Jül-4340

Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Highly-parallel Smoothed Particle Hydrodynamics modelling of protoplanetary discs

A. Breslau

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
FORSCHUNGSZENTRUM

Highly-parallel Smoothed Particle Hydrodynamics modelling of protoplanetary discs

A. Breslau

Berichte des Forschungszentrums Jülich; 4340
ISSN 0944-2952
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)
Jül-4340

Vollständig frei verfügbar im Internet auf dem Jülicher Open Access Server (JUWEL)
unter http://www.fz-juelich.de/zb/juwel

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek, Verlag
D-52425 Jülich · Bundesrepublik Deutschland
☎ 02461 61-5220 · Telefax: 02461 61-6103 · e-mail: zb-publikation@fz-juelich.de

Zusammenfassung

Sterne entstehen durch gravitativen Kollaps aus molekularen Wolken. Während dieses Prozesses bilden sich aufgrund der Erhaltung des Drehimpulses aus Gas und Staub bestehende Scheiben um die jungen Sterne, sogenannte protoplanetare Scheiben. Aus diesen Scheiben können im Laufe ihrer Entwicklung Planeten entstehen.

Durch Fluktuationen oder externe Störung können sich dichtere Klumpen innerhalb der Scheiben bilden. Wenn das Material der Klumpen ausreichend kalt ist, überwiegt die Gravitationskraft den thermischen Kräften und das Material wird gebunden. Durch Energieabstrahlung können die Klumpen ihre Größe verringern, an Dichte gewinnen und letztendlich möglicherweise Planeten bilden. Scheiben, deren Massen 10% der Sternmasse überschreiten, verhalten sich näherungsweise wie selbstgravitierende, viskose Flüssigkeiten. Durch die Dissipation von kinetischer Energie in Wärme, bedingt durch die Viskosität, wird die Scheibe geheizt, was die Fragmentierung möglicherweise verhindern kann, wohingegen das Kühlen der Scheiben durch Abstrahlung von Energie die Fragmentierung begünstigt. Das genaue Zusammenspiel dieser und anderer Prozesse ist bislang nicht bekannt.

Ziel dieser Arbeit ist die Entwicklung eines Codes zur Simulation viskoser, selbstgravitierender Scheiben auf hoch parallelen Superrechnern, um damit zum Verständnis dieser Prozesse beizutragen. Dazu wird der bisher für Plasma-Simulationen verwendete, hoch skalierende Code PEPC (Pretty Efficient Coulomb Solver) zunächst für die Berechnung von Gravitationskräften modifiziert. Anschließend wird der Code um eine “Smoothed Particle Hydrodynamics” (SPH) genannte Methode zur Lösung von Strömungsgleichungen erweitert.

Um nachzuweisen, dass der neue Code physikalisch korrekte Ergebnisse liefert, wird er auf Testprobleme angewendet. Es wird gezeigt, dass der neue Code akustische Wellen über 10^4 Zeitschritte mit nur kleinen Abweichungen von der analytischen Lösung behandeln kann und Schockwellen ausreichend auflöst. Darüber hinaus wird das korrekte Zusammenspiel der neuen mit der alten Codekomponente demonstriert.

Die Simulation einer protoplanetaren Scheibe mit dem neuen Simulationscode, wird schließlich erfolgreich getestet. Dabei zeigt sich, dass zur stabilen Modellierung einer solchen Scheibe die Integration weiterer physikalischer Prozesse, wie zum Beispiel Kühlung, in den Code unerlässlich ist.

Abstract

Stars form by gravitational collapse of molecular clouds. Due to angular momentum conservation discs consisting of gas and dust form during this process around the young stars. Due to the potential of later formation of planets out of the disc-material they are called protoplanetary discs.

Fluctuations in the discs can lead to the formation of dense clumps. When the material inside the clumps is cold enough, the gravitational force outbalances the thermal forces and the material is bound. By radiative energy loss the clumps can shrink and may ultimately form planets. Especially discs with masses $M_{\text{disc}} > 0.1 M_{\text{star}}$ behave in first approximation like self-gravitating, viscous fluids. Viscous heating of the disc can suppress fragmentation processes, while radiative cooling can support them. The exact interplay of these and other processes is still unknown.

The aim of this work is the development of a code for the simulation of viscous, self-gravitating discs with highly parallel supercomputers, to contribute to the understanding of these processes. Therefore the highly scalable plasma-code PEPC (Pretty Efficient Coulomb Solver) is modified for the computation of gravitational forces. Afterwards the code is extended with the Smoothed Particle Hydrodynamics (SPH) method for fluid computation.

To demonstrate, that the code produces correct physical results, several test problems are investigated. It is shown, that the code can propagate sound waves over 10^4 time steps with only small deviations from the analytical solution and that the code can resolve shock waves sufficiently. Further, the correct cooperation of the new and old code components is demonstrated.

The simulation of a protoplanetary disc with the new simulation code is tested successfully. Nevertheless it shows, that for the stable modelling of such a disc, the inclusion of additional physical processes, like cooling, is necessary.

Contents

1 Motivation

Discs around young stars consisting of gas and dust are the starting material for the formation of planets. In the standard theory of planet formation the μm-sized dust particles grow to planetesimals ($\varnothing \approx 10$ km) by coagulation processes. Afterwards, the growth processes is dominated by gravitational interactions. If the disc provides a certain amount of material, the planetesimals are able to grow further until they reach planet size. If the formed planetary embryos are massive enough, they can accrete gas from the disc to form gas giant planets with masses up to several M_{Jupiter}.

Alternatively, it was proposed that planets might form through gravitational instabilities (GI) in the gas disc. Given the right conditions, the disc can fragment and form dense clumps. Whether these clumps can stay stable enough to form planets is an open question. This problem is currently investigated intensely with computer simulations.

For the simulation of the fragmentation of discs around young stars, codes are needed, solving primarily the gravitational and hydrodynamical interactions among the gas and dust particles. Therefore gravity solvers are often combined with the Smoothed Particle Hydrodynamics (SPH) method as fluid solver. This is done, due to advantages of this combination over the combination of the gravity solver with a grid-based fluid solver. These advantages are the intrinsic resolution adaptiveness of SPH and that SPH as well as most gravity solvers are particle based methods.

For simulations of fragmenting protoplanetary discs particle numbers of some 10^6 are needed. This requirement results from two conditions, which have to be fulfilled. On the one hand the disc height ($H \ll r$) has to be resolved sufficiently to avoid numerical heating suppressing fragmentation. On the other hand the Jeans mass has to be resolved sufficiently for correct modelling of gravitational fragmentation. The high particle numbers result in a high computational effort, why highly efficient codes are needed.

The aim of this work is to develop an efficient simulation code for parallel computers, which combines SPH with a gravity solver. With this code highly resolved protoplanetary discs will be investigated to contribute to the understanding of the fragmentation process.

2 Introduction

2.1 The formation of stars and protoplanetary discs

Only a fraction of the matter in a galaxy like our own Milky Way is bound in stars. Large parts are confined in cold molecular clouds. These clouds consist mainly of molecular hydrogen (H_2) and can be up to several 10 parsecs[1] (pc) in diameter. The clouds are basically stabilised by a balance between attracting gravitational force and repulsive pressure. The cloud is stable as long as its mass M_C is below the Jeans mass M_J

$$M_J = 5.46 \left(\frac{k_B T}{m_M G} \right)^{3/2} \rho^{-1/2}, \tag{2.1}$$

where ρ is the density of the cloud, T its temperature, k_B the Boltzmann constant, G the gravitational constant and m_M the mass of an average gas molecule (e.g. Unsöld & Baschek, 2005, p. 380). Fluctuations in the properties of the gas, like density or temperature, can lead to the local dominance of the gravitational force resulting in the collapse of this part of the cloud (see Fig. 2.1 a). During such a collapse part of the gas contracts from an extension of several 10^5 Astronomical Units[2] (AU) to a few AU meanwhile the density increases by several orders of magnitude. Due to the prior inhomogeneity and motion of the gas, it has an angular momentum which is conserved during the collapse. This leads to the formation of a rotating disc-like structure with a denser core in the centre (see Fig. 2.1 b).

During these early times in the formation of a star the core has a mass of less than $10^{-2}\ M_{Sun}$ while the disc can have a mass of several M_{Sun}. While the core is approximately in hydrostatic equilibrium, matter from the disc continues falling onto the core. This can only happen, when the angular momentum of the infalling matter is somehow transported outwards. Simultaneously, magnetic fields are responsible for mass ejections of the protostar perpendicular to the midplane of the disc. The kinetic and potential energy of this accreted matter is transformed into thermal energy and radiated away. At that evolutionary stage this is the main source of the luminosity of the core, now called protostar.

