trapezoidal method

The simplest time integration scheme to solve the initial value problem is the Euler method

$$y_{i+1} = y_i + hf(t_i, y_i) + \mathcal{O}(h^2) \tag{2.93}$$

The Euler method is first order accurate $\mathcal{O}(h)$ and explicit. The implicit Euler method can be written as

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}) + \mathcal{O}(h^2) \tag{2.94}$$

which has to be iteratively solved for y_{i+1}. It is also first order accurate. By combining these two methods, we get the trapezoidal method

$$y_{i+1} = y_i + \frac{h}{2}\left[f(t_i, y_i) + f(t_{i+1}, y_{i+1})\right] + \mathcal{O}(h^3) \tag{2.95}$$

which is also implicit and second order accurate $\mathcal{O}(h^2)$.

2.4.2 Runge-Kutta

Runge-Kutta methods combine the information of several Euler-style steps to match a Taylor series expansion of higher order. The simplest example is

$$k_1 = hf(t_i, y_i) \tag{2.96}$$

$$k_2 = hf\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1\right) \tag{2.97}$$

$$y_{i+1} = y_i + k_2 + \mathcal{O}(h^3) \tag{2.98}$$

which uses two function evaluations to get the second-order Runge-Kutta or midpoint method. The most common variant of Runge-Kutta methods is the following fourth-order scheme

$$
\begin{aligned}
k_1 &= hf(t_i, y_i) && (2.99)\\
k_2 &= hf\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1\right) && (2.100)\\
k_3 &= hf\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2\right) && (2.101)\\
k_3 &= hf(t_i + h, y_i + k_3) && (2.102)\\
y_{i+1} &= y_i + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + \mathcal{O}(h^5) && (2.103)
\end{aligned}
$$

which is often called 'The Runge-Kutta method'. Embedded Runge-Kutta methods allow to use the same function evaluations done for the main step to be used to calculate an error estimation that can be used for adaptive time step control. Runge-Kutta methods can be constructed with arbitrary order, but for orders $n > 4$, more than n function evaluations are required and for $n \geq 8$ at least $n + 3$ function evaluations are required.

2.4.3 Leapfrog

The Leapfrog integrator is a second-order symplectic integrator that is widely used to integrate the second-order ODEs that occur in N-Body simulations

$$\ddot{x} = \dot{v} = a(x) \tag{2.104}$$

where x is the position, v the velocity and $a(x)$ the acceleration due to gravity. As the accelerations do not depend on the velocity, one can split the positions and the velocities and save the latter with half a step offset

$$
\begin{aligned}
x_{i+1} &= x_i + v_{i+\frac{1}{2}}h && (2.105)\\
v_{i+\frac{3}{2}} &= v_{i+\frac{1}{2}} + a(x_{i+1})h && (2.106)
\end{aligned}
$$

Symplectic integrators can be seen as canonical transformations of the system and thus conserve the energy of the system. Neither of the methods described before (explicit/implicit Euler, trapezoidal and Runge-Kutta methods) are symplectic. For an actual implementation in an N-body code, the formal notation above is split, resulting in either the Drift-Kick-Drift scheme

$$
\begin{aligned}
x_{i+\frac{1}{2}} &= x_i + v_i\frac{h}{2} && (2.107)\\
v_{i+1} &= v_i + a\left(x_{i+\frac{1}{2}}\right)h && (2.108)\\
x_{i+1} &= x_{i+\frac{1}{2}} + v_{i+1}\frac{h}{2} && (2.109)
\end{aligned}
$$

or the Kick-Drift-Kick scheme

$$
\begin{aligned}
v_{i+\frac{1}{2}} &= v_i + a(x_i)\frac{h}{2} && (2.110)\\
x_{i+1} &= x_i + v_{i+\frac{1}{2}}h && (2.111)\\
v_{i+1} &= v_{i+\frac{1}{2}} + a(x_{i+1})\frac{h}{2} && (2.112)
\end{aligned}
$$

The KDK scheme is preferable, as accelerations and potentials are known at the second-order-accurate positions and not at the auxiliary intermediate position allowing them to be used for time-step control. When a block time-stepping scheme is used, KDK also results in synchronized force computations for all active particles [57].

2.4.4 Predictor-Corrector Scheme

The leapfrog scheme is well suited, when the force does not depend on the velocity, as in pure gravity, but hydrodynamic forces depend on the velocity and the thermodynamical variable. So in order to calculate the force, we need predicted values at the time of the force evaluation. Usually a predictor-corrector method is woven into the leapfrog to get the predicted values. The predictor-corrector scheme to solve the initial value problem uses the following two steps

$$\tilde{y}_{i+1} = y_i + hf(t_i, y_i) \tag{2.113}$$
$$y_{i+1} = y_i + \frac{1}{2}h\left[f(t_i, y_i) + f(t_{i+1}, \tilde{y}_{i+1})\right] + \mathcal{O}(h^3) \tag{2.114}$$

which consists of an Euler prediction with a trapezoidal correction called PECE mode. One can also skip the second evaluation to get

$$\tilde{y}_{i+1} = y_i + hf(t_i, \tilde{y}_i) \tag{2.115}$$
$$y_{i+1} = y_i + \frac{1}{2}h\left[f(t_i, \tilde{y}_i) + f(t_{i+1}, \tilde{y}_{i+1})\right] + \mathcal{O}(h^3) \tag{2.116}$$

where the value $f(t_i, \tilde{y}_i)$ is reused from the last step. This is called PEC mode. The PEC mode fits perfectly in the KDK leapfrog as it can also be split into the kick-drift-kick steps

$$v_{i+\frac{1}{2}} = v_i + a(t_i, \tilde{v}_i)\frac{h}{2} \tag{2.117}$$
$$\tilde{v}_{i+1} = v_i + a(t_i, \tilde{v}_i)h \tag{2.118}$$
$$v_{i+1} = v_{i+\frac{1}{2}} + a(t_{i+1}, \tilde{v}_{i+1})\frac{h}{2} \tag{2.119}$$

2.4.5 Symplectic integration in comoving coordinates

In cosmological simulations, we also have to consider the Hubble parameter H and the scale factor a, described by the Friedmann equation

$$H = \frac{\dot{a}}{a} = H_0\left[\Omega_M a^{-3} + (1 - \Omega_M - \Omega_\Lambda)a^{-2} + \Omega_\Lambda\right]^{\frac{1}{2}} \tag{2.120}$$

where Ω_M and Ω_Λ are the density parameters for matter (dark and baryonic) and dark energy. The comoving frame complicates the integration, as H and a enter the equations of motion (for details see [58]), and so to derive a symplectic integrator, we start from the Lagrangian for the particle motion in the comoving frame (where the position is denoted as $\boldsymbol{r}'$ and the velocity as $\boldsymbol{v}'$)

$$\mathcal{L} = \frac{1}{2}(a\boldsymbol{v}' + \dot{a}\boldsymbol{r}')^2 - \phi \tag{2.121}$$

By introducing the time dependent generating function $\psi = \frac{1}{2}a\dot{a}\boldsymbol{r}'^2$, we can transform this into

$$\mathcal{L} = \frac{1}{2}a^2\boldsymbol{v}'^2 - \frac{\phi'}{a} \qquad \phi' = a\phi + \frac{1}{2}\ddot{a}a^2\boldsymbol{r}'^2 \tag{2.122}$$

We can switch to the Hamiltonian formalism, with the canonical momentum to $\boldsymbol{r}'$ given as $\boldsymbol{p}' = a^2\boldsymbol{v}'$, to get the Hamiltonian

$$H = \frac{\boldsymbol{p}'^2}{2a^2} + \frac{\phi'}{a} \tag{2.123}$$

This Hamiltonian is separable and we can write the drift and kick operators as

$$D(\tau) = \boldsymbol{r}'_{t+\tau} = \boldsymbol{r}'_t + \boldsymbol{p}\int_t^{t+\tau}\frac{\mathrm{d}t}{a^2} \tag{2.124}$$

$$K(\tau) = \boldsymbol{p}'_{t+\tau} = \boldsymbol{p}'_t - \boldsymbol{\nabla}'\phi'\int_t^{t+\tau}\frac{\mathrm{d}t}{a} \tag{2.125}$$

and the KDK leapfrog operator as $K\left(\frac{h}{2}\right)D\left(h\right)K\left(\frac{h}{2}\right)$.

3 Implementing SPH in pkdgrav3

In this section, the implementation of SPH into PKDGRAV3 is described.

3.1 The gravity code pkdgrav3

Here, a short description of the relevant aspects of PKDGRAV3 is given. A more in-depth description can be found in [36] and [59]. PKDGRAV3 is the latest version in the line of gravity tree codes initially developed by Joachim Stadel [58]. It was developed to be able to perform cosmology dark matter simulations with trillions of particles on hundreds of super-computer nodes in parallel. The force evaluation uses a fast multipole tree code with a binary tree (it used to use a k-D tree in the first version, hence the name PKDGRAV). During domain decomposition, the domain is decomposed into sub-volumes which are local to each processor. Each processor then builds its own local tree. The smallest tree node is called a bucket with a default bucket size of 16 particles.
For the force calculation, the tree is 'walked' starting with the sink cell being the root of the local tree of a processor while the checklist contains the global root cell as the source cell. Tree cells are then recursively opened and interactions are calculated depending on an opening criterion (described in Section 3.4) with one of four different interaction types:

- PP: Direct softened particle-particle interaction
- PC: Multipole expansion of the mass in a distant cell
- CP: Multipole expansion of the potential inside the sink cell induced by a single source particle
- CC: Approximating the potential landscape inside the sink cell induced by the source cell

On the sink side, the tree walk does not proceed down to bucket level, but is stopped at the first cell that contains fewer particles than the group size. The group size is 64 by default. The group and bucket sizes are chosen to maximize performance, and choosing a bucket size larger than one also saves memory by decreasing the size of the tree.
In cosmological simulations, the size of the simulation domain should be infinite, because otherwise the lack of mass outside will affect the force acting on the particles and result in a collapsing solution. The most common approach to solve this problem is to duplicate the simulation volume in all directions infinitely to generate a periodic boundary. Spectral methods can handle this perfectly by the nature of the fast Fourier transform, but for tree methods like PKDGRAV3, this has to be done explicitly using Ewald summation [60].
The PP interactions for a group of particles are calculated by adding all particles with which these particles directly interact to the interaction list PP (ILP). At the end of the walk for a sink group, the ILP and the particle information of the group are then bundled in an object called a `workParticle`. The interactions between the sink particles and the source particles in the `workParticle` are then calculated in a loop using CPU SIMD (single instruction multiple data) vector instructions (or GPU instructions if available). PC interactions are handled similarly with an interaction list PC (ILC), while CC and CP interactions use a second instance of the ILC to calculate the multipoles during the tree walk. The multipole expansions are generally done to 4th order (hexadecapole), but the local expansion in the CP and CC, as well as the Ewald summation use a 5th order expansion.
The time integration is a leapfrog scheme based on a power-of-two rung structure, where particles are placed on a rung depending on their time step size. Forces are calculated for all particles on the current or higher rung, whereas particles on lower rungs only act as sources. With the new force, the opening kick for the particle velocities of active particles is performed, while the closing kick is performed at the beginning of the next step. With the new velocities, positions are drifted for all particles to the next smallest step.
Parallelization is implemented with a hybrid pthreads/MPI model, where each group of threads has a dedicated MPI thread that handles inter-group communication over MPI, while threads in the same group exchange data using shared memory. On low thread count nodes (like Daint), one such group

would be a node, while on larger thread count nodes (like Eiger) there have to be multiple groups per node to decrease the load on the MPI thread (see discussion in Section 4.3). If the machine has NVIDIA GPUs, they can be used to offload the floating point operations of the PP, PC and Ewald interactions, but not the tree walk. The machine Eiger at CSCS does not have GPU nodes, so only the CPU capability of the code is used there.

3.2 General concept

In Gasoline and the existing SPH implementation in pkdgrav3, SPH related calculations are done using so-called smooths, which means that each processor loops over its active particles, searches for the closest `param.nSmooth` particles to that particle and then executes the necessary operations to calculate the result of the respective smooth (density, intermediate values like $\boldsymbol{v} \cdot \boldsymbol{x}$ or SPH forces). The process of neighbor finding is slow and can take longer than the whole gravity calculation (see Section 4.2). Thus in this work, the neighbor finding is included into the tree walk process of the gravity calculation. All the particles that are needed (and some more) are added to the PP list by modifying the opening criterion (described in Section 3.4). This increases the gravity work as is discussed in Section 4.2, but the increase is of the order of 50 %. The density calculation is done in an additional walk, that has all the gravity related operations disabled, before the force walk, in an operation similar to the PP interaction using the ILP. The SPH accelerations are then calculated in the second walk together with the gravitational accelerations, also as a PP operation. All parameters and flags necessary to control the different types of the tree walk by enabling or disabling certain operations (like the calculation of density, gravity forces or SPH forces) are contained in a structure called `SPHOptions` that is made available at all levels of the code where it is needed.

3.3 The Kernel

The kernel is split into 4 functions defined in `SPHOptions.h` as preprocessor macros. Each contains a switch-case structure to switch between kernel types at run-time (even though only one kernel, the M4 cubic spline kernel, is implemented at this time). Usually, SPH kernels are defined as a function of $\frac{r}{h}$ where h is called the smoothing length. Kernels with compact support have a support of $n \cdot h$ where n is a number of order 1. In this work, the kernel is defined as a function of $q = \frac{r}{f_{Ball}}$ where f_{Ball} is such that the kernel has a support of $q \in [0, 1]$. We thus define

$$W(q) = Cw(q) \qquad q = \frac{r}{f_{Ball}} \tag{3.1}$$

where C is a normalization factor (that depends on the kernel), such that the kernel satisfies

$$\int W(q) \mathrm{d}^3 r = 1 \tag{3.2}$$

The kernel base function of the M4 cubic spline kernel in this formulation is given as

$$w = \begin{cases} 1 - 6q^2 + 6q^3 & 0 \leq q \leq \frac{1}{2} \\ -2(p-1)^3 & \frac{1}{2} < q \leq 1 \\ 0 & q > 1 \end{cases} \qquad C = \frac{8}{f_{Ball}^3 \pi} \tag{3.3}$$

The gradient of the kernel which is needed in the force calculation is given as

$$\boldsymbol{\nabla} W = \frac{\boldsymbol{r}}{r f_{Ball}} C \frac{\mathrm{d}w}{\mathrm{d}q} \tag{3.4}$$

where the derivative of the kernel base function for the M4 kernel with respect to q is given as

$$\frac{\mathrm{d}w(q)}{\mathrm{d}q} = \begin{cases} 18q^2 - 12q & 0 \leq q \leq \frac{1}{2} \\ -6(q-1)^2 & \frac{1}{2} < q \leq 1 \\ 0 & q > 1 \end{cases} \tag{3.5}$$

The derivative of the kernel with respect to f_{Ball} that is needed in the Newton-Raphson iterations of the density calculation (see below) is then calculated using the chain rule as

$$\frac{\mathrm{d}W(q)}{\mathrm{d}f_{Ball}} = -\frac{C}{f_{Ball}}\left(3w(q) + \frac{\mathrm{d}w(q)}{\mathrm{d}q}q\right) \tag{3.6}$$

Equations (3.3), (3.5) and (3.6) have to be recalculated when adding new kernels. In order to use the vector instructions of the CPU or GPU in the functions where the kernel is used, the kernel defines and uses masks in order to do the necessary case distinctions and only uses the SIMD operators defined in `simd.h`. The code snippet below shows the implementation of the kernel functions. In Section 3.6 it is shown how these are called in the density evaluation. Note that in the code, q is called `r` while r^2 is `d2`.

```
/*
 * Initializes the SPH kernel, gives back all masks needed, can calculate
 * temporary variables needed in SPHKERNEL and DSPHKERNEL_DR and
 * calculates the kernel normalization C
 */
#define SPHKERNEL_INIT(r, ifBall, C, t1, mask1, kernelType) { \
    switch(kernelType) { \
    case 0: { \
        mask1 = r < 0.5f; \
        t1 = r - 1.0f; \
        C = 8.0f * M_1_PI * ifBall * ifBall * ifBall; \
        break; } \
    default: assert(0);\
    }\
    }
/*
 * Evaluates the non-normalized kernel function, using the masks and
 * temporary variables initialized in SPHKERNEL_INIT.
 * Has to be normalized with C at the end
 */
#define SPHKERNEL(r, w, t1, t2, t3, r_lt_one, mask1, kernelType) { \
    switch(kernelType) { \
    case 0: { \
        t2 = 1.0f + 6.0f * r * r * t1; \
        t3 = - 2.0f * t1 * t1 * t1; \
        w = maskz_mov(r_lt_one,t3); \
        w = mask_mov(w,mask1,t2); \
        break; } \
    default: assert(0);\
    }\
    }
/*
 * Evaluates the derivative of the non-normalized kernel function with
 * respect to r, using the masks and temporary variables initialized in
 * SPHKERNEL_INIT. Has to be normalized with C at the end
 */
#define DSPHKERNEL_DR(r, dwdr, t1, t2, t3, r_lt_one, mask1, kernelType) { \
    switch(kernelType) { \
    case 0: { \
        t2 = 6.0f * r * (3.0f * r - 2.0f); \
        t3 = - 6.0f * t1 * t1; \
        dwdr = maskz_mov(r_lt_one,t3); \
        dwdr = mask_mov(dwdr,mask1,t2); \
        break; } \
    default: assert(0);\
```

```
    }\
    }
/*
 * Calculates the derivative of the normalized kernel function with respect
 * to fball. Do not normalize, is already normalized.
 */
#define DSPHKERNEL_DFBALL(r, ifBall, w, dwdr, C, dWdfball, kernelType) { \
    switch(kernelType) { \
    case 0: { \
        dWdfball = - C * ifBall * (3.0f * w + dwdr * r); \
        break; } \
    default: assert(0);\
    }\
    }
```

3.4 Opening criterion

The opening criterion implements a decision tree to decide whether the gravitational forces between two cells are calculated as Particle-Particle, Particle-Cell, Cell-Particle or Cell-Cell interactions.