During this accretion phase temperature and density in the protostar increase and initiate hydrogen burning. At this point, there can still be a remaining disc, now having only a fraction of the central stars mass ($M_{Disc} \lesssim 0.1 M_{Star}$) (see Fig. 2.1 c). Eventually accretion stops and the star reaches hydrostatic equilibrium considering the pressure caused by the nuclear fusion. Then the star has reached

[1] 1 parsec is the distance from which the distance from the earth to the sun is seen under an angle of $1''$, $\approx 3 \cdot 10^{16}$ m or 3.26 light-years.

[2] 1 Astronomical Unit is the distance between earth and sun, $\approx 1.5 \cdot 10^8$ km.

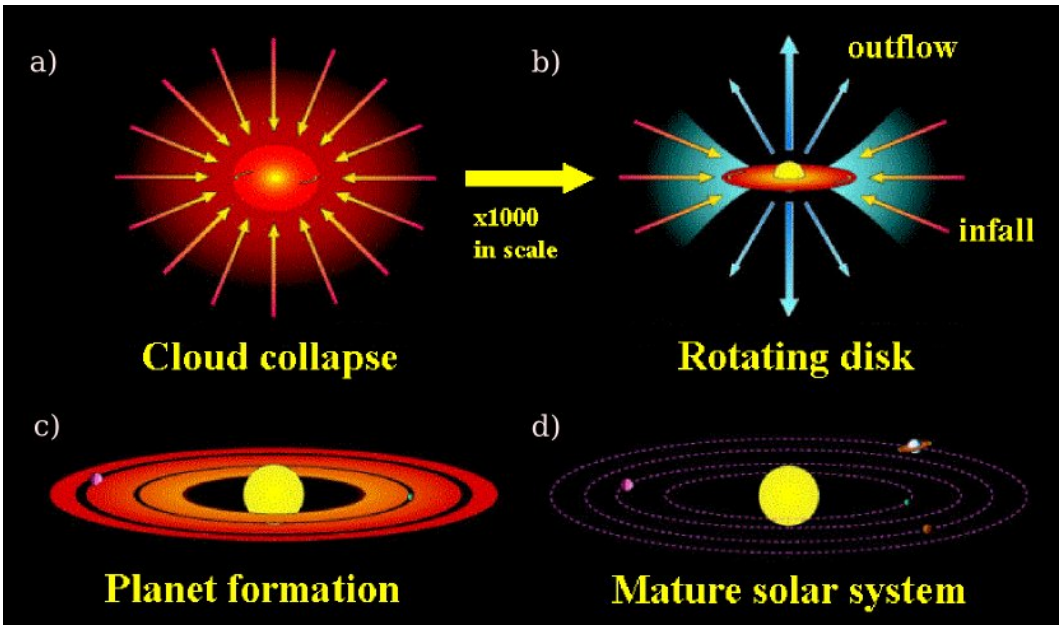

Figure 2.1: Schematic view of the formation of a single star. (McCaughrean, M., n.d.)

the main sequence. The matter forming the disc develops under several influences which can result either in the destruction of the disc or the formation of planets (see Fig. 2.1 d).

This is the current view of the formation of a single star. However, most stars are not only binaries or higher multiples, but often part of a cluster of stars containing several 100 to 100000 of stars. In this case the stellar environment can influence the star formation process significantly.

2.2 Observations and properties of protoplanetary discs

Observations have confirmed above theoretical picture that most, if not all, young stars are surrounded by a protoplanetary disc (Unsöld & Baschek, 2005). The huge contrast in luminosity between the star and disc combined with the small angular scale make direct observations of the disc very difficult.

Figure 2.2 shows the Spectral Energy Distribution (SED) of the star disc system WL 18 in Ophiuchus (Andrews et al., 2010). The SED is the distribution of the energy flux of radiation depending on the wavelength or frequency. The black dots are the observational data and the red line is the best fit of star and disc properties to the data. The dashed blue line shows the stars long-wavelength emission according to the Rayleigh-Jeans approximation of the Plank-law

$$B_\lambda(T) = \frac{2ck_BT}{\lambda^4}, \tag{2.2}$$

where c is the speed of light and λ the wavelength (Unsöld & Baschek, 2005). This is a typical sample SED of a star surrounded by a protoplanetary disc. It can be easily seen that in the optical wavelengths around 0.5 μm the observational data fit the black body radiation of the star (shown by the left part of the red line and the dashed blue line). There is no noticeable contribution by the discs radiation.

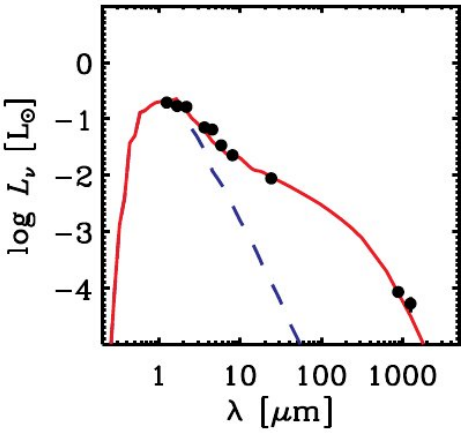

Figure 2.2: Spectral Energy Distribution of the young star WL 18 in the Ophiuchus star forming region. The black dots are the observational data and the red line is the best fit of star and disc properties to the data. The dashed blue line shows the stars long-wavelength emissions. (Image by Andrews et al. (2010))

Therefore it is almost impossible to observe protoplanetary discs directly in optical wavelengths.

On the other hand the plot shows another property of the SED of a star disc system exploited by the method primarily used to observe circumstellar discs. According to Wien's displacement law

$$\lambda T = 2.90 \cdot 10^{-3} \text{m K}, \tag{2.3}$$

the maximum-flux wavelength of a star with a surface temperature of several 1 000 K and a disc with a surface temperature of some 10 to few 100 K differ by a factor of 10 to 100 (Unsöld & Baschek, 2005).The Infrared[3] (IR) to sub-mm wavelength regime of the stars emitted radiation is given by (2.2). The thermal radiation of a protoplanetary disc in this wavelength regime is often much more luminous as shown by the red line on top of the stars dashed blue line. The area between these lines is the IR excess caused by the radiation of the protoplanetary disc. The gap between the data points in the range $[1:30]$ μm and around 1 000 μm is due to missing instruments for this wavelength regime. According to (2.3) the wavelengths of the data points at the left and right side of the IR excess correspond to temperatures of a few Kelvin at the outer regions of the disc and some 100 K at the inner regions.

Because of our large distance to star forming regions[4] ($>$ 100 pc), even large discs with approximately 1 000 AU in diameter are seen very close to their host star (here 5″ for the outer regions of this large discs). This small angular separations makes it impossible to image discs around stars more than a few 1000 pc distant, even with modern high resolution telescopes. For closer star disc systems the direct imaging of the disc is challenging due to the high contrast between star and disc. Some closer discs have been imaged by using a coronagraph. This is an attachment to the telescope blocking the light from the star. Figure 2.3 shows a sample of a direct image of a disc. The image shows the edge

[3] Infrared light is the light with wavelength between visible light ($\approx$ 750nm) and radio waves ($\approx$ 1mm).

[4] Ophiuchus-Scorpius and Taurus Auriga are the closest star forming regions to the earth with a distance of $\approx$ 140 pc Ghez et al. (1993).

on view of a star surrounded by a protoplanetary disc in the orion nebula, ≈ 400 pc away.

Figure 2.3: Edge on view of a star surrounded by a protoplanetary disc in the Orion Nebula. The star can not be seen, because it is shadowed by the disc. (Cut from image by Bally & Throop (n.d.))

The observational data are usually fitted to theoretical models to determine specific properties of the discs. In general, the results cover big ranges of values. This can have various reasons. First, different observations are in general not in the same wavelength, which makes it difficult to compare these observations, because in different wavelengths different matter is observed (see Sec. 2.3). When the observed discs differ in age, it is crucial to determine the age correctly, before the observations can be compared. Furthermore, the environment of the discs can influence their properties, as described in the following. And finally, the results depend on the used theoretical model.

Fitting a surface-density profile of the form

$$\Sigma(r) \propto r^{-p} \tag{2.4}$$

produced values for p in general in the range $[0:1]$(Williams & Cieza, 2011), but also values $p < 0$ and $p > 1$ have been found (Isella et al., 2009; Andrews & Williams, 2007). Masses were found mainly from 10^{-3} to $10^{-1} M_{\text{sun}}$ (Williams & Cieza, 2011), but here as well other values have been found. In general younger discs seam to be more massive.

Usually the surface-density profile declines rapidly at some outer cutoff radius. For example Vicente & Alves (2005) have measured the diameters of protoplanetary disc silhouettes against the bright orion nebula. They found radii from $r_{\text{disc}} = 50$ to $r_{\text{disc}} = 200$ AU. Rarely also discs with $r_{\text{disc}} > 1000$ AU were observed (Williams & Cieza, 2011).

The typical disc thickness has been found to be $\approx 10\%$ of the discs diameter (Vicente & Alves, 2005).

Surface-temperature T_{s} profiles of the form

$$T_{\text{s}} \propto r^{-q} \tag{2.5}$$

were fitted to the data. Beckwith et al. (1990) found $q = 0.5$ and Lynden-Bell & Pringle (1974) found $q = 3/4$ far from the star. Temperatures have been found from 10 K at ≈ 100 AU up to 2 000 K at the discs midplane in the inner regions (see Boss, 1998b, for an overview).

2.3 The evolution of protoplanetary discs

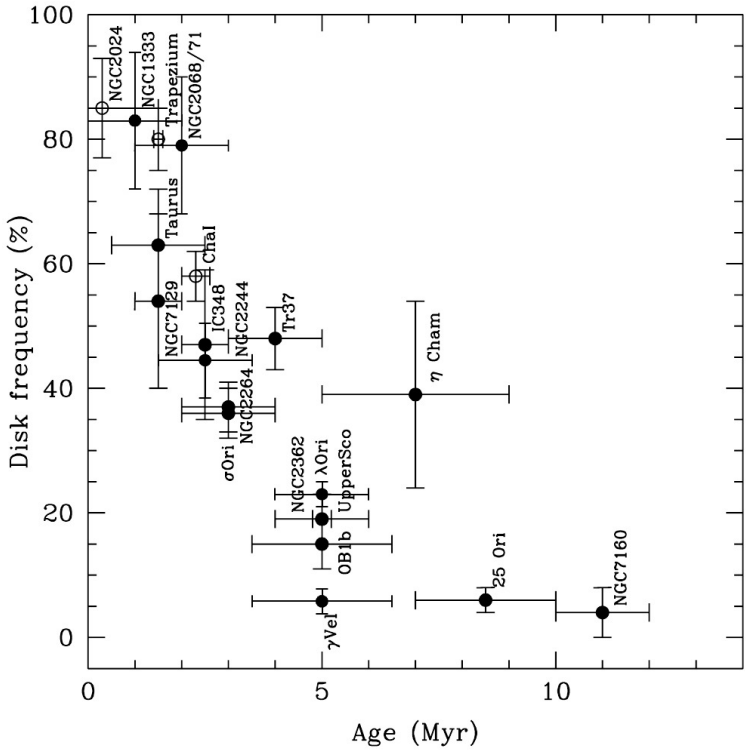

Figure 2.4: Fraction of stars with near-infrared disc emission as a function of the age of the stellar group they belong to. Open circles and solid symbols represent the disc frequency for stars in the T Tauri (TTS) mass range (K̃5 or later), derived from different wavelength data. For further description see original paper. (Image and text by Hernández et al., 2008)

Observations of young stars in star clusters of different age showed that the proportion of stars with IR excess, indicating a disc, decreases with time (Haisch et al., 2001; Hernández et al., 2008) (see Fig. 2.4). In clusters of an age of 5 million years (Myr) and above most systems seem to be disc-less. There are in principle two possible reasons for this observations:

The first reason is, that the discs materials properties change in a way, that it stops radiating in observable wavelengths. The SED of the radiation emitted by the discs depends basically on the temperature and size of the dust grains. Thus the observation depend on the chosen wavelength and these properties of the dust. When a specific wavelength was chosen for an observation and the dust does not emit light of this wavelength because it is to cold, or the grains are to big, the disc can not be observed. As mentioned before, observations in wavelengths around 0.1μm are not possible, because no instruments are available for this wavelength regime. Because km-sized planetesimals emit mainly in this wavelengths, it is not possible to observe them directly. That means, when the dust grains become larger due to coagulation processes, the IR radiation declines even though the disc as a whole remains (see Sec. 2.3.1). This can lead to the effect, that existing disc material is no longer observable.

The second reason is, that the protoplanetary discs are destroyed by the removal of their material. Several processes are known to play a role in this disc destruction.

Viscous disc spreading

The material from the disc can be accreted by the star with a smaller rate than in the protostar phase. Because the disc is rotating, this can only happen, when the inward migrating matter looses angular momentum. Due to general angular momentum conservation it can only be transfered to other material. A good candidate for causing the outward transport of angular momentum is viscosity. The faster rotating inner material is decelerated by friction, which accelerates the slower outer material. The decelerated inner material moves finally into the star, the accelerated outer material moves to a higher orbit. By this means the disc spreads outward (Pringle, 1981).

Photoevaporation

As the protostar increases its mass, its luminosity increases as well. When the star is luminous enough the inner regions of the disc are heated so much, that the gas molecules can reach escape velocity and leave the potential well of the system. This process is called photoevaporation. For very luminous stars this can lead to the destruction of the disc from the inside outwards.

As well, the radiation from another nearby luminous star can cause the destruction of the disc by photoevaporation from the outside inwards.

Tidal stripping

Another process is the removal of disc material by encounters with other stars (Pfalzner et al., 2006). When the star-disc system is located in a star cluster, encounters with other stars are likely to occur. When an encounter is close enough and the perturber mass is high enough part of the disc material can become unbound and leave the system. Strong or repeated encounters can even lead to a complete destruction of the disc.

2.3.1 Two planet formation scenarios

Besides the above mentioned processes, the formation of planetesimals or planets lead to a removal of observable disc material. While the gas and dust can be observed by its IR radiation the planetesimals are too large, to be observable in the IR and too small, to be observable in the optical. Even the bigger and much more massive planets are often difficult to observe.

Two competing planet formation scenarios were proposed.

The core accretion model

The material of a protoplanetary disc consists of $\approx 99\%$ of gas and $\approx 1\%$ of dust. Initially the dust grains are of μm-size. They feel the pressure of the gas and move with it. These dust grains can

collide and stick together and form bigger grains. When the grains are of metre-size, they no longer feel the pressure of the gas and move independent of it. Now the dust grains can sediment to the midplane of the disc. In the so increased grain density in the midplane, the collision-rate of the grains is much higher (see top-left image of Fig. 2.5). When the "grains" have grown to 10-kilometre-sized planetesimals their gravitational attraction is strong enough to boost the growth (see Boss, 1998a) (see top-right image of Fig. 2.5). When these growing "planetary embryos" reach a certain mass after approximately 10^6 years ($\approx 10 M_{\text{Earth}}$), they start accreting the discs gas to form gas giant planets (see bottom-left image of Fig. 2.5). This phase lasts for approximately 10^7 years (Boss, 1998a). When there is enough disc gas left at the positions of the planet embryos in this phase the resulting planets can have up to several M_{Jupiter}. At any time of this process, not accreted planetesimals can be gravitationally slingshot out of the system (see bottom-right image of Fig. 2.5).