3.4.1 Decision tree

In Figure 2, the decision tree of the opening criterion is shown. The sink cell for which the force has to be calculated is called `k`, while the source cell is called `c`. The outcome of the opening criterion is calculated for a single sink cell and a whole SIMD vector of source cells at a time. The meaning of the different outcomes are as follows:

- iOpen = 0: `c` stays on the checklist and is visited again later with a smaller `k` cell
- iOpen = 1: All particles of `c` are added to the PP list, end of the process for c
- iOpen = 2: `c` is a bucket which is opened and each particle is added as a cell with zero size, the particle can end up as a CP or PP interaction
- iOpen = 3: `c` is opened and the children are added to the checklist
- iOpen = 4: The multipole expansion is used, end of the process for `c`: PC interaction
- iOpen = 8: The local expansion is used, end of the process for `c`, CC or CP interaction
- iOpen = 10: `c` is empty, and removed, end of the process for `c`

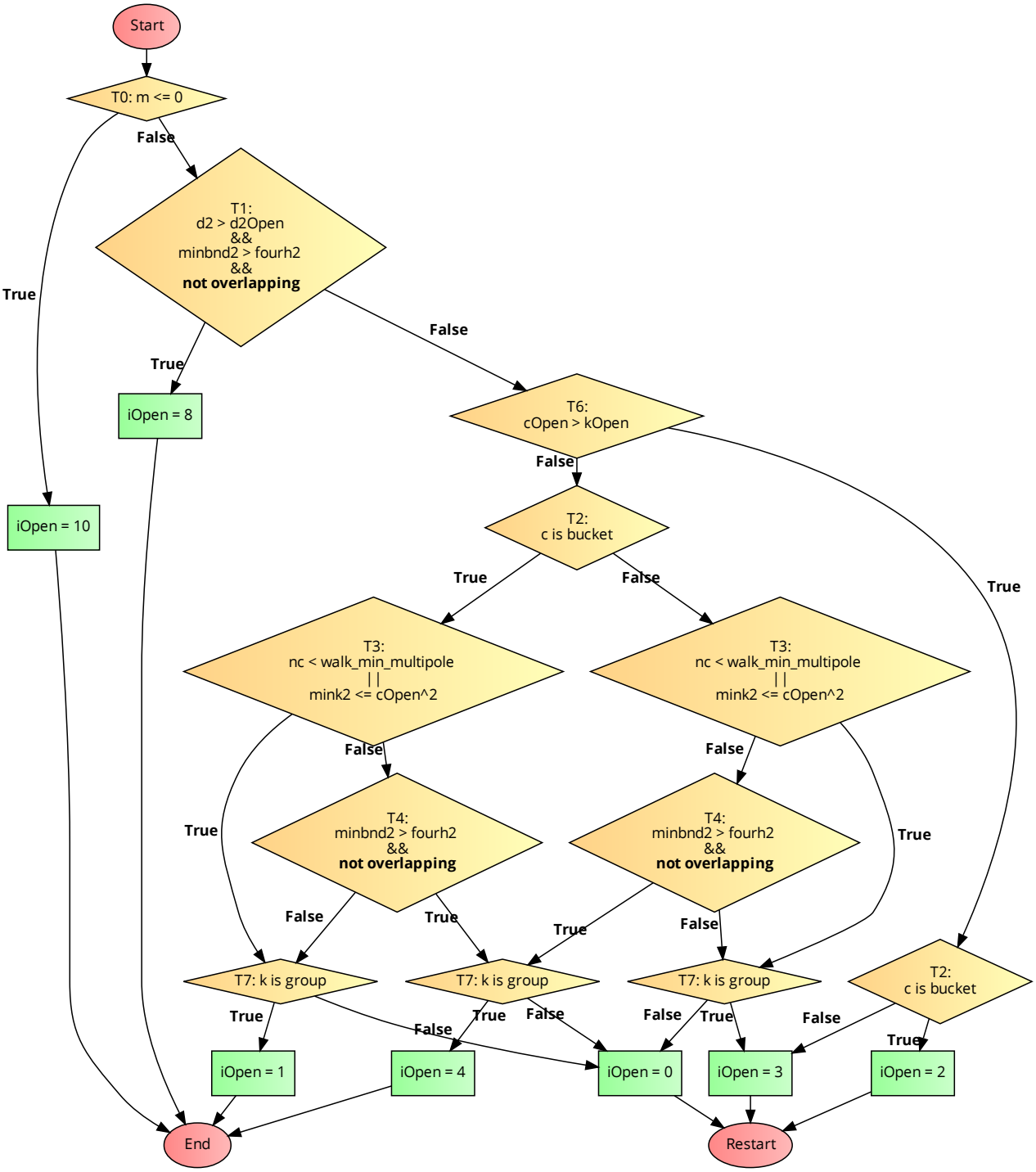

Figure 2: Decision tree of the opening criterion. The sink cell for which the force is to be calculated is called `k`, while the source cell is called `c`. The new SPH overlap tests are highlighted in bold.

3.4.2 SPH criterion

In order to make sure that any source cell particle that is inside the kernel of any of the active sink cell particles (gather) or has any active sink cell particle in its own kernel (scatter) ends up on the PP list, two additional criteria are added. Scatter interactions are only used for the forces calculation, as particles do not scatter density. These two criteria are overlap tests between either a box and a ball (Ball of Balls) or two boxes (Box of Balls) described below.

- The first way to select the necessary particle uses a Ball of Balls which is calculated during the

tree building (or updating) process. On bucket level, the particles are added one by one and the radius and center of the new Ball of Balls is calculated such that it fully encompasses the old Ball of Balls and the ball of the newly added particle. Cells that are not buckets have their Ball of Balls calculated such that it encompasses the Ball of Balls of their two children. In the opening criterion, the decision to include the particles of a source cell on the PP list is based on if the source cell overlaps with the Ball of Balls of the sink cell for the gather particles and vice-versa for the scatter particles.

- In the second way, a Box of Balls is used. During the tree build or update process, for each bucket the minimum and maximum coordinate value of the particle positions plus or minus the respective ball size is calculated. For non-bucket cells, the box is calculated such that it encompasses the boxes of its two children. In the opening criterion, an overlap test between the Box of Balls of the sink cell and the cell bounds of the source cell is done for the gather particles and vice-versa for the scatter particles.

In Figure 3, the two variants are shown for an example configuration of a sink cell k and four source cells c_x. Which one of these two ways is used can be chosen using a preprocessor define in `SPHOptions.h`. The overlap tests for either the Ball of Balls or Box of Balls are added to the tests T1 and T4, such that every interaction either ends up on the PP list or passes a no-overlap test at least once.

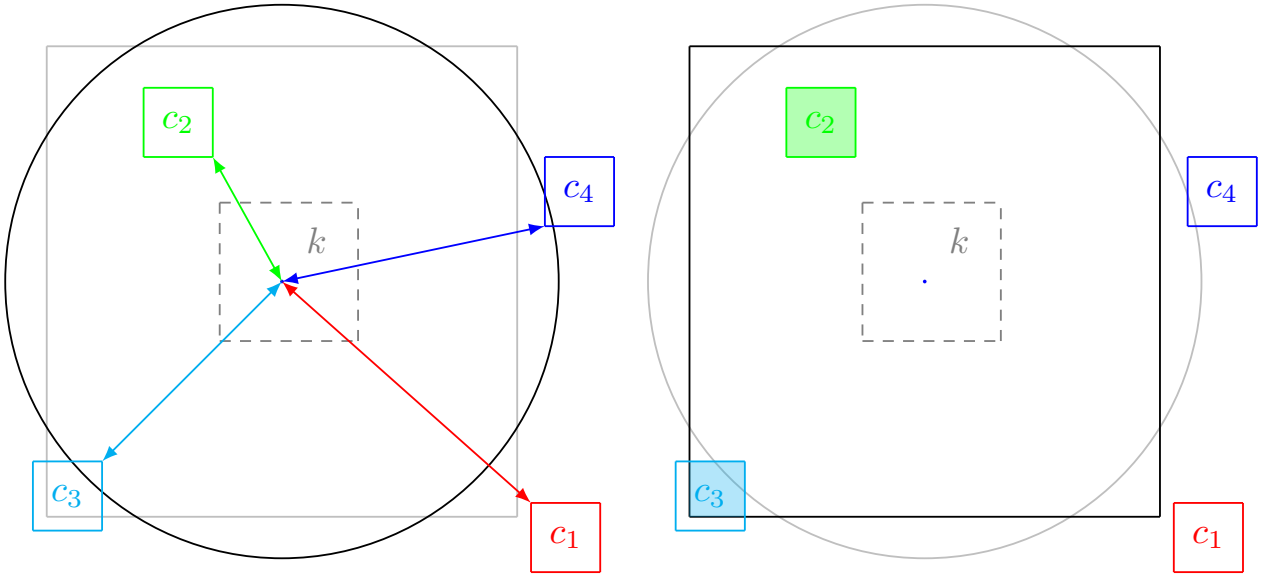

Figure 3: Selection of cells c_x for the ILP of group k in the Ball of Balls case (left) and the Box of Balls case (right). The Ball of Balls variant selects cells c_2, c_3 and c_4 while the Box of Balls variant selects only cells c_2 and c_3.

3.4.2.1 Comparison In tests, it has been observed that using the Box of Balls selects around 30 % less particles on average than using the Ball of Balls. In Figure 4 the Ball of Balls and Box of Balls for 16 random balls (coordinates random $\in [-0.5, 0.5]$ and radii random $\in [1.0, 1.5]$) are shown and one can see that the Ball of Balls reaches farther than the Box of Balls along the coordinate axes. It thus makes intuitive sense, why in the coordinate aligned tree structure, more particles could get selected when the Ball of Balls is used.

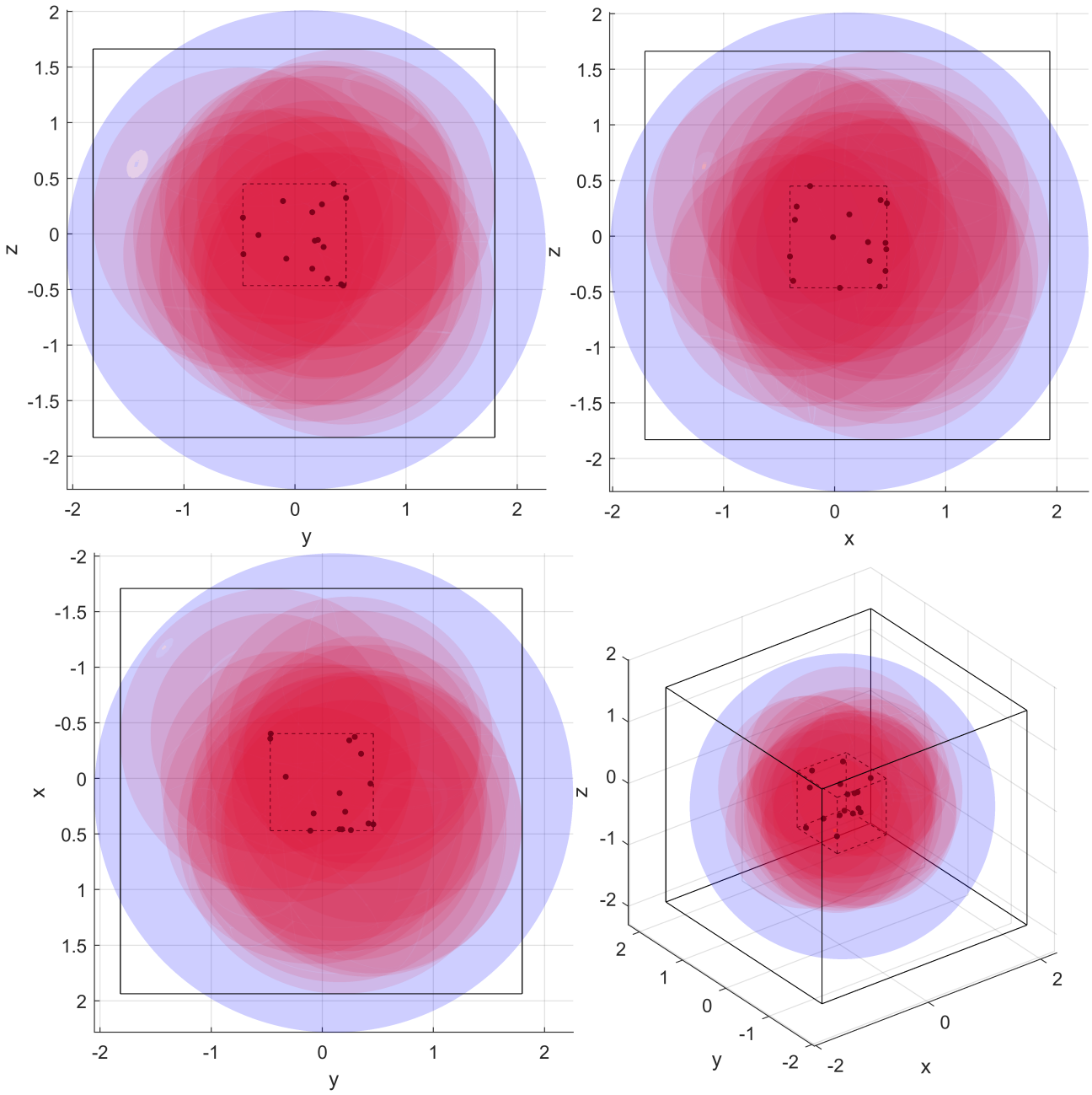

Figure 4: Example of 16 random balls (red), centered on the dark points in a cell (dashed) and the corresponding Ball of Balls (blue) and Box of Balls (black). Due to the way the Ball of Balls is created, only one of the balls is touching it (can be seen as the artifact in all figures).

3.5 Initialization

After reading the initial conditions and setting up everything for gravity calculations, three additional steps have to be performed, before time-stepping can begin.

3.5.1 Kernel target

The method to find the smoothing length described in Section 2.3.2.2 needs a kernel target, either the mass or the particle number in the kernel. This target is set as either `param.nSmooth` (the number of smoothing neighbors used in the old SPH implementation) when the particle number density is used,

or

$$\text{Target} = \frac{M_{tot}}{N_{tot}} N_{smooth} \tag{3.7}$$

when the mass density is used. This value is then set to the parameter structure, from where it is set in each initialization of the `SPHOptions` structure, so that it is always available when needed.

3.5.2 Ball size

The value f_{Ball} has to be initialized too, because it enters the size of the Ball of Balls or Box of Balls described above. So, to set f_{Ball}, an empty or ball smooth is performed with the smoothing neighbors set to twice `param.nSmooth`. This smooth only searches for the nearest particles, calculates f_{Ball} and saves this value to the particle memory. No additional calculations are performed.

3.5.3 Thermodynamical variable

The thermodynamical variable that is read from the initial condition file is the temperature in Kelvin, but the thermodynamical variable used by the code is either the specific internal energy u or the entropic function $A(s)$ described in 2.3.2.8. So, during reading the initial condition, the temperature is converted to specific internal energy following

$$u = T \frac{R}{(\gamma - 1)\mu} \tag{3.8}$$

where R is the gas constant, γ is the adiabatic index, and μ the mean molar weight of the gas. As the relation between $A(s)$ and u in Equation (2.90) also contains the density, the conversion can only be done once the density is known. Thus, a density walk is performed which then directly converts from u to $A(s)$ using Equation (2.90).