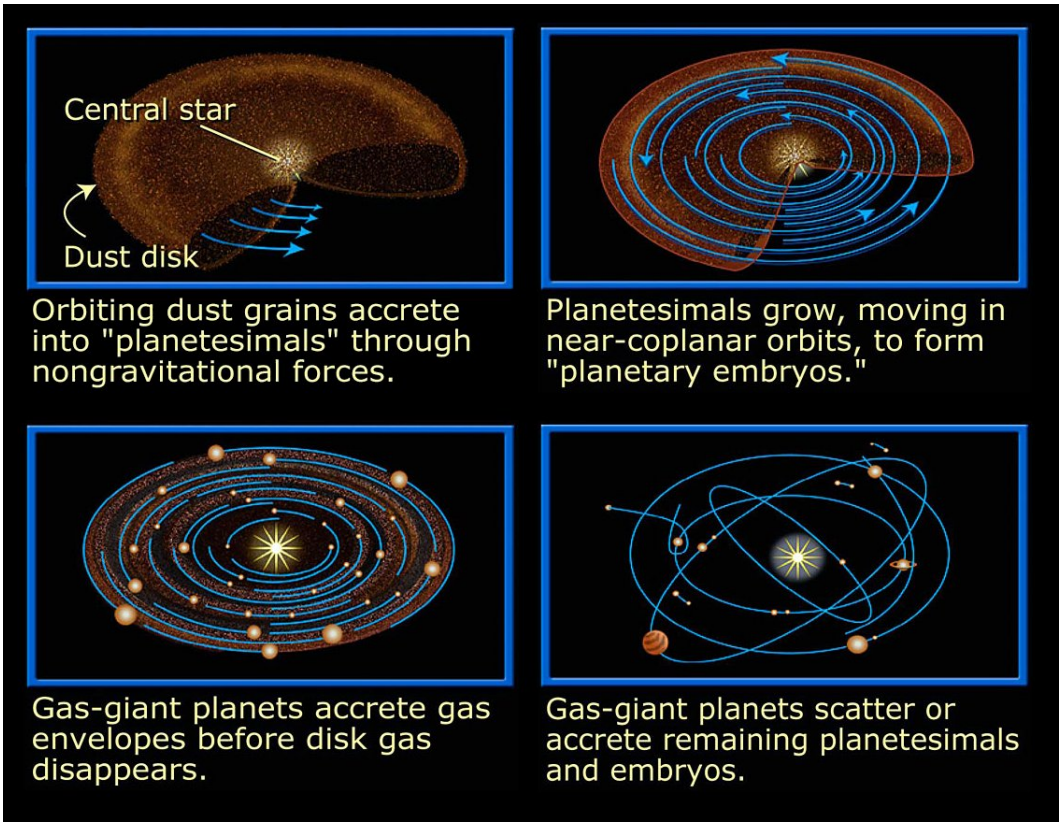

Figure 2.5: Schematic view of the core accretion model for planet formation. (Rearranged from image by NASA/ESA and Feild, A., 2003)

As already mentioned this process lasts for 10^7 years. On the other hand, the disc lifetime is only $5 \cdot 10^6$ years (see Sec. 2.3). So there seems to be a problem with the time-scale. Also there are processes reducing the planet formation efficiency. Small colliding dust grains do not only stick together. Depending on the relative velocity, they can also destroy each other increasing the time needed to form planetesimals. While very small dust grains move with the gas and kilometre-sized rocks barely feel the drag, the metre-sized clumps should be influenced by the gas. This should cause

the rapid migration of these clumps into the central star. Additionally, the forming planets can produce gaps in the disc which stops the inflow of gas (see Boss, 1998a). Therefore, the material available in the inner part of the disc can be insufficient to form gas giant planets. However, gas giant planets are indeed observed within 1 AU from their star.

The gravitational instability model

The second planet formation scenario is based on the possible gravitational instability of the discs. Toomre (1964) derived a criterion for the stability of a self-gravitating disc as

$$Q = \frac{c_s \Omega}{\pi G \Sigma}, \tag{2.6}$$

where c_s is the sound speed, Ω the local angular frequency, and G the gravitational constant and Σ the local surface density. This is basically a modified Jeans criterion considering the stabilisation by the rotation. The disc is stable against gravitational collapse for $Q \gtrsim 1$. Since the sound speed is $c_s \propto \sqrt{T}$, the disc can be destabilised by decrease of temperature or increase of the surface density Σ.

Fluctuations in the disc, for example caused by spiral arms, can lead to the formation of dense clumps. When the gas is cold enough ($Q < 1$), the gas becomes gravitationally bound and the clump is stable (see top-right image of Fig. 2.6). This clump can accrete disc material very fast (within some 10^3 years (Boss, 1998a)). Inside the clump, the dust grains can settle to the centre and form a compact core (see bottom-left image of Fig. 2.6). The clump can shrink by converting gravitational energy into thermal energy, which is radiated away. While the clump, now called planetary embryo, accretes more disc material, it can clean its orbit (see bottom-right image of Fig. 2.6).

Various investigations were performed to determine the influence of cooling on the stability of the discs (Gammie, 2001; Lodato & Rice, 2005; Rice et al., 2003, 2005). Usually cooling is modelled by the introduction of a cooling-time t_{cool}, which is related to the local angular frequency Ω by the cooling parameter $\beta = \Omega t_{\mathrm{cool}}$ (e.g. Lodato & Clarke, 2011). For β, values were found between 3 and 13 (see Meru & Bate, 2011). In this model, the discs become gravitationally unstable as a natural outcome of the cooling process. Because in the inner regions of the disc viscous heating balances the cooling and the equilibrium temperature is too high, the discs tend to fragment at the outer regions at some 10 AU. In this model, stable clumps can form outside 50 AU, but most extrasolar giant planets found so far, were found close to their central star (< 5 AU). This can not be explained by this model. On the other hand, since planets can not be formed out of some 10 AU from the central star according to the core accretion model (e.g. Levison & Stewart, 2001) the gravitational instability model is currently the only explanation at hand for planets found that far away from their host star.

Alternatively to the evolution into instability by cooling, some groups (Clarke et al., 2008; Forgan & Rice, 2009) investigated recently, whether star disc encounters play a role in the fragmentation of protoplanetary discs. While Clarke et al. (2008) found no significant influence of the encounter on the susceptibility to fragment, Forgan & Rice (2009) found, that encounters rather stabilise the discs

with compressive and shock heating, than triggering the fragmentation.

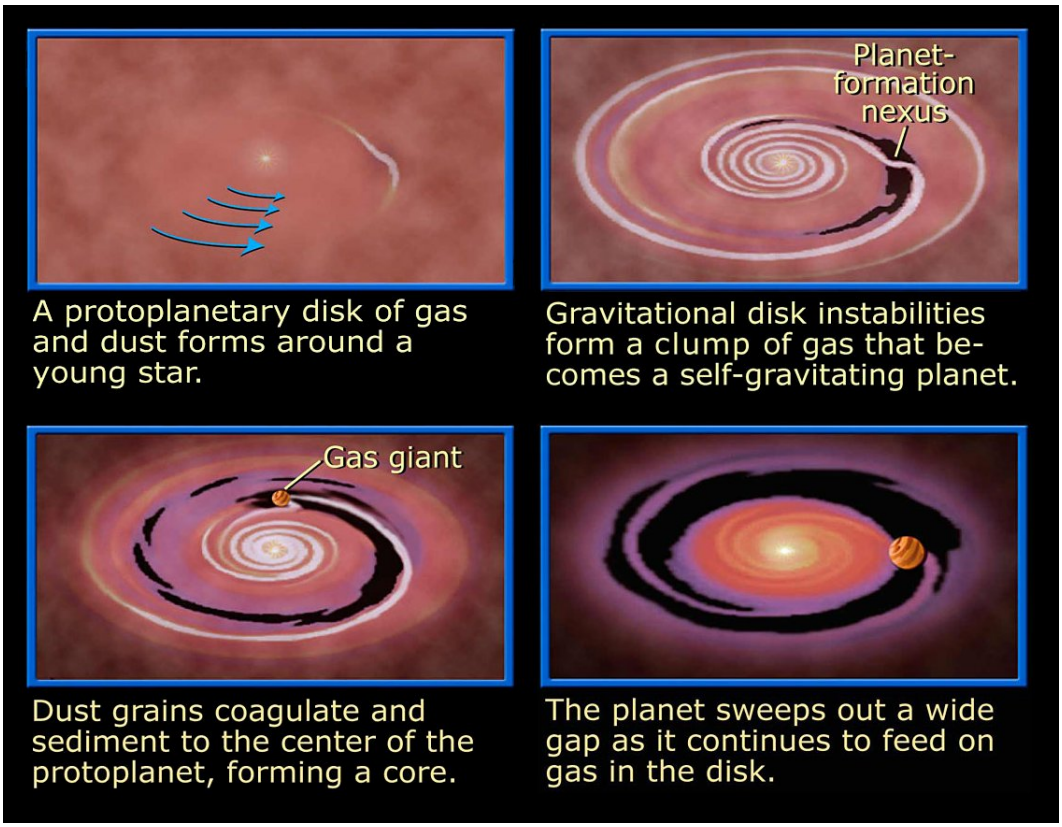

Figure 2.6: Schematic view of the gravitational instability model for planet formation. (Rearranged from image by NASA/ESA and Feild, A., 2003)

3 Smoothed Particle Hydrodynamics

Many processes described in the previous chapter can be reduced to the dynamics of a self-gravitating gas. The numerical solution of the gravitational forces will be described in Sec. 4.1. In this chapter the SPH method will be described. It is used as fluid-solver in the code developed in this work.