3.6 Density calculation

In the density walk, only gather particles are selected in the opening criterion, but a small factor ≥ 1 is applied to the individual ball sizes of the particles during the tree build. This makes sure that if a particle increases its ball size during the density calculation, all needed particles are on the PP list. In the density walk, Newton-Raphson iterations are applied to solve for the density ρ and the ball size f_{Ball} such that they satisfy either Equations (2.55) or (2.56) depending on a flag. This is done by first doing a single density evaluation that also calculates the derivative of the density and the number density with respect to the ball size. If the deviation from the kernel target is too large for any of the particles in the group, new ball sizes are calculated for all active particles in the group, using a modified Newton-Raphson method

$$f_{Ball}^{new} = f_{Ball} - \frac{\delta}{\frac{\mathrm{d}\delta}{\mathrm{d}f_{Ball}}} \tag{3.9}$$

where δ is given by either Equation (2.55) or (2.56) minus the kernel target (ρ denotes either mass or particle number density)

$$\delta = \frac{4}{3}\pi f_{Ball}^3 \rho - \text{Target} \tag{3.10}$$

and $\frac{\mathrm{d}\delta}{\mathrm{d}f_{Ball}}$ is calculated from the derivatives

$$\frac{\mathrm{d}\delta}{\mathrm{d}f_{Ball}} = \frac{4}{3}\pi f_{Ball}^2 \rho + \frac{4}{3}\pi f_{Ball}^3 \frac{\mathrm{d}\rho}{\mathrm{d}f_{Ball}} \tag{3.11}$$

This new value is then limited to be $\in [0.5, 1.5] f_{Ball}$ to ensure stability of the solution process. A second limiter limits the number of particle in the kernel to ten times `param.nSmooth`, in order to

reduce the huge PP list sizes that can occur when extremely low density particles dominate the Ball of Balls or Box of Balls for some groups. The `workParticle` is then sent to do a new density evaluation and the process is repeated until convergence. Once converged, the values of the density ρ, the ball size f_{Ball} and the grad-h correction Ω (see Equation (2.49)) are extracted from the `workParticle` and saved to the particle memory. The code snippet below shows how the kernel functions are called and the (number-) density and its derivative is calculated.

```
// Calculate distance
dx = Idx + Pdx;
dy = Idy + Pdy;
dz = Idz + Pdz;
d2 = dx*dx + dy*dy + dz*dz;
// Calculate inverse fBall and scaled distance
ifBall = 1.0f / fBall;
r = sqrt(d2) * ifBall;
// Evaluate the kernel
SPHKERNEL_INIT(r, ifBall, C, t1, mask1, kernelType);
SPHKERNEL(r, w, t1, t2, t3, r_lt_one, mask1, kernelType);
DSPHKERNEL_DR(r, dwdr, t1, t2, t3, r_lt_one, mask1, kernelType);
DSPHKERNEL_DFBALL(r, ifBall, w, dwdr, C, dWdfball, kernelType);
// return the density
anden = C * w;
arho = Im * anden;
// return the density derivative
adndendfball = dWdfball;
adrhodfball = Im * adndendfball;
// return the number of particles used
anSmooth = maskz_mov(r_lt_one,1.0f);
```

3.7 FastGas

As an updated density is needed not only for active particles, but for all particles that are inside the kernel of an active particle, the density update is generally done for all particles. But for rungs on which only a small fraction of the particles is active, it is beneficial to calculate the density only for those particles that end up on the PP list of an active particle. To facilitate this, a third walk is added in front of the density walk with the same opening criterion settings as the force walk, but which only sets a marking flag on any particle that it would add to the PP list without actually adding anything and without doing any calculations besides the walk. The density walk then considers active particles and marked particles as active and calculates the density for all these particles. The force calculation is then done normally, for all active particles. The threshold below which this is done is exposed to the parameter file. At the moment, a reasonable value seems to be 10 %. In the rung distribution plots in Section 4.5 and 4.6, this threshold is indicated and when comparing them with the step wallclock time plots, one can see sharp drops or rises, whenever a rung crosses it.

3.8 Equation of State

At the moment, the only supported equation of state is the ideal gas EOS

$$P = (\gamma - 1)\rho u \qquad c = \sqrt{\gamma(\gamma - 1)u} \tag{3.12}$$

in two different formulations. One can switch at runtime between the formulation with the internal energy u as the thermodynamical variable and the adiabatic formulation described in Section 2.3.2.8. The EOS is evaluated when the `workParticle` is assembled, so it does not use particle memory for the pressure and the sound speed but it is evaluated many times more than would be the case if we would save the values. This can and maybe has to be moved into the density walk when more computationally intensive EOS are used or for the interface correction.

3.9 SPH forces calculation

The calculation of the SPH forces is done together in the same walk as the gravity forces, so they can be both kicked together. The SPH force calculation uses the same method as the density calculation. The contributions of each particle on the PP list is calculated using SIMD instructions and the SIMD-ready kernel functions. This step calculates the acceleration (Equation (2.52)), the divergence of the velocity $\boldsymbol{\nabla} \cdot \boldsymbol{v}$, the time derivative of the internal energy (Equations (2.37) and (2.54) for the formulation with internal energy or Equation (2.91) for the entropic formulation) and the maximum time step size for each interaction:

$$\Delta t_{SPH,ij} = \eta_C \min \frac{\frac{f^i_{Ball}}{2}}{(1 + 0.6\alpha)c_i + 0.6\beta \left|\min(0, \mu_{ij})\right|} \tag{3.13}$$

where η_C is the Courant parameter, α and β are the viscosities that also appear in the artificial viscosity described in Section 2.3.1.6 and μ_{ij} is given by Equation (2.36). This ensures that the CFL condition is satisfied and follows the description in [42] and [55]. The actual time step size of a particle is then calculated as the minimum of all the interactions in the kernel and the time step size from the gravity condition

$$\Delta t_i = \min(\min_j(\Delta t_{SPH,ij}), \Delta t_{grav}) \tag{3.14}$$

with the gravity time step

$$\Delta T_{grav} = \eta \sqrt{\frac{\epsilon}{|\boldsymbol{a}|}} \tag{3.15}$$

where $\eta = 0.2$ is the time step parameter, ϵ is the particle softening and $|\boldsymbol{a}|$ is the acceleration magnitude. This time step size then defines the rung on which the particle is for the next step.

3.10 Time integration

Positions and velocities are integrated using a leapfrog scheme. To evaluate the SPH forces, the thermodynamical variable and the velocity are predicted to the current step using a predictor-corrector scheme. In order to keep the memory footprint as small as possible, no predicted values are kept in memory. Thus the prediction is done on-the-fly, whenever a predicted value is needed. As particles get their velocities and thermodynamical variable updated during the forces walk, this prediction has to be done either forward or backward. The necessary kick factors are computed at the beginning of the walk for all rungs, according to Equation (2.125), depending on which fraction of the step the velocity and thermodynamical variable stand in relation to the current fraction of the step. The prediction is then done in the form

$$v_{pred} = v_{saved} + a_{saved}\Delta t \tag{3.16}$$

where the kick factor Δt can be positive (forward prediction) or negative (backward prediction). The backward prediction for the active particles yields the precise result at the current substep, while the forward prediction and the backward prediction for inactive particles gives the result with the precision of the predictor-corrector scheme. With this, a small execution order dependence is introduced, but the precision of the prediction is either the same as in the predictor-corrector scheme or better. In Figure 5 the operations in a single step (without substeps) is shown. First, the density update is done for all active particles and their neighbors. Then the force calculation, the closing kick of the last step and the opening kick of the new step are done in succession, asynchronously for each group of active particles.

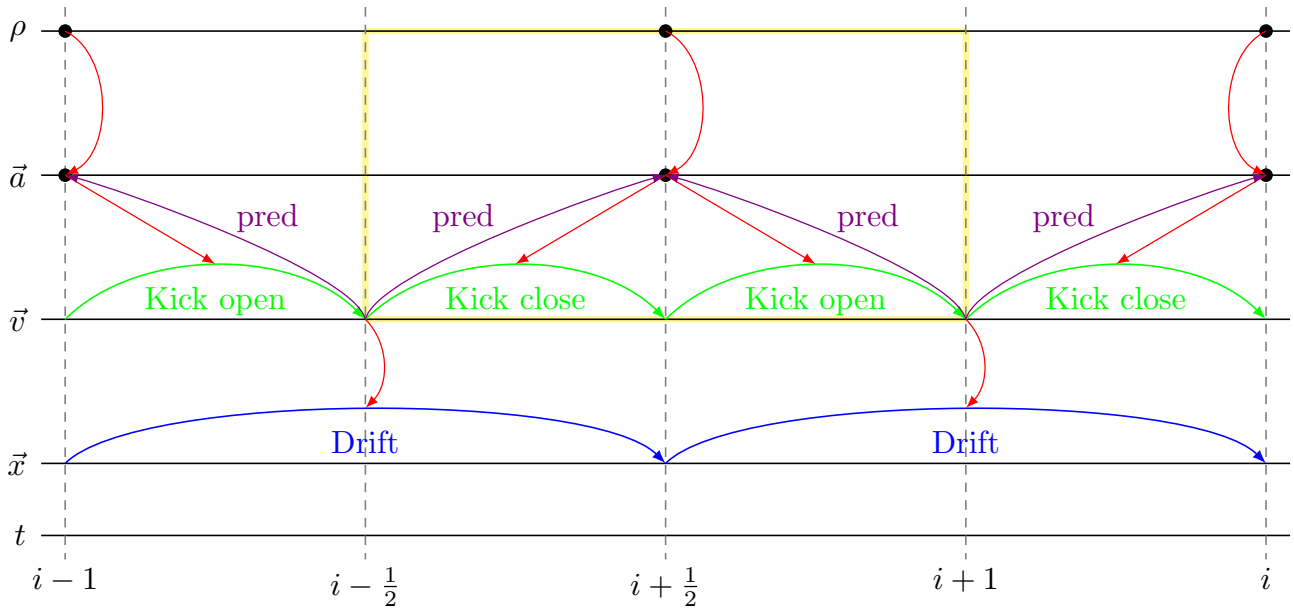

Figure 5: This diagram shows how a single kick operation (indicated by the yellow box) is done. First, the density update is done for all active particles and their neighbors. Then the force calculation, the closing kick of the last step and the opening kick of the new step are done in succession, asynchronously for each group of active particles. The red arrows imply a flow of information, while the violet arrows indicate the on-the-fly forward and backward prediction of the velocity (and thermodynamical variable) described in the text.

4 Results

4.1 Relaxation test

As a first test of the new SPH code, an ideal gas ball ($\gamma = \frac{5}{3}$) of one Jupiter mass with 1×10^6 particles was generated using BALLIC [2]. This initial condition was then relaxed for 9500 steps using both the new code and GASOLINE. In Figure 6, the resulting minimum, maximum and mean values for the density and the temperature are shown over time. The result of GASOLINE shows visible oscillations in all values, while the new implementation oscillates much less, even though it uses the M4 kernel which has worse stability properties than the Wendland C2 kernel used in GASOLINE [43]. These better results can be attributed to the grad-h terms which remove spurious forces due to the varying smoothing length. The large difference in the minimum values between the two implementations are due to the vacuum surface correction that is implemented in GASOLINE but missing in the new SPH implementation in PKDGRAV3. In Figure 7, the RMS velocity of the gas particles is shown over time. Here too, the new implementation shows much less oscillation than GASOLINE.

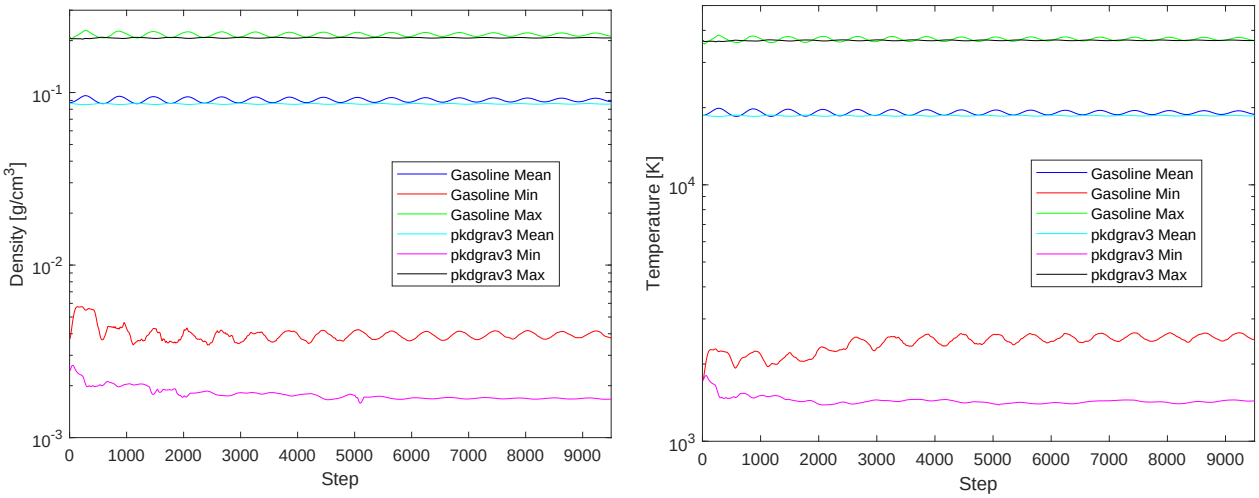

Figure 6: Comparison of the relaxation of a $1\,M_J$ ideal gas ball with GASOLINE and the new SPH implementation in PKDGRAV3. The new code shows less oscillations than the GASOLINE result. The difference in minimum values are due to the missing vacuum surface correction in the new code.

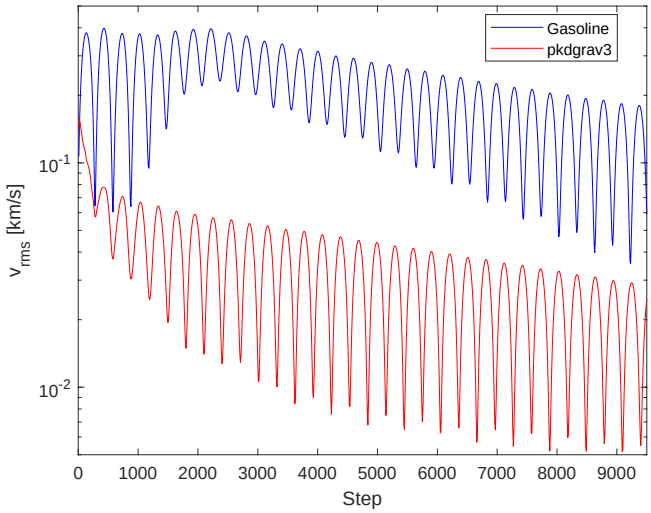

Figure 7: RMS velocities of the two relaxation runs with GASOLINE and the new implementation. The new code shows much less oscillation than GASOLINE.

4.2 Performance compared to pure gravity

As described in 3.2, adding all particles that are needed for the SPH evaluations to the PP list adds gravity work. To assess the extent of this increase, the first step of the collision simulations with 200×10^6 and 2×10^9 particles described in Section 4.5 were analyzed. These tests were performed on Eiger, using a single node in the 1x1x256 configuration (see Section 4.3). The first force evaluation was performed three times. The first evaluation was done with only the gravity opening criterion active and only gravity forces being calculated. The second evaluation then had the SPH opening criterion active, but also only the gravity forces were calculated. The third evaluation had then everything active, both the SPH selections and also the SPH force evaluation. As a comparison, also the density calculation and an empty smooth (meaning, only finding the particles and the ball size, without doing an actual computation like density or forces) were timed. The wallclock times of the different evaluations are shown in Table 2. The additional particles on the PP list add around 40 % to the gravity work, while a full force evaluation takes around twice of what a pure gravity calculation takes. The density walk takes around 80 % of a pure gravity walk, and less than half of a full force calculation, even though Newton-Raphson iterations are performed to solve for the ball size. The empty smooth takes more than twice the time of a pure gravity evaluation, and still more than a full force walk, without doing any actual kernel evaluations.

	200×10^6		2×10^9	
Gravity pure	6.587 s	100 %	67.213 s	100 %
Gravity with SPH selection	8.953 s	136 %	97.494 s	145 %
Gravity and SPH force	12.860 s	195 %	134.339 s	200 %
Density	5.338 s	81 %	54.972 s	82 %
Empty Smooth	15.103 s	229 %	144.509 s	215 %

Table 2: Comparison between wallclock time of pure gravity, gravity with SPH opening criterion, full force calculation, density calculation and an empty smooth for the first step of the 200×10^6 and 2×10^9 particle collision simulations.

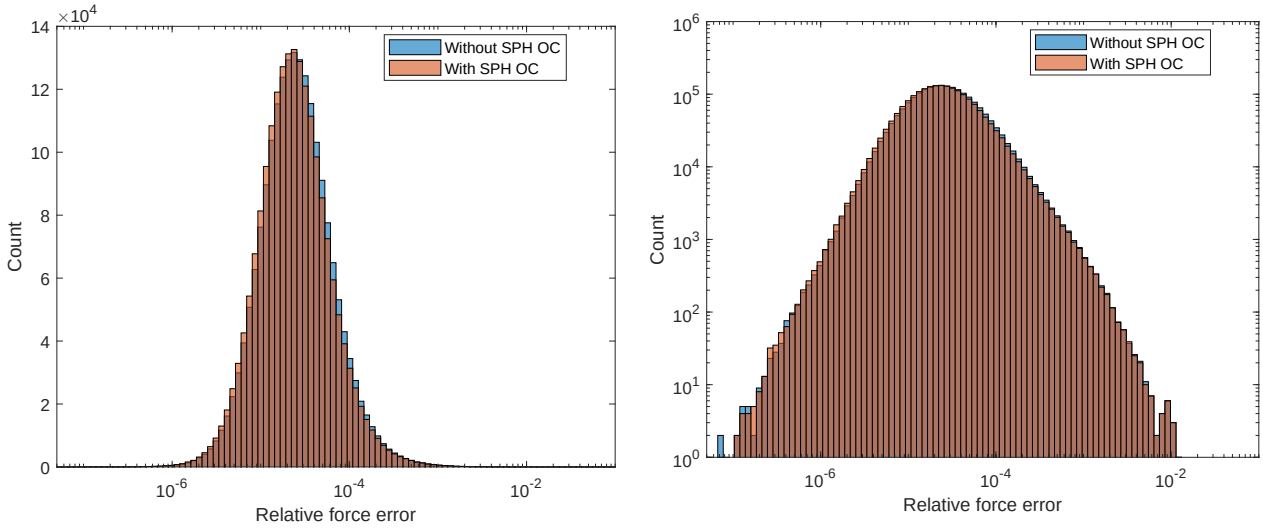

Figure 8: Force error compared to the direct force evaluation for the end result of the 2×10^6 particle collision for the opening criterion with and without the SPH part.

We expect these additional PP particles to increase the accuracy of the gravitational force calculation. To test this, the gravitational accelerations of all particles in the end result of the 2×10^6 particle collision simulation (see Section 4.5) were calculated both with and without using the SPH opening criterion and compared to the accelerations calculated with the direct accelerations. The direct accel-

erations are calculated by forcing all particles onto the PP list, which takes very long as it is $\mathcal{O}(N^2)$ (see Section 2.2.1). In Figure 8, the resulting force error histograms are shown, and one can see that the additional particles decrease the mean force error.

4.3 Multinode scaling

One of the strengths of PKDGRAV3 is its scaling on many supercomputer nodes. As described in Section 3.1, PKDGRAV3 uses a hybrid model of MPI and pthread. To uniquely describe the configuration with which each simulation was run, we use the notation NxMxT where

- N: the number of nodes, between 1 and 32 (it could be more, Eiger has around 500 nodes at the moment)
- M: the number of MPI tasks per node
- T: the number of threads per task
- The total number of threads used by PKDGRAV3 is then $N \times M \times (T-1)$. As an example, 16x16x16 gives 3840 threads.

To benchmark the parallelization performance of the new SPH implementation in PKDGRAV3, $1\,\mathrm{M_J}$ ideal gas balls were generated in the same way as described in Section 4.1 with different numbers of particles. With these balls, collision initial conditions were then created with impact parameter $b = 0.5$ and impact velocity $v_{imp} = 50\,\mathrm{km\,s^{-1}}$ as in Section 4.5. These were then run for 11 steps and the wallclock time of the steps 6 to 10 were averaged (the two bodies have not collided at this point). The ideal configuration for this test turned out to be either 1x1x256 for single node or Nx16x16 for multiple nodes (and for single node, when the particle number is low). On the left side of Figure 9 the strong scaling behavior for the configurations Nx16x16 is shown.

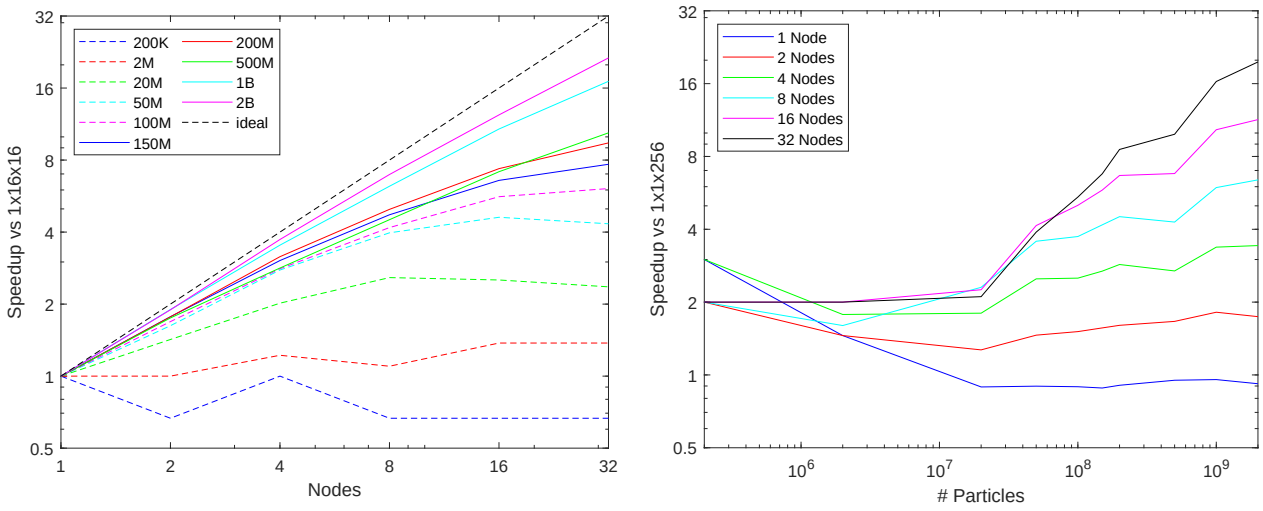

Figure 9: Left: Strong scaling speedup for the Nx16x16 configuration with different particle numbers in the first steps of a collision simulation. For increasing particle numbers, the scaling approaches the ideal scaling. Right: Speedup as a function of the particle number for the same tests, but normalized with the 1x1x256 run. The 1x1x256 configuration is slower than the 1x16x16 configuration for low particle numbers.

The results look as expected, with increasing particle number, the scaling gets better and closer to the ideal scaling. On the right side of Figure 9, the speedup of the Nx16x16 configurations in relation to the 1x1x256 configuration is shown as function of the particle number. It reveals multiple interesting features. First, the 1x16x16 configuration is faster than the 1x1x256 configuration for the 200×10^3 and 2×10^6 particle runs, but slightly slower for larger particle numbers. Secondly, only the single

node and 2 node lines seem to have reached their maximum. So, increasing the particle number will further increase the scaling. But this is not possible at the moment, as we run into limitations in the initial condition generation (see Section 4.7). Lastly, there is a regular pattern with steep increases and then dips in scaling. These are caused by the rung structure of the simulations. The runs where the scaling dips have a low fraction of particles on the lowest rung (12 % and 1 %, for the 100×10^6 and 500×10^6 particle runs respectively), while those that have steep increases have their lowest rung well filled (51 % and 32 % for the 50×10^6 and 200×10^6 particle runs). A low particle fraction means that many threads have no or few active particles, which reduces the scaling, as they sit idle.

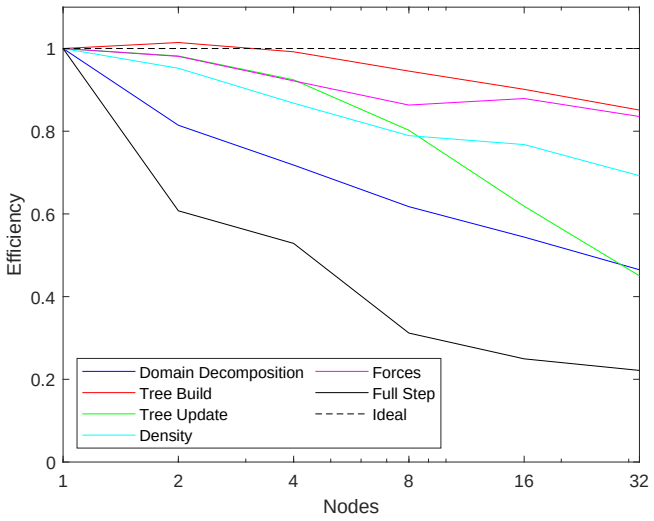

Figure 10: Weak scaling results divided into the parts of a substep. Most of the time is spent in the density (30.1 %) and forces (56.8 %) calculation, while the domain decomposition (9.91 %), tree build (2.76 %) and tree update (0.452 %) have smaller contributions.

Figure 10 shows a weak scaling analysis. For this, the 2×10^9 particle case was run with the 32x16x16 configuration. Additional runs were performed while halving the number of nodes and particles, until a case with 62.5×10^6 particles was run on a single node (this results in 260×10^3 particles per thread). As changing the particle number also changes the rung distribution and the total number of substeps necessary, comparing the wallclock time of a full step is not fair (shown as the black line in the figure). Thus, the mean of the wallclock time for the individual parts of the rung 0 substeps of steps 6 to 10 are shown. The different parts do not contribute evenly to the wallclock time. In the single node simulation, they are split into 9.91 % for the domain decomposition, 2.76 % for the tree build, 0.452 % for the tree update, 30.1 % for the density calculation and 56.8 % for the force calculation. The two largest contributors, the density and force calculation scale reasonably well. As the contribution of the tree update is very small, it is not alarming that it scales worse than the density and force calculation.

4.4 Performance compared to Gasoline

One of the main motivations to do this work was the fact that the tool currently used to perform collision simulations for publications (Gasoline for example in [6, 61]) does not scale well with multiple nodes and is quite slow compared to modern codes. To compare the new implementation to Gasoline, the same test is performed as for pkdgrav3, but only up to 150×10^6 particles, as Gasoline would not start with more than that, but the reason for that is unclear. On the left side of Figure 11, the strong scaling result for Gasoline is shown, and we only get a maximum speedup of 2 at 150×10^6 particles. On the right side of Figure 11 the speedup of the new SPH implementation in pkdgrav3 compared to Gasoline is shown. Even on a single node, the new implementation soon outperforms Gasoline by a factor of around 10, while with 32 nodes, the speedup is 47 at 150×10^6 particles. The speedup would probably increase further with more particles, but we could not test that.

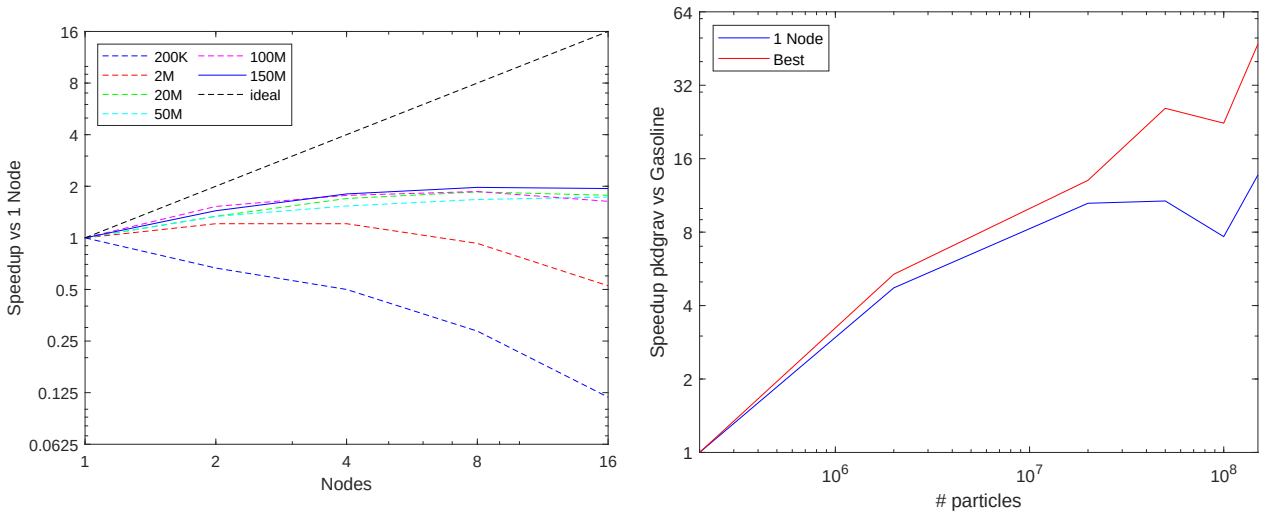

Figure 11: Left: Strong scaling results for GASOLINE. A maximum speedup of 2 can be achieved for 16 nodes with 150×10^6 particles. Right: Speedup for the new implementation compared to GASOLINE. On a single node, the new implementation delivers a speedup of around 10, while with 32 nodes, the speedup is up to 47.

4.5 Collision example

As an example, a collision between two bodies was simulated. For this, an ideal gas ball of one Jupiter mass was created with $\gamma = \frac{5}{3}$ using BALLIC with 1×10^5, 1×10^6, 1×10^7, 1×10^8 and 1×10^9 particles. The 1×10^5 particle case represents the resolution that is currently used to perform production simulations with GASOLINE. Two copies of these balls are then set up for a collision with impact parameter $b = 0.5$ and impact velocity $v_{imp} = 50\,\mathrm{km\,s^{-1}}$ which is $1.5v_{esc}$ for a mutual escape velocity of $33.44\,\mathrm{km\,s^{-1}}$. The collision should thus result in a hit-and-run collision. In Figure 12 some results of the 2×10^5 particle simulation are shown to illustrate the kind of details we can expect at this resolution. In Figures 13 - 15 the same plot with the same settings are shown for the higher resolution simulations. Starting at 2×10^8 particles, the shearing collision interface between the two bodies starts to show (suppressed) fluid mixing instabilities in Figure 14. In Figure 15, the material that is lifted away from the main bodies by the shockwave that has traveled through the body shows structure that is not resolved with less particles.

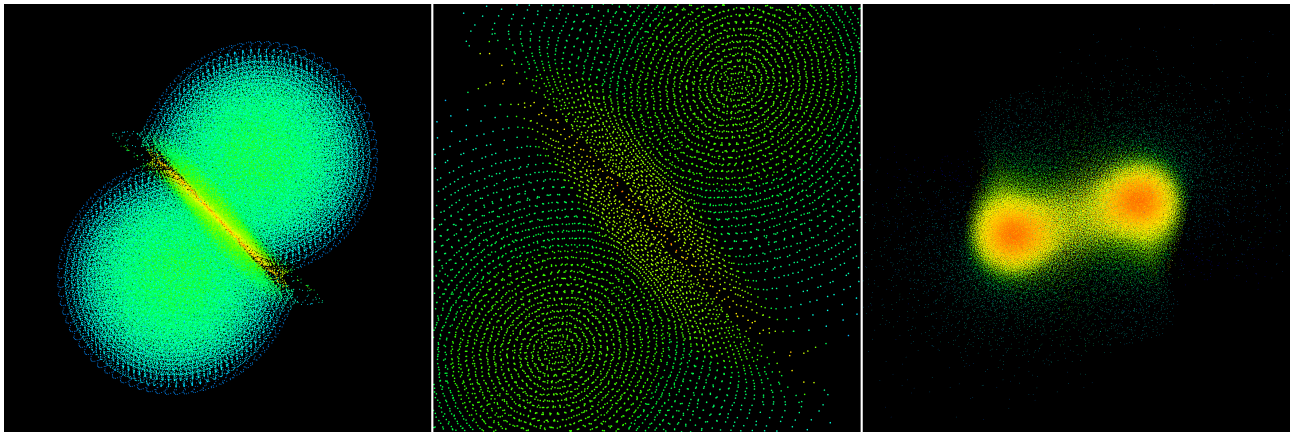

Figure 12: Results from the 2×10^5 particle collision simulation. Left: full temperature plot at step 100, middle: thin slice through the temperature plot at step 100, right: temperature plot at step 600. Logarithmic temperature scale for left and middle figure: $1 \times 10^3\,\mathrm{K}$ (blue) - $1 \times 10^6\,\mathrm{K}$ (red), for right figure: $1 \times 10^2\,\mathrm{K}$ (blue) - $1.585 \times 10^4\,\mathrm{K}$ (red).

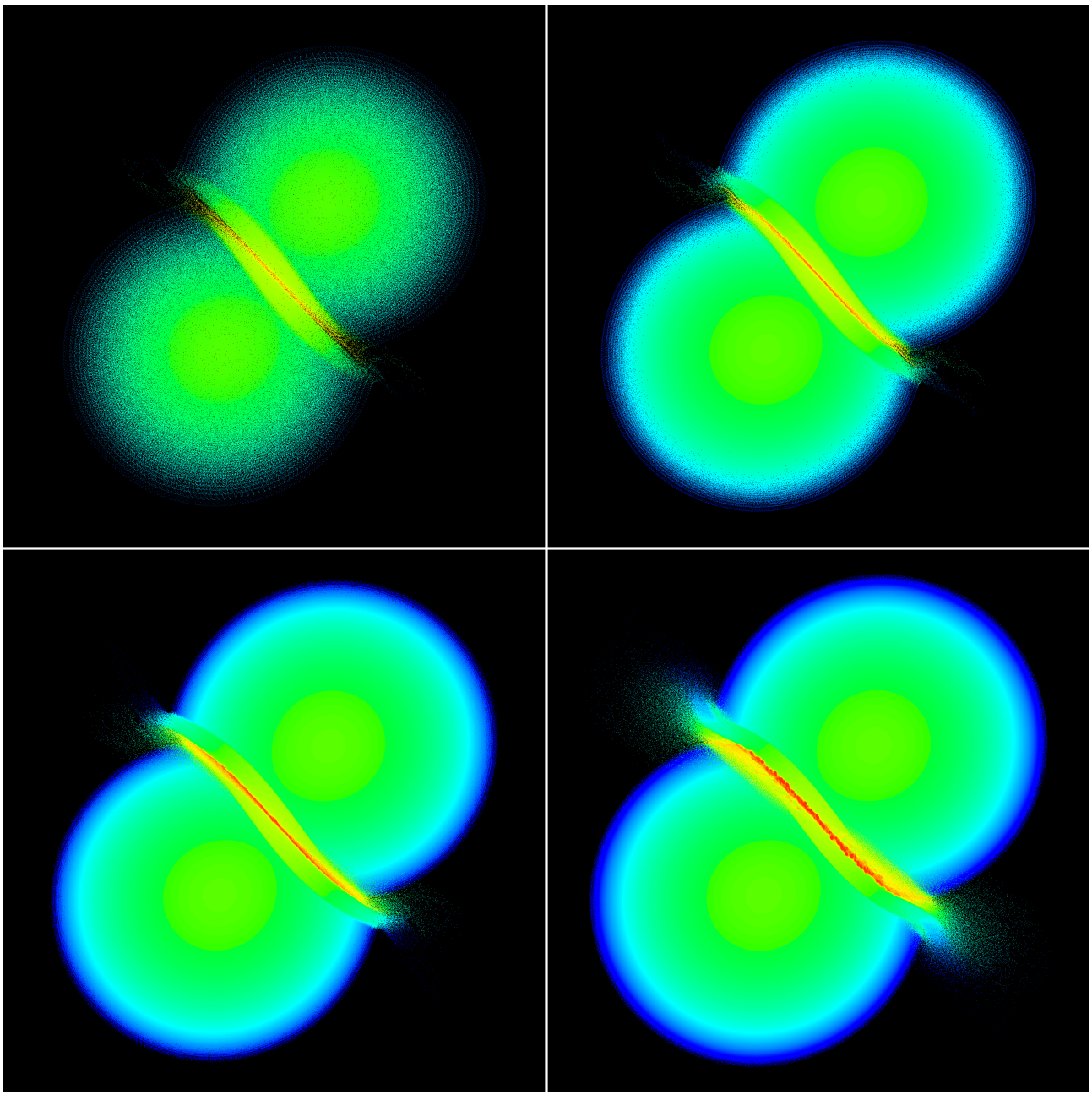

Figure 13: Full view of the collision at step 100 for different particle numbers. Top left 2×10^6, top right 2×10^7, bottom left 2×10^8 and bottom right 2×10^9 particles. Logarithmic temperature scale: 1×10^3 K (blue) - 1×10^6 K (red).