Discretisation of the fluid-equations

In general, fluid flows are described by the Navier-Stokes-equation

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{\nabla P}{\rho} - (\mathbf{v}\nabla)\mathbf{v},$$

where P is the pressure, ρ the density and $\mathbf{v}$ the velocity vector. The first part of the right hand side of the equation is the acceleration caused by a pressure-gradient, the second part is the acceleration due to convection. Neglecting the convective part, for the following example, one obtains,

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{\nabla P}{\rho}. \tag{3.1}$$

This equation has to be discretised to solve it numerically. The classical discretisation is done at the intersection points of a regular grid (see Fig. 3.1 a)). The x-component of the velocity at time $t + \Delta t$ is then given by

$$v_{x,i,j}(t+\Delta t) = -\left(\frac{P_{i+1,j(t)} - P_{i-1,j}(t)}{2\Delta x \rho(t)}\right)\Delta t, \tag{3.2}$$

where the subscripts i and j enumerate the grid-points in x- and y-direction and Δx is the x-distance between two grid-points.

Analog the continuity equation

$$\frac{\partial \rho}{\partial t} = \nabla \rho \mathbf{v}$$

is discretised as

$$\rho_{i,j}(t+\Delta t) = \rho_{i,j}(t)\left(\frac{v_{x,i+1,j}(t) - v_{x,i-1,j}(t)}{2\Delta x} + \frac{v_{y,i,j+1}(t) - v_{y,i,j-1}(t)}{2\Delta y}\right)\Delta t. \tag{3.3}$$

The SPH method was introduced as an alternative approach to discretise the fluid-equations by Gingold & Monaghan (1977) and Lucy (1977). Here, not the space is discretised, but the flow itself

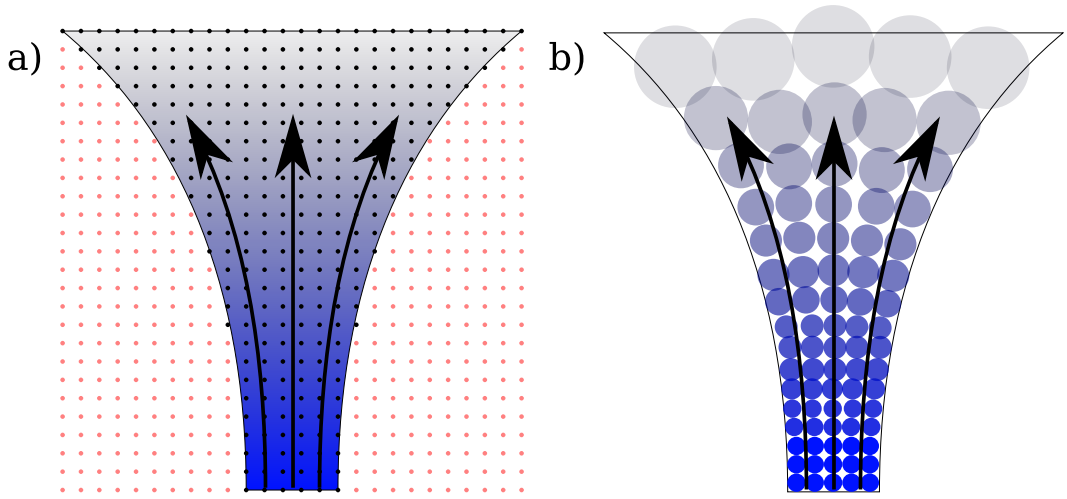

Figure 3.1: To types of discretisation of the expansion of a stream into a vacuum. a) shows the discretisation with a regular grid, as used by many numerical fluid solvers. At each point density and velocity is evaluated according to Eq. 3.2 and 3.3. At the red points density and velocity have to be evaluated without any gain. b) shows the corresponding SPH-discretisation. Here not the space, but the flow is discretised.

(see Fig. 3.1 b)). Each interpolation point is linked to a specific amount of mass. By this means, pseudoparticles are defined. This makes SPH a particle based fluid method, where the particles have intrinsic properties like density, temperature etc. The mass associated with the particle is distributed around the interpolation point with a function called 'kernel'. During the temporal evolution, these interpolation points and their associated density 'clouds' move with the fluid, what makes SPH a Lagrangian method. Because the interpolation points represent a fixed amount of mass, the resolution automatically increases at denser regions. This intrinsic resolution adaptiveness and the fact, that it does not depend on a mesh like Eulerian fluid methods, makes SPH ideal for the simulation of gas flows with strong density gradients and free-surface flows. Another advantage over grid based fluid-solver is, that mass is conserved exactly.

Additionally, this Lagrangian fluid dynamics method seems the natural counterpart of an also Lagrangian n-body method used for gravitation (see Sec. 4.1). Both methods determine the acceleration for the particles and then these particles are moved according to a given integration scheme.

3.1 A basic SPH formulation

The here presented derivation of the fundamental equations will basically follow the work of Monaghan (1992). It should be mentioned that there exist a mathematically more consistent version (see Springel & Hernquist, 2002), but here the more-widely used version by Monaghan is used. First a basic set of equations needed for the SPH method will be derived, then some improvements will be presented.

3.1.1 The SPH principle

The central idea of SPH is that any function $f(x)$ can be expressed with the delta-distribution as

$$f(x) = \int f(x')\delta(x-x')dx'.$$

In three dimensions, one can similarly define a function A of the position vector $\mathbf{r}$ in the following way

$$A(\mathbf{r}) = \int A(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}')d^3r'.$$

Defining a function W, in the following called kernel, with the property

$$\lim_{h\to 0} W(\mathbf{r},h) = \delta(\mathbf{r}),$$

where the smoothing length h is a parameter determining the width of the kernel, one can write

$$A(\mathbf{r}) = \lim_{h\to 0}\int A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)d^3r'.$$

For a small enough h follows

$$A(\mathbf{r}) \approx \int A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)d^3r'. \tag{3.4}$$

With the relation between mass m and density ρ $dm = \rho(\mathbf{r})d^3r$ one finally obtains

$$A(\mathbf{r}) = \int \frac{A(\mathbf{r}')}{\rho(\mathbf{r}')}W(\mathbf{r}-\mathbf{r}',h)dm'.$$

With this equation it is possible to interpolate any scalar field A at the location $\mathbf{r}$ with the value of this field at different locations $\mathbf{r}'$. For later computational usage, the integral has to be discretised:

$$A_a = \sum_b \frac{m_b}{\rho_b}A_b W(\mathbf{r}_a-\mathbf{r}_b,h). \tag{3.5}$$

Here the subscripts a denotes a specific interpolation point and b enumerates other interpolation points. From (3.5) one can see, that an arbitrary property A_a of the interpolation point a can be computed by summing up the properties A_b of the points b weighted with their masses m_b, densities ρ_b and the function W.

3.1.2 Density and acceleration

With $\rho(\mathbf{r})$ as function $A(\mathbf{r})$, one obtains an expression for the density at point a as

$$\rho_a = \sum_b m_b W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{3.6}$$

Furthermore, the gradient of a function A can be expressed as

$$\nabla A_a = \sum_b \frac{m_b}{\rho_b} A_b \nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{3.7}$$

Until here, the derivation was straight forward and the obtained equations are very generic. For later application, the arbitrary function A has to be identified with physical quantities.

The accelerations $\mathbf{a}$ of the interpolation points is given by the Navier-Stokes-equation as

$$\mathbf{a} = -\frac{\nabla P}{\rho}.$$

Straight forward transformations leads to

$$\mathbf{a}_a = -\frac{1}{\rho_a} \sum_b \frac{m_b}{\rho_b} P_b \nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{3.8}$$

As pointed out by Monaghan (1992), this equation does not guarantee the conservation of linear and angular momentum. To avoid this, it is better to symmetrize the formula in a way, that the properties of both interpolation points a and b are taken into account:

$$\begin{aligned} \nabla\left(\frac{P}{\rho}\right) &= \frac{\nabla P}{\rho} - P\frac{\nabla \rho}{\rho^2} \\ \Rightarrow \frac{\nabla P}{\rho} &= \nabla\left(\frac{P}{\rho}\right) + P\frac{\nabla \rho}{\rho^2}. \end{aligned} \tag{3.9}$$

Together with (3.7) follows

$$\begin{aligned} \mathbf{a}_a = \left(\frac{\nabla P}{\rho}\right)_a &= \sum_b \frac{m_b}{\rho_b}\frac{P_b}{\rho_b} \nabla W(\mathbf{r}_a - \mathbf{r}_b, h) + \frac{P_a}{\rho_a^2} \sum_b \frac{m_b}{\rho_b} \rho_b \nabla W(\mathbf{r}_a - \mathbf{r}_b, h) \\ &= \sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2}\right) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \end{aligned}$$

This gives the force term

$$\mathbf{F}_a = -m_a \sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2}\right) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{3.10}$$

Using (3.10) instead of (3.8) improves the momentum conservation (see Monaghan, 1992; Monaghan & Lattanzio, 1985). Because $\nabla W(\mathbf{r}_a - \mathbf{r}_b, h) = -\nabla W(\mathbf{r}_b - \mathbf{r}_a, h)$, the use of both pressures P_a and P_b

in (3.10) guarantees, that the forces are symmetric:

$$\mathbf{F}_{a\to b} = -\mathbf{F}_{b\to a}.$$

3.1.3 Thermal energy equation

With the equations 3.6 and 3.10 it is already possible to implement a very basic fluid solver. However, these equations do not guarantee energy conservation.

When two interpolation points move towards each other with contrary velocities, they both slow down because their kinetic energy is transformed into thermal energy. The thermal energy u is defined in terms of pressure, density and velocity as

$$\frac{du}{dt} = -\frac{P}{\rho}\nabla\mathbf{v}.$$

This can be transformed in a similar way as (3.9):

$$\begin{aligned}
&\nabla\rho\mathbf{v} = \rho\nabla\mathbf{v} + \mathbf{v}\nabla\rho \\
\Rightarrow &\nabla\mathbf{v} = \frac{\nabla(\rho\mathbf{v}) - \mathbf{v}\nabla\rho}{\rho} \\
\Rightarrow &\frac{du}{dt} = -\frac{P_a}{\rho_a^2}\sum_b m_b(\mathbf{v}_b - \mathbf{v}_a)\nabla W(\mathbf{r}_a - \mathbf{r}_b, h).
\end{aligned} \tag{3.11}$$

Here only the pressure of interpolation point a is taken into account. Again, for better conservation behaviour, another equation has to be derived:

$$\begin{aligned}
&\nabla\left(\frac{P\mathbf{v}}{\rho}\right) = \frac{P}{\rho}\nabla\mathbf{v} + \mathbf{v}\nabla\left(\frac{P}{\rho}\right) \\
\Rightarrow &\frac{P}{\rho}\nabla\mathbf{v} = \nabla\frac{P\mathbf{v}}{\rho} - \mathbf{v}\nabla\left(\frac{P}{\rho}\right) \\
\Rightarrow &\frac{du}{dt} = -\sum_b \frac{m_b P_b}{\rho_b^2}(\mathbf{v}_b - \mathbf{v}_a)\nabla W(\mathbf{r}_a - \mathbf{r}_b, h).
\end{aligned} \tag{3.12}$$

Here only the pressure of interpolation point b is taken into account. By averaging (3.11) and (3.12) one obtains the symmetric energy-equation as

$$\frac{du}{dt} = -\frac{1}{2}\sum_b m_b\left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2}\right)(\mathbf{v}_b - \mathbf{v}_a)\nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{3.13}$$

3.1.4 Summary

The conservation laws of mass, momentum and energy are now given by the set of equations (3.6), (3.10) and (3.13), here recalled for an overview. This set of equations are the basis of the SPH method.

$$\rho_a = \sum_b m_b W(\mathbf{r}_a - \mathbf{r}_b, h)$$
$$\mathbf{F}_a = -m_a \sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} \right) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h)$$
$$\frac{du}{dt} = -\frac{1}{2} \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} \right) (\mathbf{v}_b - \mathbf{v}_a) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h)$$

3.2 Improvements to the basic formulation

The basic SPH equations derived above can already be used for the simulation of fluids. In the following an improvement for the application to shock waves will be derived. Additionally, a method will be introduced, to reduce the computation-time fundamentally.

3.2.1 Artificial viscosity

SPH, as well as many other numerical solver for differential equations, has limited capability to resolve strong gradients as occurring for example in shocks. Unlike in sound waves, the physical properties, like temperature, pressure or density, change in shocks very fast and by huge amounts. Due to the Lagrangian nature of SPH, in shock waves it is possible that the interpolation points approach very close and even penetrate each other. Since the interpolation points represent macroscopic matter flows, this is unphysical and should not happen. To avoid this, an artificial viscosity as introduced by Monaghan & Gingold (1983) is used in most SPH-codes. As the term “artificial” already implies, this is not strictly analytical derived from the physical viscosity in fluids, but more an empirical approach.

To include this artificial viscosity, the term

$$\left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} \right)$$

in (3.10) and (3.13) is replaced by

$$\left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} + \Pi_{ab} \right),$$

with

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

and

$$\mu_{ab} = \frac{h \mathbf{v}_{ab} \cdot \mathbf{r}}{\mathbf{r}_{ab}^2 + \eta^2},$$

where α, β and η are dimensionless parameters, $\bar{c}_{ab}$ is the mean sound speed and $\bar{\rho}_{ab}$ the mean density.

This viscosity is only non zero, if the two particles move towards each other. In practice, the parameters α and β are usually chosen as $\alpha \approx 1$ and $\beta \approx 2\alpha$. η is just needed to prevent singularities when $\mathbf{r}_{ab}$ is close to zero and is usually set to $0.1h$ (see Monaghan, 1992).

Integrating this improvement one obtains

$$\mathbf{F}_a = -m_a \sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} + \Pi_{Nb} \right) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h) \tag{3.14}$$

$$\frac{du}{dt} = -\frac{1}{2} \sum_b m_b \left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2} + \Pi_{Nb} \right) (\mathbf{v}_b - \mathbf{v}_a) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{3.15}$$

3.2.2 The kernel

Up to this point, to calculate a property of an interpolation point the properties of the $N-1$ other interpolation points have to be summed up, which results in an overall computation cost of $\mathscr{O}\left(N^2\right)$. However, by using a function W as kernel, with $\lim_{h \to 0} W(\mathbf{r}, h) = \delta(\mathbf{r})$, the majority of the points will contribute almost nothing to the properties of point a. The reason is that the distance is large and so the kernel for this pair of points very small. One can benefit from this by using a kernel function with compact support[1]. One such kernel is a spline of the form first used by Monaghan & Lattanzio (1985):

$$W(r,h) = \sigma \begin{cases} 1 - \frac{3}{2}\left(\frac{r}{h}\right)^2 + \frac{3}{4}\left(\frac{r}{h}\right)^3 & \text{if} \quad 0 \le \left(\frac{r}{h}\right) \le 1 \\ \frac{1}{4}(2 - \left(\frac{r}{h}\right))^3 & \text{if} \quad 1 \le \left(\frac{r}{h}\right) \le 2 \\ 0 & \text{otherwise,} \end{cases}$$

[1] A function with compact support is zero everywhere except a small range.

where

$$\sigma = \begin{cases} \frac{2}{3h} & \text{1D} \\ \frac{10}{7\pi h^2} & \text{2D} \\ \frac{1}{\pi h^3} & \text{3D} \end{cases}$$

to fulfil the normalisation criterion $\int_{\infty}^{\infty} W(r,h)d^d r = 1$. This kernel has compact support, its second derivative is continuous and the dominant error term in the integral interpolant is $\mathcal{O}(h^2)$ (Monaghan, 1992).

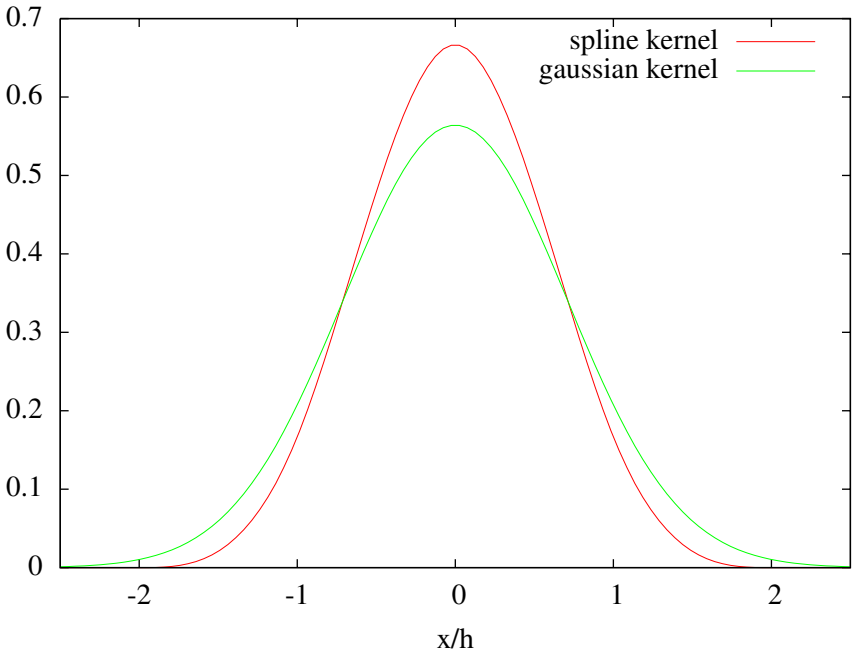

Figure 3.2: Spline kernel for 1D compared to normalised Gaussian.