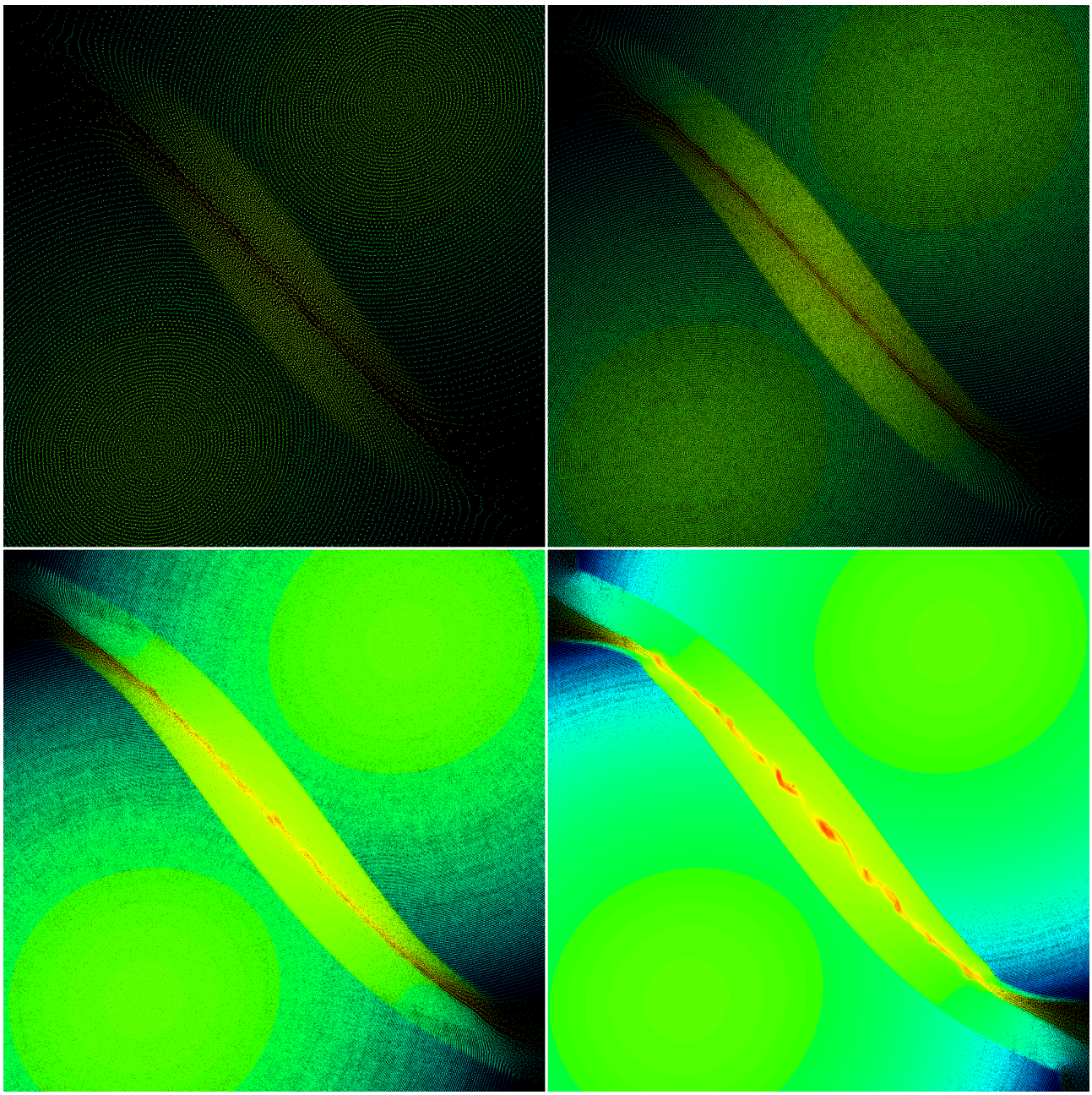

Figure 14: Temperature plot of a thin slice through the collision at step 100 for different resolutions. Top left 2×10^6, top right 2×10^7, bottom left 2×10^8 and bottom right 2×10^9 particles. At higher resolutions, the shearing interface between the two bodies starts to show (SPH suppressed) Kelvin-Helmholtz instabilities. Logarithmic temperature scale: 1×10^3 K (blue) - 1×10^6 K (red).

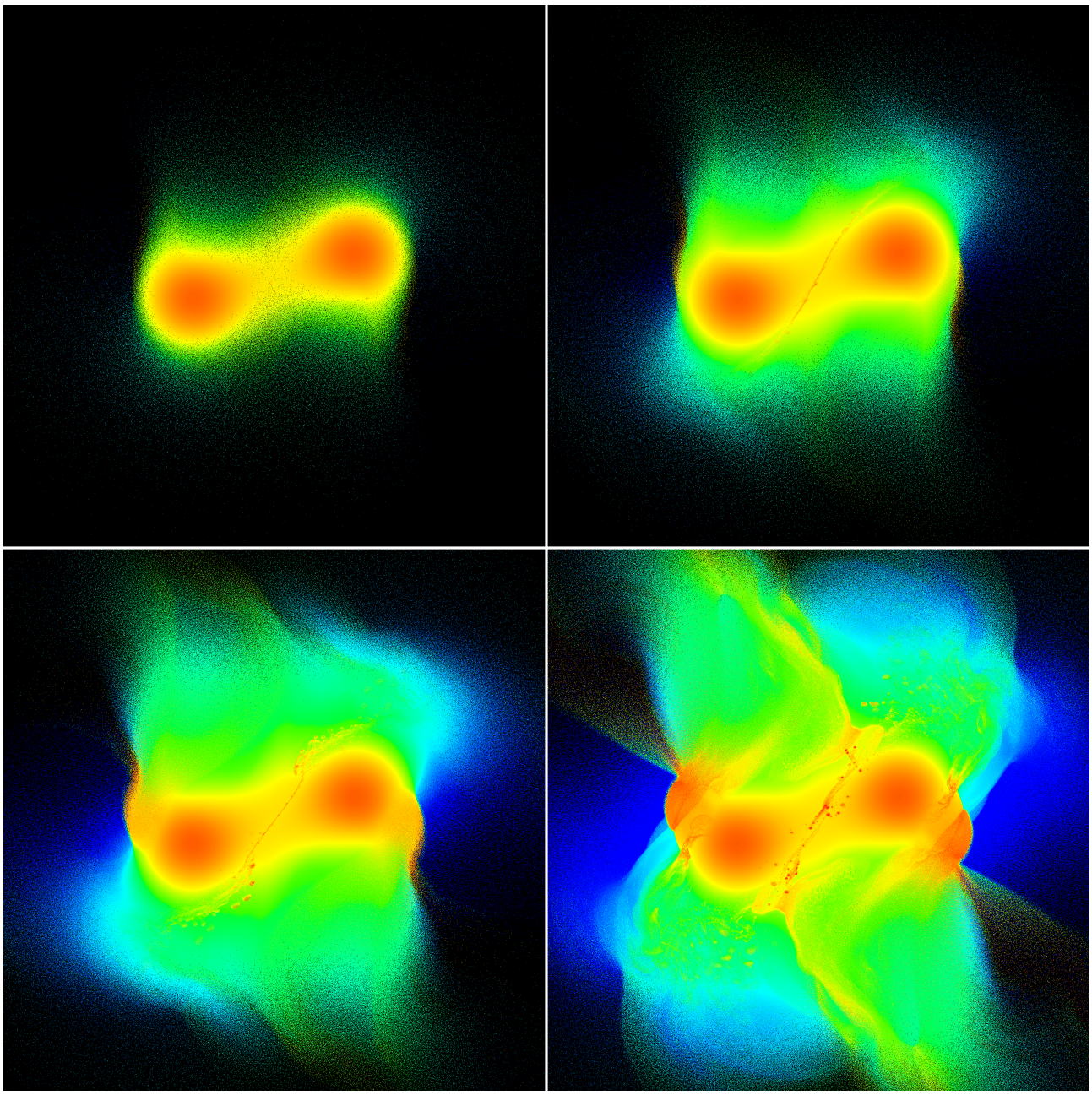

Figure 15: Temperature plot of the result at step 600. Top left 2×10^6, top right 2×10^7, bottom left 2×10^8 and bottom right 2×10^9 particles. As expected, with larger particle count, more and more details are resolved. Logarithmic temperature scale: 1×10^2 K (blue) - 1.585×10^4 K (red).

In Figure 16, the rung distribution and step wallclock time are shown over the course of the simulation with 2×10^8 particles. As expected, the rung distribution shifts towards higher rungs during the actual collision and then towards lower rungs again, once the bodies separate. The step wallclock time follows that, leading to high step times during the collision and low step times when particles go to lower rungs again. There is an increase in step wallclock towards the end of the run, which coincides with an increase of particles on rung 3, caused by the infall of mass that was ejected before, towards the denser remnants.

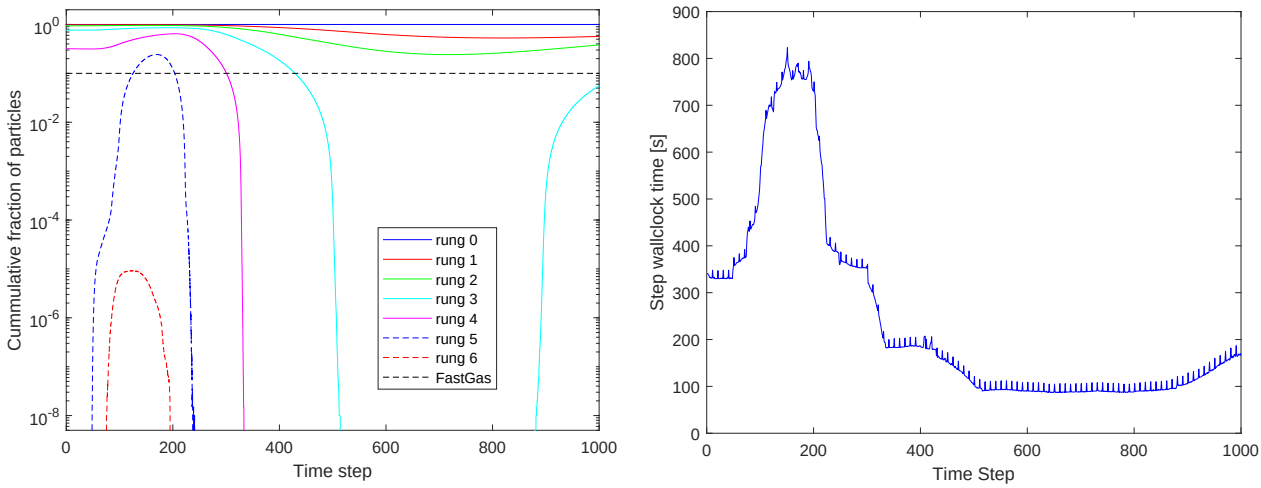

Figure 16: Cumulative particle rung evolution and step wallclock time for the 2×10^8 particle collision simulation. The periodic upwards spikes represent the time steps in which a result was written, which needs an additional density and gravity evaluation.

For the simulation with 2×10^9 particles, the global timestep (corresponding to rung 0) had to be reduced by a factor of ten, because the simulation kept crashing when the shockwave that traveled through the body hit low density gas. The reason for this is that high rung particles suddenly interact with particles that are on a much lower rung and thus a much larger time step size. It was already pointed out by [62] that problems start to occur when the timestep sizes between interacting particles differ by more than a factor of 4. Due to this difference in global time step, the x-axis in Figure 17 is different than in Figure 16 but represents the same physical time. This also causes the maximum rung to be 4 compared to 6 in the 2×10^8 particle simulation, but when using the same global time step, the maximum rung in the 2×10^9 particle simulation would be 8.
In the wallclock diagram in Figure 17 four different runs of the 2×10^9 particle simulation are shown with different configurations. The same initial conditions were simulated with three different parallelization configurations. In the beginning, the step wallclock times show the stacking expected from the scaling analysis presented in Section 4.3, where the 32x8x16 configuration is roughly in the middle between the 16x16x16 and the 32x16x16 configuration. But as the action increases and thus also the amount of communication between the nodes, the 32x16x16 configuration starts to get outpaced by the other two configurations.
A second peculiar feature starts to occur at around step 7000, where a small number of particles with large smoothing lengths in the low density outer regions of the domain can lead the PP list size of single groups to get extremely large, when cells from much denser regions are selected. This happens in all simulations, but only becomes a problem when multiple nodes are used, as the large PP list only occurs for a small number of particles. The amount of additional computations is small, but when the particles that end up on this single large PP list have to be fetched from other nodes, this can massively increase the wallclock time for the whole simulation. In the diagram, we can see that this not only happens in the 32xNx16 simulations, but also in the 16x16x16 simulation. As only half as many nodes are involved, the amount of communication is much smaller and thus the impact is also much smaller.
The fourth line labeled 32x8x16 modified is from an additional simulation where two parts of the code were modified. The first change was the increase of the maximum cache communication data size from 512 bytes to 4096 bytes. This allows the cache to exchange 16 instead of 4 particles and 16 instead of 2 cells in one communication, reducing the total number of communications needed. This leads to a reduction of the step wallclock time of around 11 % (mean value until step 7000). The second modification was to change the ball size limit from the one described in Section 3.6 based on the number of particles in the kernel to a hard limit at 10 code units. Even though this can not be the final solution, it definitely solves the problem at hand, as this sudden increase in PP list size and the

corresponding jump in step wallclock time does not occur in this simulation.

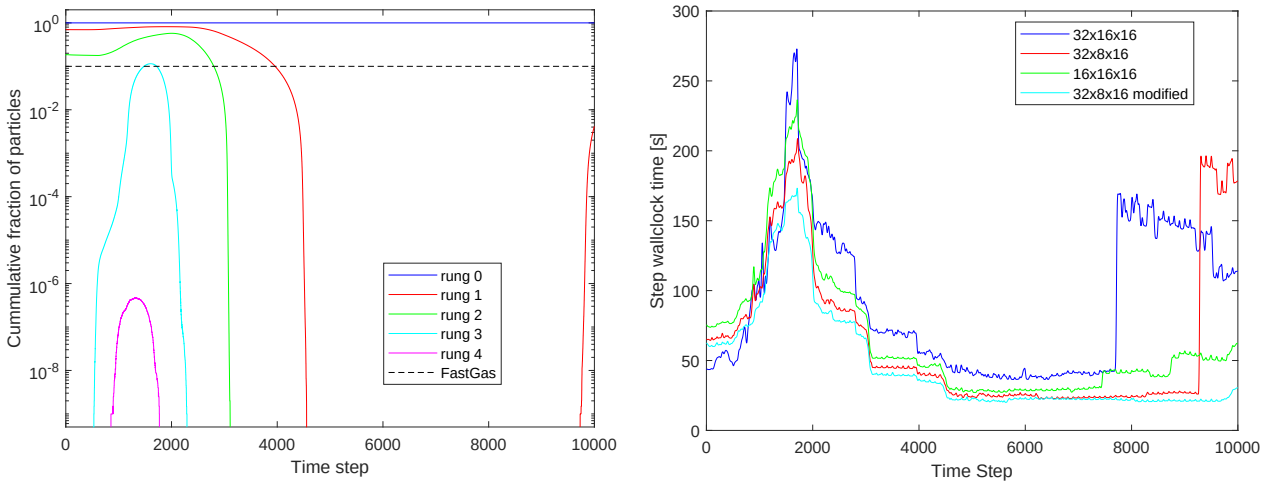

Figure 17: Cumulative particle rung evolution and step wallclock time for the 2×10^9 particle collision simulation. The step wallclock time is smoothed with a Gaussian filter with window size 50. Even though the 32x16x16 thread configuration is faster than the 32x8x16 configuration in the beginning, it looses when the action is more intense. It soon becomes slower than the 16x16x16 configuration. The sudden jumps in step wallclock time at the end are caused by an increase of the PP size of a single thread due to a few particles with extremely large smoothing lenghts.