Figure 3.2 shows a comparison of the spline kernel with a Gaussian with same normalisation. While the exact form of the kernel is of minor importance, it is crucial that it is smooth close to the maximum and where it approaches zero. Both is given for the Gaussian and the spline kernel. For $|x| \geq 2h$ the spline kernel equals zero.

Monaghan (2005) noted that various other kernels have been studied but none produced significantly better results than this one.

Because the kernel is a polynomial, the first derivative can be easily derived as

$$\nabla W(\mathbf{r},h) = \sigma \begin{cases} \left(\frac{9r-12h}{4h^3}\right)\mathbf{r} & \text{if} \quad 0 \leq \left(\frac{r}{h}\right) \leq 1 \\ \frac{-3\left(2-\left(\frac{r}{h}\right)\right)^2}{4rh}\mathbf{r} & \text{if} \quad 1 \leq \left(\frac{r}{h}\right) \leq 2 \\ 0 & \text{otherwise,} \end{cases}$$

with $r = |\mathbf{r}|$ and σ as noted above. With this kernel the computation time per interpolation point is linear in the number of interpolation points within the $2h$ smoothing-sphere which is a lot better than $\mathscr{O}\left(N^2\right)$ for good choices of h.

3.2.3 Choosing the neighbours

By using this spline kernel the sums in the SPH-formulas change from $\sum_b^N$ to $\sum_b^{\tilde{N}<<N}$, where $\tilde{N}$ is the number of interpolation points in the region around a where $W(\mathbf{r}_a - \mathbf{r}_b, h) \neq 0$. In principle, there are two methods to specify $\tilde{N}$:

- By choosing one h for all interpolation points. Then $\tilde{N}$ is the number of interpolation points inside a sphere with radius $2h$ around interpolation point a. In this case, the algorithm for finding the interaction partners can be very easy because the interpolation points can be sorted in equal-sized boxes and then only the neighbouring boxes of an interpolation point have to be searched for neighbours. On the other hand, this would destroy the resolution adaptiveness and can result in long computation times if the interpolation points cluster.

- By choosing h individually for each interpolation point for example in a way that there is always the same mass inside the kernel or that $\rho h^3 = constant$. The first would be the same as choosing a fixed $\tilde{N}$ for all interpolation points. This would reduce the runtime of the force summation to $\mathscr{O}\left(N \cdot \tilde{N}\right)$.

Even though the SPH-formulas have already been symmetrised above, there is still an error source in the way how the neighbouring interpolation points of a are determined during the computation. Using the $\tilde{N}$ closest interpolation points as neighbours for a corresponds to a criterion like

$$b \text{ neighbour of } a, \text{ if } dist(a,b) \leq 2h(a), \tag{3.16}$$

where $h(a)$ is here defined by the distance to the furthest of the $\tilde{N}$ neighbours.

As shown in Fig. 3.3, it is likely in strong density gradients, that a particle a is neighbour of a particle b but b is not a neighbour of particle a. That means that the force from b to a is taken into account, but not vice versa. This corrupts the momentum conservation achieved with the derivation of the symmetric force equation (3.10) in case of strong density gradients.

To avoid this problem another criterion for the interaction partners of interpolation point a is needed:

$$b \text{ neighbour of } a, \text{ if } dist(a,b) \leq max(2h(a), 2h(b)). \tag{3.17}$$

This has to be considered when searching the interaction partners.

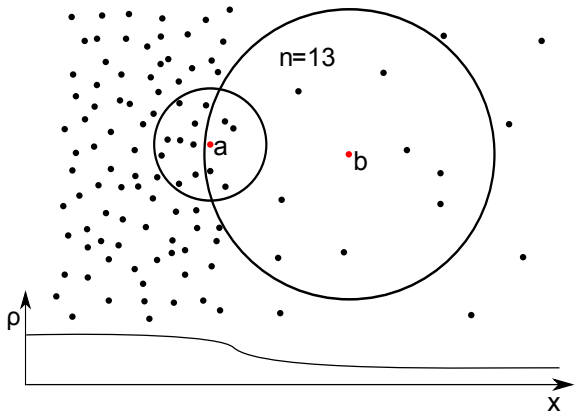

Figure 3.3: Example of a particle distribution where a symmetric neighbour criterion is needed to avoid violation of momentum conservation.

3.3 A resolution requirement

In simulations of protoplanetary discs, the discs heated for example by viscosity, are typically cooled by assuming a cooling-time t_{cool}, which is related to the local dynamical time Ω by the cooling parameter $\beta = \Omega t_{\text{cool}}$ (e.g. Lodato & Clarke, 2011). In the last years several numerical investigations tried to find reasonable values for β. Recent investigations by Lodato & Clarke (2011) showed that some results obtained by SPH simulations of fragmenting protoplanetary discs can be reinterpreted in a way that the effects found in the simulations are caused by insufficient resolution. For a disc with density ρ

$$\rho(r,z) = \frac{\Sigma(r)}{\sqrt{2\pi}H} exp\left(-\frac{z^2}{2H^2}\right),$$

where Σ is the previously used surface-density, z is the coordinate perpendicular to the discs plain and H is the so called scale height Williams & Cieza (2011), physical fragmentation can occur when

$$\frac{h}{H} < 1,$$

where h is the SPH smoothing length. Together with the Toomre stability parameter

$$Q = \frac{c_{\text{s}}\Omega}{\pi G\Sigma},$$

where c_{s} is the sound speed and G the gravitational constant Toomre (1964),

$$m(r) = \Sigma r^2 / M_{\text{star}}$$

and

$$\rho h^3 = \eta m_\mathrm{p},$$

where η is a parameter for SPH and m_p the mass of one particle according to $m_\mathrm{p} = M_\mathrm{disc}/N$, a criterion for the minimal particle number needed to ensure sufficient resolution can be constructed

$$\frac{h}{H} \approx \left(\frac{\eta}{m(r)}\right)^{3/2} \left(\frac{2p}{\pi^2 Q^2 N}\right)^{1/2}.$$

For a power-law surface-density with index $p = 1$ this can be evaluated at the outer edge to obtain

$$\frac{h}{H} \approx 0.3 \left(\frac{q}{0.1}\right)^{-2/3} \left(\frac{N}{2.5 \cdot 10^5}\right),$$

where $q = M_\mathrm{disc}/M_\mathrm{star}$.

To get a vertical resolution at the outer edge of the disc of $H/5$ for a disc with $q = 0.1$ approximately 850 000 particles are needed. For $H/10$ nearly $7 \cdot 10^6$ particles are needed.

To simulate such a disc with this resolution a fast simulation code is needed.

4 The tree-code PEPC

While thermodynamic forces are of short-range character, gravitational or coulomb forces with their $1/r$-potential are long-range interactions. This means that a computational solution of these forces can not be calculated by just taking the properties of the direct environment into account but has to take the whole simulation volume into account. The equations of motion for N gravitationally interacting bodies are given by the N coupled differential equations of the form

$$m_i \frac{d\ddot{\mathbf{r}}_i}{dt} = \mathbf{F}(\mathbf{r}_{1..N}),$$

where m_i is the mass of body i, $\mathbf{r}_i$ its position vector and $\mathbf{F}$ is the force caused by all particles at their positions $\mathbf{r}_{1..N}$. To solve these equations numerically, for each body $N-1$ forces have to be calculated, which leads to a $\mathscr{O}\left(N^2\right)$ computation-time. This limits the manageable total number of bodies (or particles) on today's fastest supercomputers due to limited computation resources. When many time-steps have to be calculated, only a few 100 000 particles can be treated. To simulate more particles - or the same amount of particles in a more efficient way - a better-scaling method is needed. Among several such methods which all abstract in some way from the individual particle properties to a coarser level, tree-codes, as introduced by Barnes & Hut (1986), are often used for the numerical solution of gravitational interactions in astrophysical simulations.