Two additional simulations were performed with 2×10^8 particles for lower impact velocities of $v_{imp} = 40\,\mathrm{km\,s^{-1}}$ and $v_{imp} = 30\,\mathrm{km\,s^{-1}}$ to generate a recolliding and a merging example. In Figure 18, the same plots as in Figure 15 are shown for these simulations and Figure 19 shows plots from a later point in the simulation.

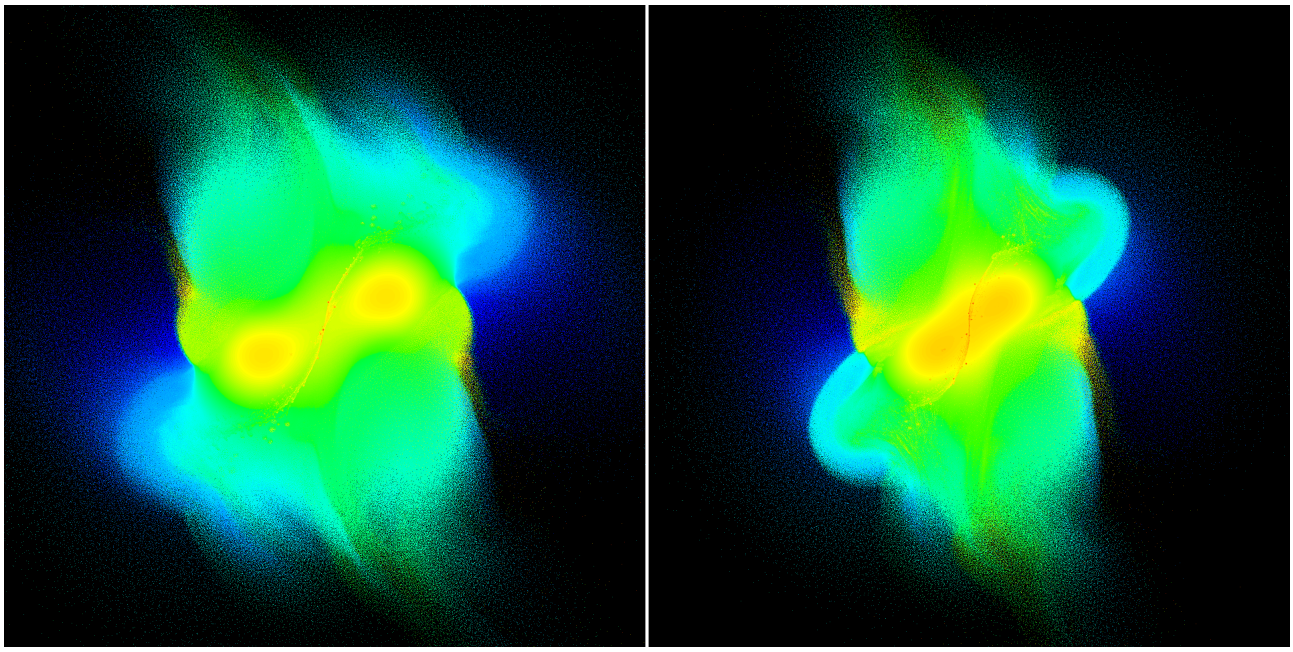

Figure 18: Temperature plots of step 600 for the 2×10^8 particle collision simulations with $v_{imp} = 40\,\mathrm{km\,s^{-1}}$ (left) and $v_{imp} = 30\,\mathrm{km\,s^{-1}}$ (right). Logarithmic temperature scale: $1 \times 10^2\,\mathrm{K}$ (blue) - $1 \times 10^5\,\mathrm{K}$ (red)

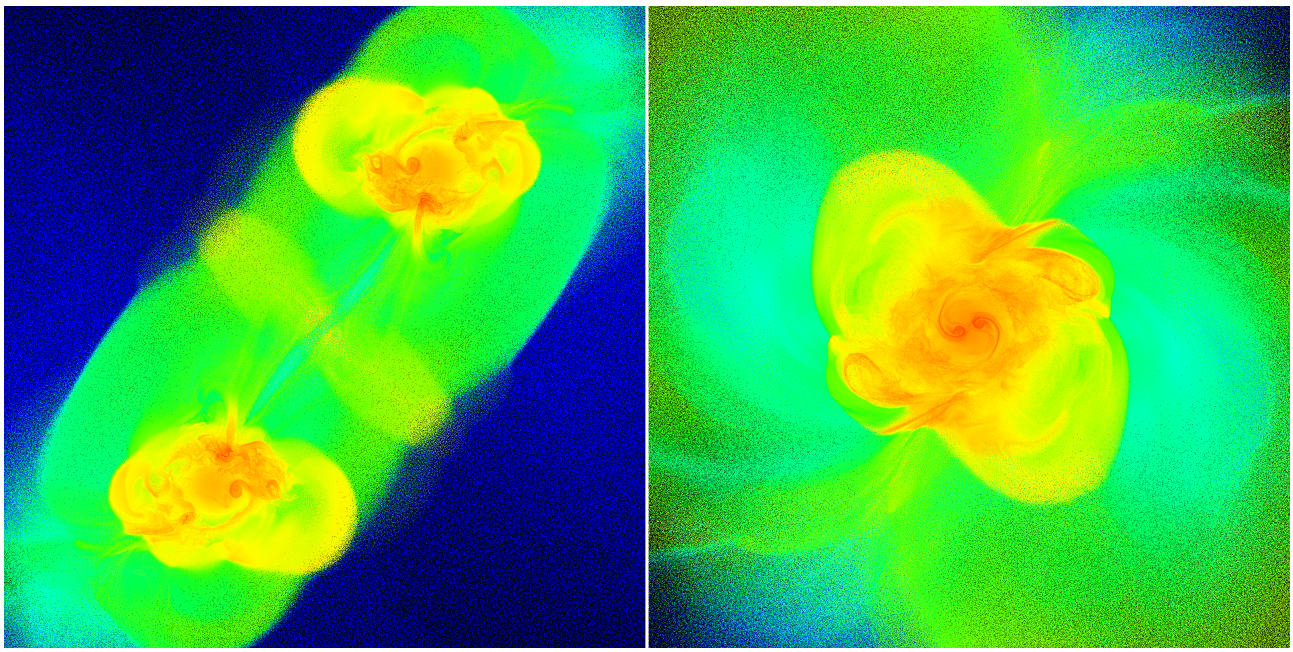

Figure 19: Temperature plots of step 3000 for the 2×10^8 particle collision simulations with $v_{imp} = 40\,\mathrm{km\,s^{-1}}$ (left) and $v_{imp} = 30\,\mathrm{km\,s^{-1}}$ (right). Logarithmic temperature scale: $1 \times 10^2\,\mathrm{K}$ (blue) - $1 \times 10^5\,\mathrm{K}$ (red)

4.6 Cosmology example

As a second example, a cosmology simulation of a 50 Mpc box was done with cosmology parameters $h = 0.67$, $\Omega_0 = 0.32$ and $\Lambda = 0.68$. For this, initial conditions for a dark matter cosmology simulation were generated using the integrated cosmology IC generator of PKDGRAV3 but modified such that instead of dark matter particles, it generates SPH particles for an ideal gas with $\gamma = \frac{5}{3}$ with a temperature of 34 K to correspond to the baryon temperature at redshift $z = 49$. This was done for grids with 128^3, 256^3, 512^3 and 1024^3 particles. These initial conditions were then simulated from $z = 49$ to $z = 0$ with the new SPH implementation. In Figure 20 the density is plotted for the four different resolutions. One can see, that what is just some blob at 128^3 gains more and more structure with increasing resolution. In Figure 21, zooms into the 1024^3 simulation are shown, revealing the detailed structure that emerges at this resolution.

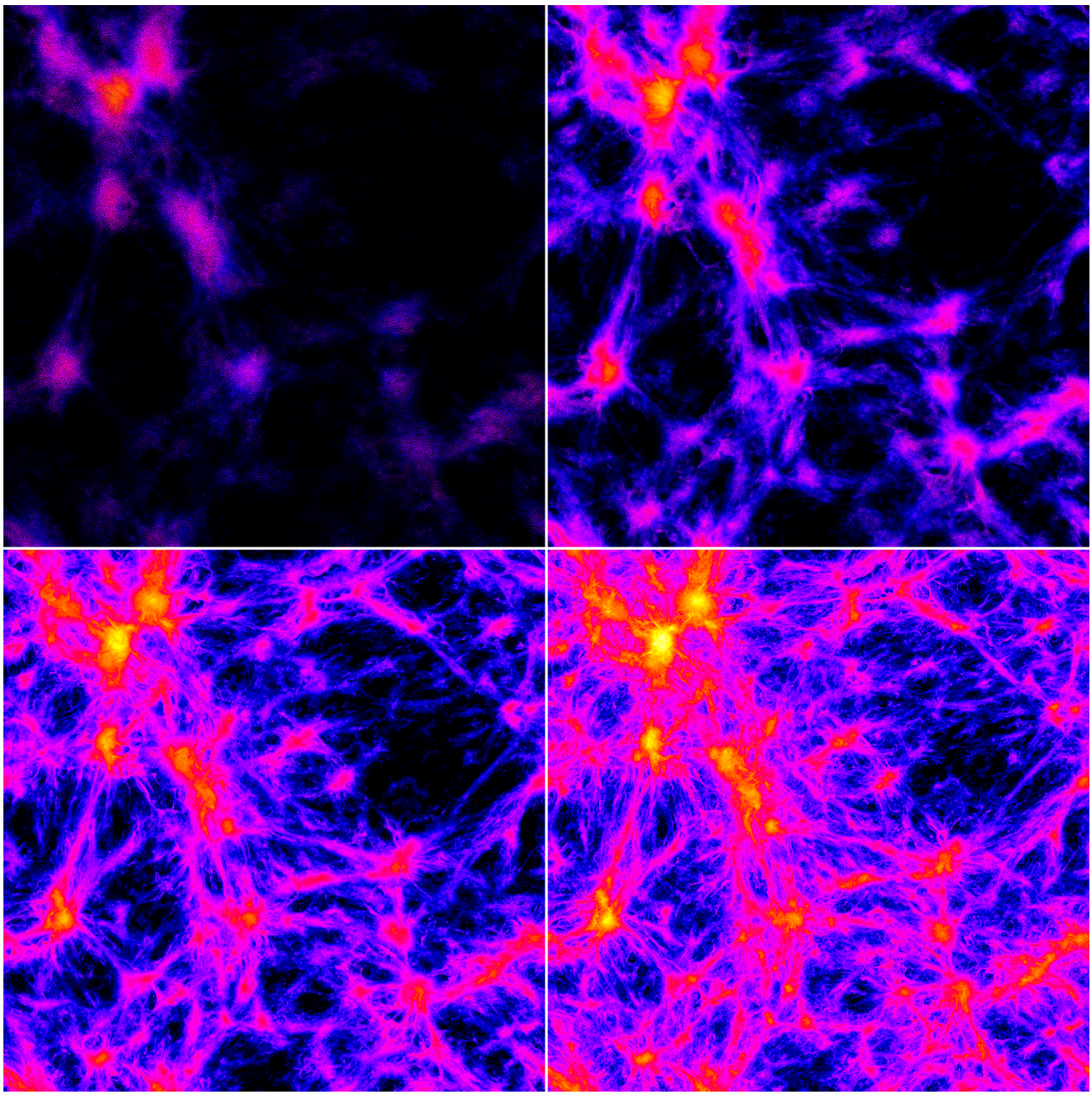

Figure 20: Logarithmic density plot of a 50 Mpc box size cosmology simulation with ideal gas and no dark matter. Top left 128^3, top right 256^3, bottom left 512^3, bottom right 1024^3 particles.

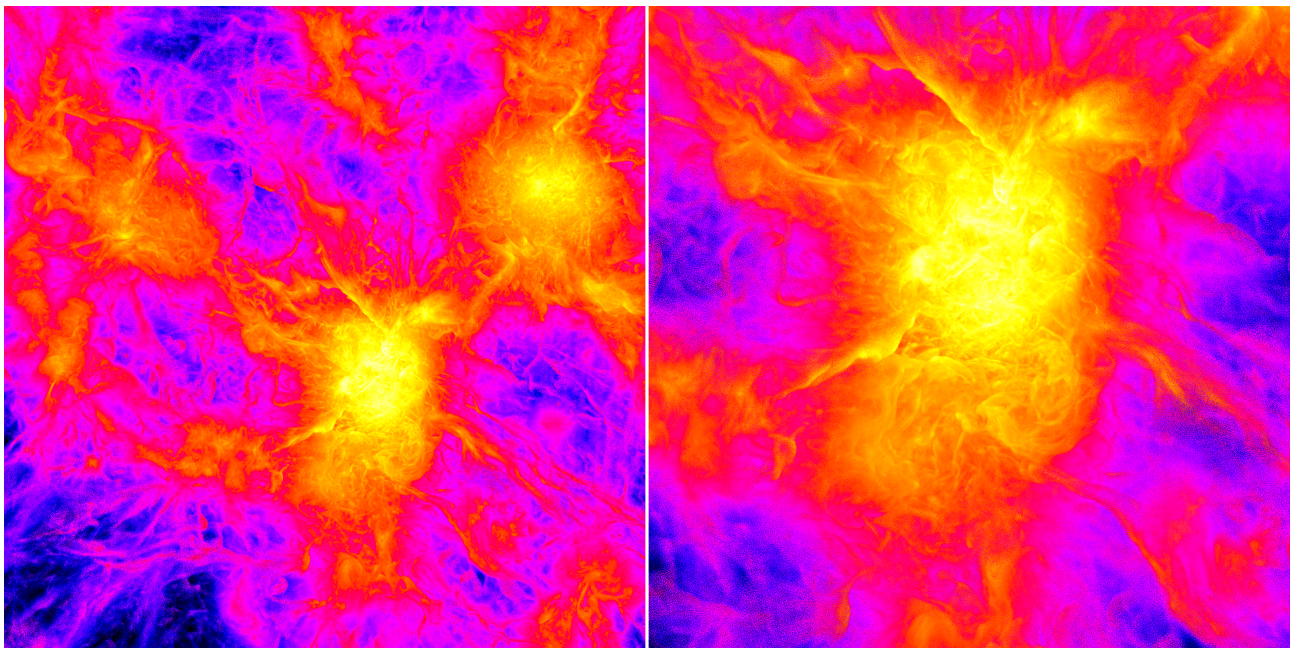

Figure 21: Zoom to the structure in the upper left corner of the 1024^3 particle run (bottom right in Figure 20), left side x4, right side x8.

In Figure 22, the rung distribution and step wallclock time over the course of the 1024^3 particle simulation is shown. One can see that even though the matter is collapsing in comoving coordinates, the work per time step actually gets less over time, as the particles move down in rungs.

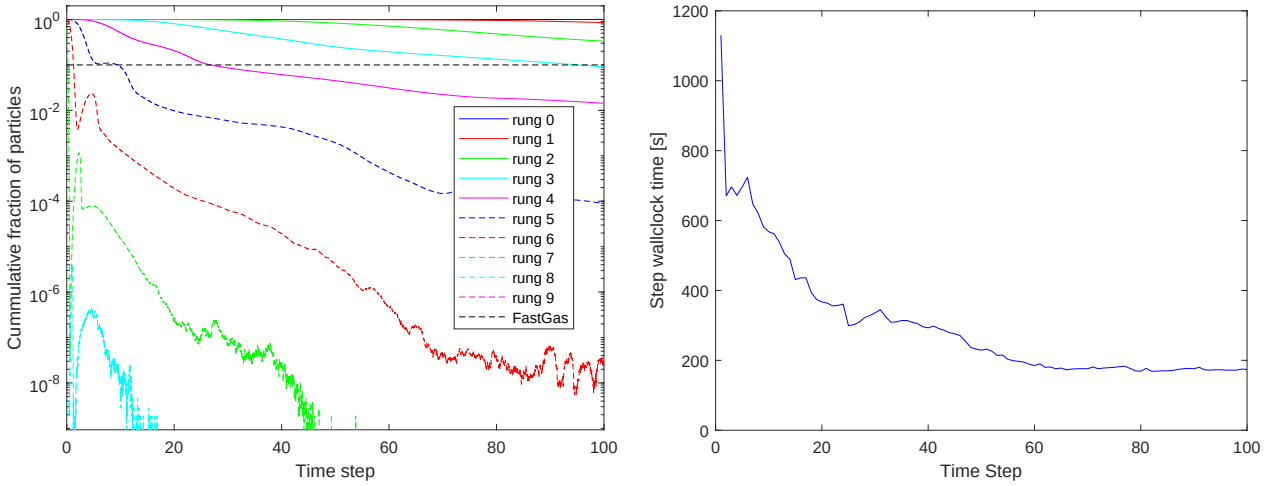

Figure 22: Cumulative particle rung evolution and step wallclock time for the 1024^3 particle simulation run with the 32x16x16 configuration.

4.7 Limitations

The number of particles that we can use in collision simulations is currently limited by the following limits from largest to smallest:

- PKDGRAV3 uses 43 bits of an unsigned 64-bit integer (`uint64_t`) to hold the particle id, which results in an upper limit of $2^{43} = 8\,796\,093\,022\,208$ particles. When running in unordered mode (particles do not have an id, which may not be acceptable in certain use cases, i.e. when the origin of the particle is of interest as for analyzing the mixing in a collision result), the theoretical maximum is $2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615$ as all count variables are `uint64_t`, but only $2^{31} - 1 = 2\,147\,483\,647$ threads can exist with $2^{31} - 1 = 2\,147\,483\,647$ particles, giving a maximum

of $(2^{31}-1)(2^{31}-1) = 4\,611\,686\,014\,132\,420\,609$ particles. As we are far away from such thread counts and amounts of memory, this is irrelevant at the moment.

- The TIPSY file format uses unsigned integers to index particles, which gives a maximum particle count of $2^{32}-1 = 4\,294\,967\,295$
- The program COLLIDE used to create the collision initial condition was initially memory limited as it held onto multiple copies of all particles. This was not a problem before, as particle counts were in the millions. It was modified such that it now uses much less memory, but it uses the `tipsy.c` file to handle TIPSY files, which uses a signed integer for indexing the particles during writing. After changing that to an unsigned integer, it can now also handle up to $2^{32}-1 = 4\,294\,967\,295$ particles.
- The program BALLIC used to create the equilibrium models mostly uses signed integers for particle counts and indices, so it is limited to $2^{31}-1 = 2\,147\,483\,647$ particles.
- The TIPSY program used to postprocess the simulations (create the figures) also uses signed integers to index particles, so the maximum particle count is $2^{31}-1 = 2\,147\,483\,647$. In order to create the picture in Figure 23, the tipsy file was split into quadrants on the XY plane and the 4 pictures were stitched together.

So, the maximum particle count for the existing pipeline is $4\,294\,967\,295$ under ideal conditions (meaning equal particle counts for target and impactor). In Figure 23, a result of such a simulation with 4.2×10^9 particles is shown. In Figure 24 the rung distribution and step wallclock time of this simulation is shown, and we can see that the new implementation is able to handle this without problems.

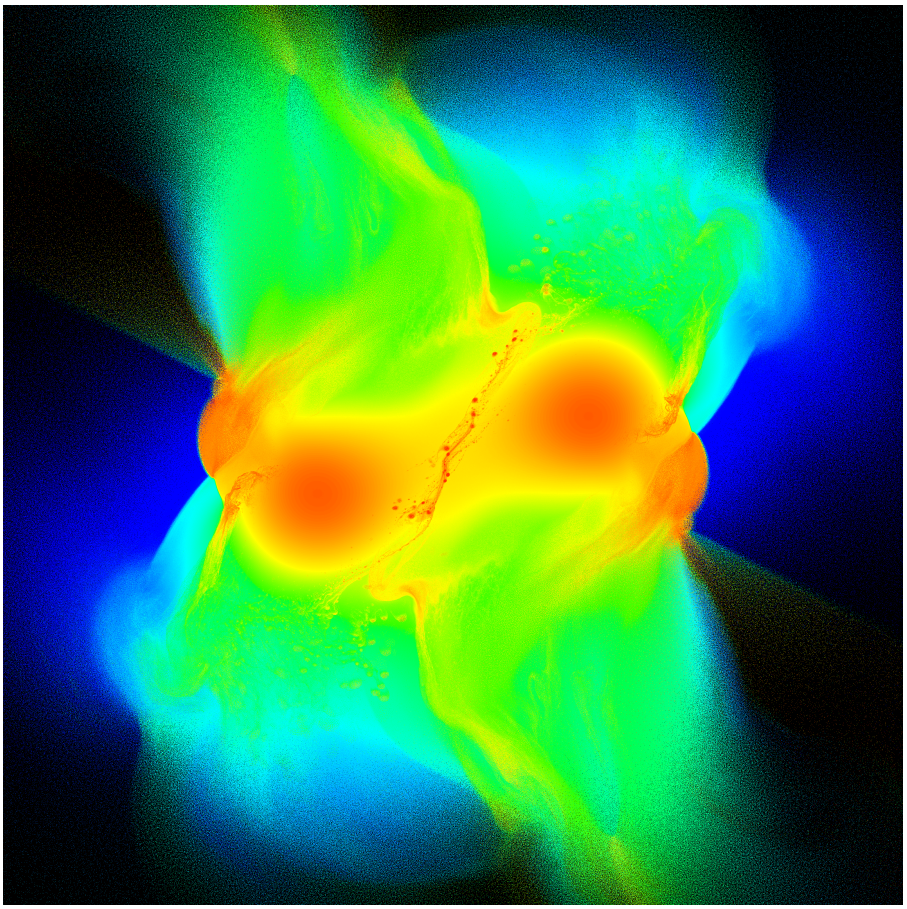

Figure 23: Temperature plot of step 600 of the 4.2×10^9 particle collision simulation. Logarithmic temperature scale: 1×10^2 K (blue) - 1.585×10^4 K (red).

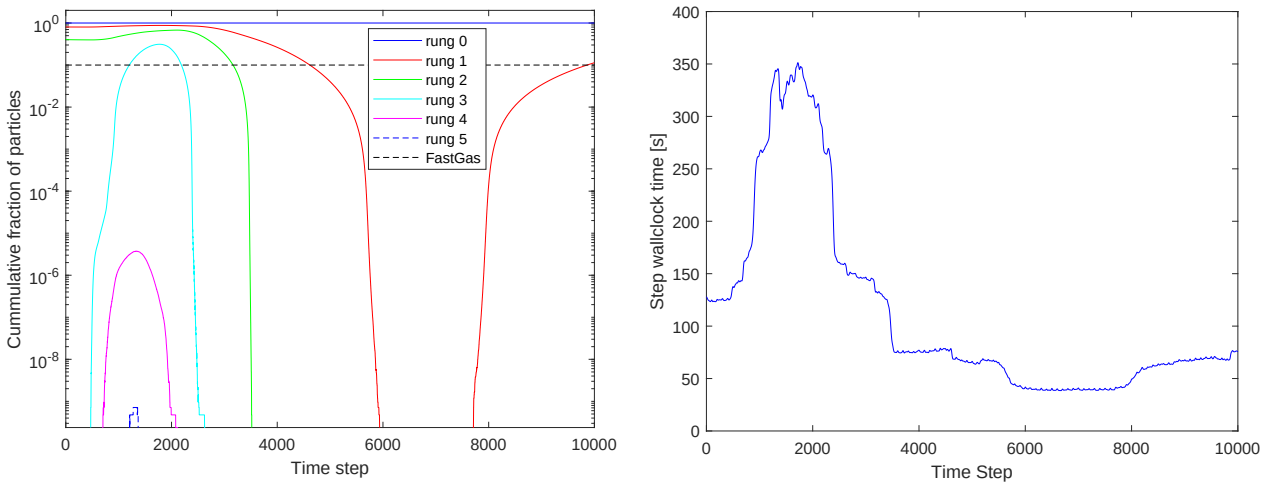

Figure 24: Cumulative particle rung evolution and step wallclock time for the 4.2×10^9 particle collision simulation run in the 32x8x16 configuration.

5 Conclusion and Outlook

Motivated by the lack of scalability of the currently used Gasoline code for planetary collision simulations, a fluid treatment in the form of SPH was implemented into the N-Body code pkdgrav3 to take advantage of its excellent gravity performance and scalability. For this, a novel approach to the neighbor finding was applied to reduce the work of this crucial step. We showed that this new concept works well and the code shows good strong and weak scaling, even though there is still some optimization potential. We also showed that higher resolution collision simulations are desirable, because higher particle numbers allow to resolve additional features (e.g. ocean, atmosphere or crust) or when more detailed results like mixing are of interest, as many fluid mixing features are not resolved at lower particle numbers. This implementation is only a proof of concept, but we obtained very promising results and are optimistic to push the state of the art in resolution of collision simulations to a new level.

Going forward, the main goal is to make the new implementation ready to replace Gasoline in the planetary collision simulations. This means to allow the use of more complex EOS (like Tillotson EOS, ANEOS, M-ANEOS and REOS) which can be accomplished by including the EOSlib developed in [41, 6]. To get correct results at the material/vacuum interface and at the interface between different materials like the core/mantle boundary, a surface correction and interface correction as described in [2] and [4] has to be added. While the surface correction should be straightforward to be included into the density walk, the interface correction needs more thought and likely an additional walk. To get an entropy conserving formulation, we can use the tools derived in [2] and already used in Gasoline and EOSlib, or for EOS that provide entropy information like (M-)ANEOS, derive something similar to the entropic formulation for the ideal gas described in Section 2.3.2.8. As the fluid model is only valid for larger bodies, adding a strength model would allow for the simulation of collisions between smaller bodies like asteroids or the analysis of impact cratering.

On the numerical side, the M4 kernel has many shortcomings and should be replaced by a member of the Wendland kernel family as described in [43] which is already implemented in Gasoline. To reduce dampening of subsonic turbulence and angular momentum transfer due to the artificial viscosity added to capture shocks, a viscosity limiter should be added. The Morris and Monaghan limiter described in Section 2.3.2.4 would fit perfectly into the existing framework, as the only change needed is the addition of two floats to the particle (the evolving viscosity α and its time derivative), as everything else is already provided. To get improved results in shearing and rotational flows, it can also be combined with the Balsara switch. Adding artificial thermal conductivity to better model instabilities should also be pretty straightforward, as even though the source term for the evolution of the thermal conductivity contains the Laplacian of the internal energy, Price [25] points out that using a rescaled first derivative of the kernel function instead of the second derivative leads to better results. Once all these improvements are in place, it is desirable to run validation tests, to show what extreme resolution simulations can achieve in tests like the Sod shock-tube, Sedov-Taylor blast wave, box test, Gresho-Chan vortex test or Rayleigh-Taylor and Kelvin-Helmholtz tests.

On the performance side, there is still optimization potential concerning cross-node communication as discussed in Section 4.5 where a wallclock time reduction of 11 % was achieved by changing a single parameter in the communication framework. A solution has to be found for the cases where a single cell with an enormous PP list dominates the wallclock time of the walk. The limit on the maximum smoothing length works well, but it has to be at least adaptive or better dynamic. A limit to the difference in time step of interacting particles has to be implemented according to [62] to improve integration stability. The machine that is currently called Eiger is actually only the multi-core partition of the upcoming general-purpose system Alps which will also contain GPU nodes. In order to harness the power of these GPU nodes, the density and SPH force calculation can be offloaded to the GPU with little modification as they use the PP infrastructure. The only tricky part is the fact that the `workParticle` has to hold onto a copy of the `ILP` instead of a reference, as it will be changed between Newton-Iterations because the GPU executions are asynchronous to the tree walk.

If we want to go beyond 4.2×10^9 particles, we need to modify the pre- and postprocessing pipeline such that it can handle these high particle numbers. One way would be to modify the tipsy file

format and the corresponding programs to use a larger integer type (`uint64_t`) or we could switch to a combination of `hdf5` for the file structure and PARAVIEW for postprocessing. This would have the advantage that PARAVIEW is fully parallelized and able to run on multiple nodes, while TIPSY is single threaded. But this would also mean to modify the programs of the pipeline to use the new format and add the ability to PKDGRAV3 to read `hdf5` files in addition to the already implemented writing capability.

References

[1] J. W. Wadsley, J. Stadel, and T. Quinn. "Gasoline: A Flexible, Parallel Implementation of TreeSPH". In: *New Astronomy* 9 (Feb. 1, 2004), pp. 137–158. ISSN: 1384-1076. DOI: 10.1016/j.newast.2003.08.004. URL: http://adsabs.harvard.edu/abs/2004NewA....9..137W (visited on 05/12/2020).

[2] Christian Reinhardt and Joachim Stadel. "Numerical Aspects of Giant Impact Simulations". In: *Monthly Notices of the Royal Astronomical Society* 467 (Jan. 28, 2017). DOI: 10.1093/mnras/stx322.

[3] Alice Chau et al. "Forming Mercury by Giant Impacts". In: *The Astrophysical Journal* 865.1 (Sept. 18, 2018), p. 35. ISSN: 1538-4357. DOI: 10.3847/1538-4357/aad8b0. arXiv: 1808.02448. URL: http://arxiv.org/abs/1808.02448 (visited on 10/30/2020).

[4] Christian Reinhardt et al. "Bifurcation in the History of Uranus and Neptune: The Role of Giant Impacts". In: *Monthly Notices of the Royal Astronomical Society* 492.4 (Mar. 11, 2020), pp. 5336–5353. ISSN: 0035-8711. DOI: 10.1093/mnras/stz3271. URL: https://academic.oup.com/mnras/article/492/4/5336/5637902 (visited on 08/10/2020).

[5] Miles L. Timpe et al. "Machine Learning Applied to Simulations of Collisions between Rotating, Differentiated Planets". In: *Computational Astrophysics and Cosmology* 7.1 (Dec. 2, 2020), p. 2. ISSN: 2197-7909. DOI: 10.1186/s40668-020-00034-6. URL: https://doi.org/10.1186/s40668-020-00034-6 (visited on 01/14/2021).

[6] Thomas Meier, Christian Reinhardt, and Joachim Gerhard Stadel. "The EOS/Resolution Conspiracy: Convergence in Proto-Planetary Collision Simulations". In: *Monthly Notices of the Royal Astronomical Society* 505.2 (June 4, 2021), pp. 1806–1816. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stab1441. URL: https://academic.oup.com/mnras/article/505/2/1806/6279686 (visited on 09/06/2021).

[7] C. Mastropietro et al. "The Gravitational and Hydrodynamical Interaction between the Large Magellanic Cloud and the Galaxy". In: *Monthly Notices of the Royal Astronomical Society* 363.2 (Oct. 21, 2005), pp. 509–520. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2005.09435.x. URL: https://doi.org/10.1111/j.1365-2966.2005.09435.x (visited on 12/06/2021).

[8] Charles H. Lineweaver and Marc Norman. *The Potato Radius: A Lower Minimum Size for Dwarf Planets*. Apr. 7, 2010. arXiv: 1004.1091 [astro-ph, physics:physics]. URL: http://arxiv.org/abs/1004.1091 (visited on 07/08/2020).

[9] Frank Shu. *Physics Of Astrophysics Volume 2 - Gas Dynamics*. First Edition. Mill Valley, Cal: University Science Books,U.S., Apr. 1, 1994. 476 pp. ISBN: 978-1-891389-67-2.

[10] A. Brandenburg et al. "The Pencil Code, a Modular MPI Code for Partial Differential Equations and Particles: Multipurpose and Multiuser-Maintained". In: *Journal of Open Source Software* 6.58 (Feb. 21, 2021), p. 2807. ISSN: 2475-9066. DOI: 10.21105/joss.02807. arXiv: 2009.08231. URL: http://arxiv.org/abs/2009.08231 (visited on 06/04/2021).

[11] Volker Springel et al. "Simulating Cosmic Structure Formation with the GADGET-4 Code". In: (2013), p. 81.

[12] A. Mignone et al. "PLUTO: A Numerical Code for Computational Astrophysics". In: *The Astrophysical Journal Supplement Series* 170.1 (May 2007), pp. 228–242. ISSN: 0067-0049, 1538-4365. DOI: 10.1086/513316. URL: https://iopscience.iop.org/article/10.1086/513316 (visited on 06/04/2021).

[13] James M. Stone et al. "Athena: A New Code for Astrophysical MHD". In: *The Astrophysical Journal Supplement Series* 178.1 (Sept. 2008), pp. 137–177. ISSN: 0067-0049, 1538-4365. DOI: 10.1086/588755. arXiv: 0804.0402. URL: http://arxiv.org/abs/0804.0402 (visited on 06/04/2021).

[14] Rainer Weinberger, Volker Springel, and Rüdiger Pakmor. "The Arepo Public Code Release". In: *The Astrophysical Journal Supplement Series* 248.2 (June 10, 2020), p. 32. ISSN: 1538-4365. DOI: 10.3847/1538-4365/ab908c. arXiv: 1909.04667. URL: http://arxiv.org/abs/1909.04667 (visited on 06/04/2021).

[15] Paul C. Duffell and Andrew I. MacFadyen. "TESS: A Relativistic Hydrodynamics Code on a Moving Voronoi Mesh". In: *The Astrophysical Journal Supplement Series* 197 (Dec. 1, 2011), p. 15. ISSN: 0067-0049. DOI: 10.1088/0067-0049/197/2/15. URL: http://adsabs.harvard.edu/abs/2011ApJS..197...15D (visited on 06/04/2021).

[16] R. Teyssier. "Cosmological Hydrodynamics with Adaptive Mesh Refinement: A New High Resolution Code Called RAMSES". In: *Astronomy & Astrophysics* 385.1 (Apr. 2002), pp. 337–364. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361:20011817. URL: http://www.aanda.org/10.1051/0004-6361:20011817 (visited on 06/04/2021).

[17] Greg L. Bryan et al. "ENZO: AN ADAPTIVE MESH REFINEMENT CODE FOR ASTROPHYSICS". In: *The Astrophysical Journal Supplement Series* 211.2 (Mar. 20, 2014), p. 19. ISSN: 0067-0049, 1538-4365. DOI: 10.1088/0067-0049/211/2/19. URL: https://iopscience.iop.org/article/10.1088/0067-0049/211/2/19 (visited on 06/04/2021).

[18] Philip F. Hopkins. "A New Class of Accurate, Mesh-Free Hydrodynamic Simulation Methods". In: *Monthly Notices of the Royal Astronomical Society* 450.1 (June 11, 2015), pp. 53–110. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/stv195. URL: http://academic.oup.com/mnras/article/450/1/53/992679/A-new-class-of-accurate-meshfree-hydrodynamic (visited on 03/09/2021).

[19] Oscar Agertz et al. "Fundamental Differences between SPH and Grid Methods". In: *Monthly Notices of the Royal Astronomical Society* 380.3 (Sept. 21, 2007), pp. 963–978. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2007.12183.x. URL: https://academic.oup.com/mnras/article/380/3/963/952655 (visited on 06/26/2020).

[20] Colin P. McNally, Jason L. Maron, and Mordecai-Mark Mac Low. "Phurbas: An Adaptive, Lagrangian, Meshless, Magnetohydrodynamics Code. II. Implementation and Tests". In: *The Astrophysical Journal Supplement Series* 200.1 (May 1, 2012), p. 7. ISSN: 0067-0049, 1538-4365. DOI: 10.1088/0067-0049/200/1/7. arXiv: 1110.0836. URL: http://arxiv.org/abs/1110.0836 (visited on 06/04/2021).

[21] Jason L. Maron, Colin P. McNally, and Mordecai-Mark Mac Low. "Phurbas: An Adaptive, Lagrangian, Meshless, Magnetohydrodynamics Code. I. Algorithm". In: *The Astrophysical Journal Supplement Series* 200.1 (May 1, 2012), p. 6. ISSN: 0067-0049, 1538-4365. DOI: 10.1088/0067-0049/200/1/6. arXiv: 1110.0835. URL: http://arxiv.org/abs/1110.0835 (visited on 06/04/2021).

[22] Debora Sijacki and Volker Springel. "Physical Viscosity in Smoothed Particle Hydrodynamics Simulations of Galaxy Clusters". In: *Monthly Notices of the Royal Astronomical Society* 371.3 (Sept. 21, 2006), pp. 1025–1046. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2006.10752.x. URL: https://doi.org/10.1111/j.1365-2966.2006.10752.x (visited on 12/06/2021).

[23] Matthew J. Colbrook et al. "Scaling Laws of Passive-Scalar Diffusion in the Interstellar Medium". In: *Monthly Notices of the Royal Astronomical Society* 467.2 (May 21, 2017), pp. 2421–2429. ISSN: 0035-8711. DOI: 10.1093/mnras/stx261. URL: https://doi.org/10.1093/mnras/stx261 (visited on 12/06/2021).

[24] Amirsaman Farrokhpanah, Markus Bussmann, and Javad Mostaghimi. "New Smoothed Particle Hydrodynamics (SPH) Formulation for Modeling Heat Conduction with Solidification and Melting". In: *Numerical Heat Transfer, Part B: Fundamentals* 71.4 (Apr. 3, 2017), pp. 299–312. ISSN: 1040-7790. DOI: 10.1080/10407790.2017.1293972. URL: https://doi.org/10.1080/10407790.2017.1293972 (visited on 12/06/2021).

[25] Daniel J. Price. "Smoothed Particle Hydrodynamics and Magnetohydrodynamics". In: *Journal of Computational Physics*. Special Issue: Computational Plasma Physics 231.3 (Feb. 1, 2012), pp. 759–794. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2010.12.011. URL: https://www.sciencedirect.com/science/article/pii/S0021999110006753 (visited on 03/09/2021).

[26] J. J. Monaghan. *A Turbulence Model for Smoothed Particle Hydrodynamics*. Nov. 12, 2009. arXiv: 0911.2523 [physics]. URL: http://arxiv.org/abs/0911.2523 (visited on 12/06/2021).

[27] Jillian Bellovary et al. "THE FIRST MASSIVE BLACK HOLE SEEDS AND THEIR HOSTS". In: *The Astrophysical Journal* 742.1 (Nov. 2011), p. 13. ISSN: 0004-637X. DOI: 10.1088/0004-637X/742/1/13. URL: https://doi.org/10.1088/0004-637x/742/1/13 (visited on 12/06/2021).

[28] C. Scannapieco et al. "Feedback and Metal Enrichment in Cosmological SPH Simulations – II. A Multiphase Model with Supernova Energy Feedback". In: *Monthly Notices of the Royal Astronomical Society* 371.3 (Sept. 21, 2006), pp. 1125–1139. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2006.10785.x. URL: https://doi.org/10.1111/j.1365-2966.2006.10785.x (visited on 12/06/2021).

[29] Kastytis Zubovas, Martin A. Bourne, and Sergei Nayakshin. "A Simple Way to Improve AGN Feedback Prescription in SPH Simulations". In: *Monthly Notices of the Royal Astronomical Society* 457.1 (Mar. 21, 2016), pp. 496–509. ISSN: 0035-8711. DOI: 10.1093/mnras/stv2971. URL: https://doi.org/10.1093/mnras/stv2971 (visited on 12/06/2021).

[30] Josh Borrow et al. "Sphenix: Smoothed Particle Hydrodynamics for the next Generation of Galaxy Formation Simulations". In: *Monthly Notices of the Royal Astronomical Society* (Nov. 8, 2021), stab3166. ISSN: 0035-8711. DOI: 10.1093/mnras/stab3166. URL: https://doi.org/10.1093/mnras/stab3166 (visited on 12/06/2021).

[31] Junichiro Makino et al. "GRAPE-6: Massively-Parallel Special-Purpose Computer for Astrophysical Particle Simulations". In: *Publications of the Astronomical Society of Japan* 55.6 (Dec. 25, 2003), pp. 1163–1187. ISSN: 0004-6264. DOI: 10.1093/pasj/55.6.1163. URL: https://doi.org/10.1093/pasj/55.6.1163 (visited on 06/10/2021).

[32] Walter Dehnen. "Towards Optimal Softening in Three-Dimensional N-body Codes — I. Minimizing the Force Error". In: *Monthly Notices of the Royal Astronomical Society* 324.2 (June 11, 2001), pp. 273–291. ISSN: 0035-8711. DOI: 10.1046/j.1365-8711.2001.04237.x. URL: https://doi.org/10.1046/j.1365-8711.2001.04237.x (visited on 12/06/2021).

[33] Josh Barnes and Piet Hut. "A Hierarchical O(N Log N) Force-Calculation Algorithm". In: *Nature* 324.6096 (6096 Dec. 1986), pp. 446–449. ISSN: 1476-4687. DOI: 10.1038/324446a0. URL: https://www.nature.com/articles/324446a0 (visited on 06/26/2020).

[34] L Greengard and V Rokhlin. "A Fast Algorithm for Particle Simulations". In: *Journal of Computational Physics* 73.2 (Dec. 1, 1987), pp. 325–348. ISSN: 0021-9991. DOI: 10.1016/0021-9991(87)90140-9. URL: https://www.sciencedirect.com/science/article/pii/0021999187901409 (visited on 06/10/2021).

[35] Walter Dehnen. "A Very Fast and Momentum-conserving Tree Code". In: *The Astrophysical Journal Letters* 536 (June 1, 2000), pp. L39–L42. ISSN: 0004-637X. DOI: 10.1086/312724. URL: http://adsabs.harvard.edu/abs/2000ApJ...536L..39D (visited on 06/10/2021).

[36] Douglas Potter, Joachim Stadel, and Romain Teyssier. "PKDGRAV3: Beyond Trillion Particle Cosmological Simulations for the next Era of Galaxy Surveys". In: *Computational Astrophysics and Cosmology* 4.1 (Dec. 2017), p. 2. ISSN: 2197-7909. DOI: 10.1186/s40668-017-0021-1. URL: https://comp-astrophys-cosmol.springeropen.com/articles/10.1186/s40668-017-0021-1 (visited on 03/15/2021).

[37] J. W. Eastwood, R. W. Hockney, and D. N. Lawrence. "P3M3DP-the Three-Dimensional Periodic Particle-Particle/Particle-Mesh Program". In: *Computer Physics Communications* 35 (Jan. 1, 1984), pp. C–618. ISSN: 0010-4655. DOI: 10.1016/S0010-4655(84)82783-6. URL: https://www.sciencedirect.com/science/article/pii/S0010465584827836 (visited on 06/04/2021).

[38] Volker Springel. "The Cosmological Simulation Code GADGET-2". In: *Monthly Notices of the Royal Astronomical Society* 364 (Dec. 1, 2005), pp. 1105–1134. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2005.09655.x. URL: http://adsabs.harvard.edu/abs/2005MNRAS.364.1105S (visited on 05/28/2021).

[39] R. A. Gingold and J. J. Monaghan. "Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars". In: *Monthly Notices of the Royal Astronomical Society* 181.3 (Dec. 1, 1977), pp. 375–389. ISSN: 0035-8711. DOI: 10.1093/mnras/181.3.375. URL: https://academic.oup.com/mnras/article/181/3/375/988212 (visited on 05/05/2020).

[40] L. B. Lucy. "A Numerical Approach to the Testing of the Fission Hypothesis". In: *The Astronomical Journal* 82 (Dec. 1, 1977), pp. 1013–1024. ISSN: 0004-6256. DOI: 10.1086/112164. URL: http://adsabs.harvard.edu/abs/1977AJ.....82.1013L (visited on 06/04/2021).

[41] Thomas Meier. "Equations of State for Collision Simulations". BA thesis. Zürich: University of Zürich, Aug. 21, 2020. 65 pp.

[42] J. J. Monaghan. "Smoothed Particle Hydrodynamics". In: *Annual Review of Astronomy and Astrophysics* 30 (1992), pp. 543–574. ISSN: 0066-4146. DOI: 10.1146/annurev.aa.30.090192.002551. URL: http://adsabs.harvard.edu/abs/1992ARA%26A..30..543M (visited on 07/08/2020).

[43] Walter Dehnen and Hossam Aly. "Improving Convergence in Smoothed Particle Hydrodynamics Simulations without Pairing Instability". In: *Monthly Notices of the Royal Astronomical Society* 425.2 (Sept. 11, 2012), pp. 1068–1082. ISSN: 00358711. DOI: 10.1111/j.1365-2966.2012.21439.x. arXiv: 1204.2471. URL: http://arxiv.org/abs/1204.2471 (visited on 07/08/2020).

[44] Lars Hernquist and Neal Katz. "TREESPH - A Unification of SPH with the Hierarchical Tree Method". In: *The Astrophysical Journal Supplement Series* 70 (June 1, 1989), pp. 419–446. ISSN: 0067-0049. DOI: 10.1086/191344. URL: http://adsabs.harvard.edu/abs/1989ApJS...70..419H (visited on 06/26/2020).

[45] Volker Springel. "Smoothed Particle Hydrodynamics in Astrophysics". In: *Annual Review of Astronomy and Astrophysics* 48.1 (Aug. 1, 2010), pp. 391–430. ISSN: 0066-4146. DOI: 10.1146/annurev-astro-081309-130914. URL: https://www.annualreviews.org/doi/10.1146/annurev-astro-081309-130914 (visited on 03/09/2021).

[46] Hongping Deng et al. "Enhanced Mixing in Giant Impact Simulations with a New Lagrangian Method". In: *The Astrophysical Journal* 870 (Nov. 13, 2019). DOI: 10.3847/1538-4357/aaf399.

[47] Dinshaw S. Balsara. "Von Neumann Stability Analysis of Smoothed Particle Hydrodynamics—Suggestions for Optimal Algorithms". In: *Journal of Computational Physics* 121.2 (Oct. 15, 1995), pp. 357–372. ISSN: 0021-9991. DOI: 10.1016/S0021-9991(95)90221-X. URL: http://www.sciencedirect.com/science/article/pii/S002199919590221X (visited on 06/26/2020).

[48] J. P. Morris and J. J. Monaghan. "A Switch to Reduce SPH Viscosity". In: *Journal of Computational Physics* 136.1 (Sept. 1, 1997), pp. 41–50. ISSN: 0021-9991. DOI: 10.1006/jcph.1997.5690. URL: https://www.sciencedirect.com/science/article/pii/S0021999197956904 (visited on 11/10/2021).

[49] Lee Cullen and Walter Dehnen. "Inviscid Smoothed Particle Hydrodynamics: Inviscid Smoothed Particle Hydrodynamics". In: *Monthly Notices of the Royal Astronomical Society* 408.2 (Oct. 21, 2010), pp. 669–683. ISSN: 00358711. DOI: 10.1111/j.1365-2966.2010.17158.x. URL: https://academic.oup.com/mnras/article-lookup/doi/10.1111/j.1365-2966.2010.17158.x (visited on 03/09/2021).

[50] Philip F. Hopkins. "A General Class of Lagrangian Smoothed Particle Hydrodynamics Methods and Implications for Fluid Mixing Problems". In: *Monthly Notices of the Royal Astronomical Society* 428.4 (Feb. 1, 2013), pp. 2840–2856. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/sts210. URL: http://academic.oup.com/mnras/article/428/4/2840/992468/A-general-class-of-Lagrangian-smoothed-particle (visited on 03/09/2021).

[51] J. I. Read, T. Hayfield, and O. Agertz. "Resolving Mixing in Smoothed Particle Hydrodynamics". In: *Monthly Notices of the Royal Astronomical Society* 405.3 (July 1, 2010), pp. 1513–1530. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2010.16577.x. URL: https://doi.org/10.1111/j.1365-2966.2010.16577.x (visited on 12/06/2021).

[52] J. I. Read and T. Hayfield. "SPHS: Smoothed Particle Hydrodynamics with a Higher Order Dissipation Switch". In: *Monthly Notices of the Royal Astronomical Society* 422.4 (June 1, 2012), pp. 3037–3055. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2012.20819.x. URL: https://doi.org/10.1111/j.1365-2966.2012.20819.x (visited on 12/06/2021).

[53] R. M. Cabezón, D. García-Senz, and J. Figueira. "SPHYNX: An Accurate Density-Based SPH Method for Astrophysical Applications". In: *Astronomy & Astrophysics* 606 (Oct. 2017), A78. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201630208. URL: http://www.aanda.org/10.1051/0004-6361/201630208 (visited on 06/02/2021).

[54] Daniel J. Price. "Modelling Discontinuities and Kelvin–Helmholtz Instabilities in SPH". In: *Journal of Computational Physics* 227.24 (Dec. 20, 2008), pp. 10040–10057. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2008.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0021999108004270 (visited on 03/09/2021).

[55] J. W. Wadsley, B. W. Keller, and T. R. Quinn. *Gasoline2: A Modern Smoothed Particle Hydrodynamics Code | Monthly Notices of the Royal Astronomical Society | Oxford Academic*. URL: https://academic.oup.com/mnras/article/471/2/2357/3906602 (visited on 11/09/2021).

[56] Volker Springel and Lars Hernquist. "Cosmological Smoothed Particle Hydrodynamics Simulations: The Entropy Equation". In: *Monthly Notices of the Royal Astronomical Society* 333.3 (July 1, 2002), pp. 649–664. ISSN: 0035-8711. DOI: 10.1046/j.1365-8711.2002.05445.x. URL: https://academic.oup.com/mnras/article/333/3/649/1002394 (visited on 10/29/2020).

[57] Walter Dehnen and J. I. Read. "N-Body Simulations of Gravitational Dynamics". In: *The European Physical Journal Plus* 126.5 (May 30, 2011), p. 55. ISSN: 2190-5444. DOI: 10.1140/epjp/i2011-11055-3. URL: https://doi.org/10.1140/epjp/i2011-11055-3 (visited on 11/09/2021).

[58] Joachim Gerhard Stadel. "Cosmological N-body Simulations and Their Analysis". University of Washington, 2001, p. 3657. URL: http://adsabs.harvard.edu/abs/2001PhDT.....21S (visited on 05/12/2020).

[59] James Douglas Potter. "Towards Exascale Computing for Cosmological Simulations". University of Zürich, 2017. 173 pp.

[60] Lars Hernquist, Francois R. Bouchet, and Yasushi Suto. "Application of the Ewald Method to Cosmological N-Body Simulations". In: *The Astrophysical Journal Supplement Series* 75 (Feb. 1, 1991), p. 231. ISSN: 0067-0049. DOI: 10.1086/191530. URL: https://ui.adsabs.harvard.edu/abs/1991ApJS...75..231H (visited on 12/08/2021).

[61] Alice Chau et al. "Could Uranus and Neptune Form by Collisions of Planetary Embryos?" In: *Monthly Notices of the Royal Astronomical Society* 502.2 (Apr. 1, 2021), pp. 1647–1660. ISSN: 0035-8711. DOI: 10.1093/mnras/staa4021. URL: https://doi.org/10.1093/mnras/staa4021 (visited on 03/19/2021).

[62] Takayuki R. Saitoh and Junichiro Makino. "A NECESSARY CONDITION FOR INDIVIDUAL TIME STEPS IN SPH SIMULATIONS". In: *The Astrophysical Journal* 697.2 (May 2009), pp. L99–L102. ISSN: 0004-637X. DOI: 10.1088/0004-637X/697/2/L99. URL: https://doi.org/10.1088/0004-637x/697/2/l99 (visited on 11/19/2021).