4.1 Tree-codes in general

In computer sciences a tree is a special type of graph. Graphs are data-structures often used in computer science to organise a set of elements (nodes). The nodes are connected pairwise with edges often denoting a relation between the two nodes they connect (see Fig. 4.1 a). When there is always only one path from each node to all other nodes, this graph is a tree (see Fig. 4.1 c). When the edges denote directed relations (like a < b, a > b, a 'is child of' b, a 'contains' b), the tree is a directed tree (see Fig. 4.1 d). In this case the nodes can be rearranged hierarchically (see Fig. 4.1 e).

Tree-codes use trees as data-structures to store and access information about a particle-distribution. The tree-nodes are spatial subsets of the whole simulation box and the edges denote the relation 'a contains b'. The tree-nodes are linked to information about the particle-distribution inside the simulation volume they represent.

To build the tree a box enclosing the whole simulation volume is identified with the root of the tree and then the box is consecutively subdivided into smaller child boxes, which are linked with the

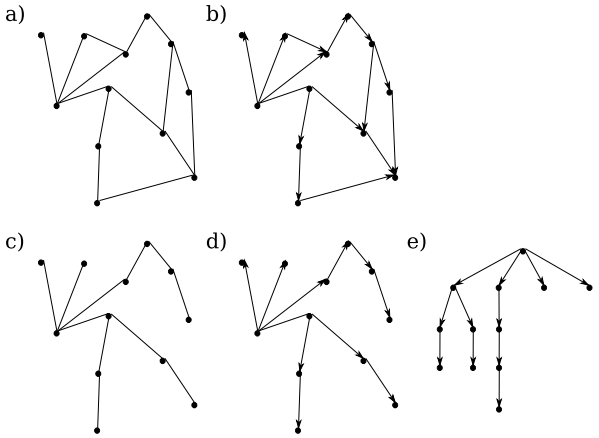

Figure 4.1: Samples for graphs and trees: a) graph b) directed graph c) tree d) directed tree e) Nodes from d) rearranged.

parent box. If one box contains only one particle, the subdivision is stopped and the particle is linked to this box. This can be done by dividing the dimensions in a rotating manner or by dividing each dimension at every step. The first would result in a binary-tree, the second in a binary-, quad- or oct-tree depending on the dimensionality of the problem. For a 2D sample of a particle distribution with associated quad-tree see Fig. 4.2.

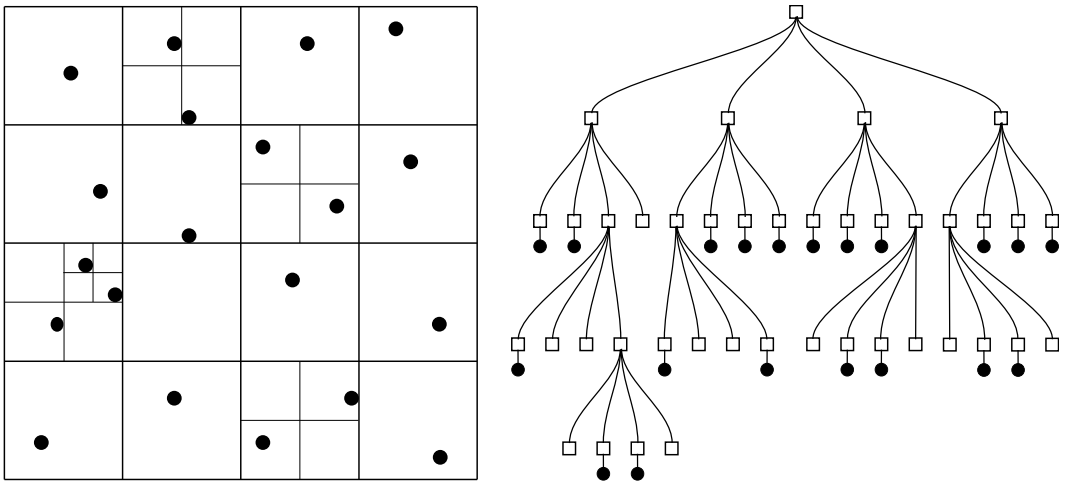

Figure 4.2: 2D particle distribution with corresponding quad-tree. The root of the tree is the full simulation-box and the subdivision is stopped, when there is only one particle in a cell. At every level, the order of the nodes is bottom-left, bottom-right, top-left, top-right.

For each box, pseudoparticles are constructed representing the particle-distribution inside the box. In the simplest case, these pseudoparticles have the total mass of all particles in the box and are located at their centre of mass.

Then for the force computation particle-pseudoparticle interactions are taken into account rather than particle-particle interactions. To decide which pseudoparticle to use for the interaction, a criterion is constructed based on the distance d between particle and pseudoparticle and the size s of the

box associated with the pseudoparticle

$$s/d \leq \theta, \tag{4.1}$$

where θ is a dimensionless parameter which has to be chosen adequately (e.g. Pfalzner & Gibbon, 1996). $\theta = 0$ results in a direct particle-particle force summation with $\mathcal{O}\left(N^2\right)$ run-time. For $\theta > 0$, only for close interactions highly resolved pseudoparticles are used, for distant interactions a few big pseudoparticles are used. This reduces the computational effort for the force-summation to an $\mathcal{O}\left(N \log N\right)$ run-time.

From a physical point of view, the total mass pseudoparticle at the centre of mass associated with a tree-node is the first order of the multipole expansion of the gravitational potential caused by the particles in the box.

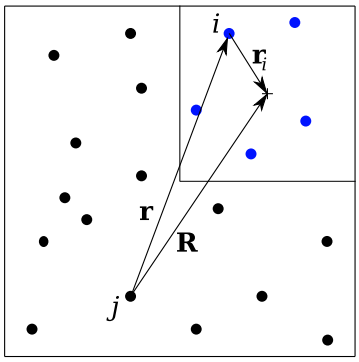

Figure 4.3: Interaction of particle j with a pseudoparticle. $\mathbf{R}$ is the vector from the particle j to the centre of mass of the pseudoparticle. $\mathbf{r}_i$ is the vector from a particle contributing to the pseudoparticle to its centre of mass.

For a particle j, the gravitational potential caused by the particles in a tree-node is given by

$$\Phi(\mathbf{R}) = \sum_i \Phi(\mathbf{R} - \mathbf{r}_i),$$

where $\mathbf{R}$ is the vector from the particle to the centre of mass of the particles in the tree-node and $\mathbf{r}_i$ is the vector from the particle i to the centre of mass (see Fig. 4.3). For a $1/r$ potential, the multipole expansion is given by

$$\Phi(\mathbf{R}) = -\sum_i m_i \left[1 - \mathbf{r}_i \frac{\partial}{\partial \mathbf{r}} + \frac{1}{2}\mathbf{r}_i \mathbf{r}_i \frac{\partial}{\partial \mathbf{r}} \frac{\partial}{\partial \mathbf{r}} + \ldots\right] \frac{1}{R}$$

with the masses m_i of the particles in the tree-node (e.g. Pfalzner & Gibbon, 1996). The terms in the square brackets are, from left to right, the monopole, the dipole and the quadrupole moment. For a pure gravitational tree-code, the dipole moment vanishes. The force on particle j can then be obtained

with

$$\mathbf{F}_j(\mathbf{R}) = -m_j \frac{\partial}{\partial \mathbf{r}} \Phi(\mathbf{R}).$$

The multipole moments of a tree-node only depend on the mass distribution inside the node. This makes it possible to pre-compute the multipole moments. This is important, because the multipoles are used very often and pre-computing them once saves a lot of computation time. To calculate the multipole moments of a bigger tree-node, the multipole moments of its child-nodes can be used (see Pfalzner & Gibbon, 1996).

In more sophisticated tree-codes, higher multipole moments of the potential are used to reduce errors in the force summation instead of reducing θ. This needs more memory but is often faster. Because the computation time of a multipole moment of order[1] l has an $\mathscr{O}\left(l^2\right)$ dependence (see Pfalzner & Gibbon, 1996), a good compromise between the highest used multipole moment and the choice of θ has to be found. In practice $\theta \lesssim 0.6$ is often sufficient. In PEPC (Pretty Efficient Coulomb Solver) the multipoles up to the quadrupole term are implemented.

4.2 Parallel tree code - PEPC

4.2.1 Parallel computing

To simulate large physical problems in a reasonable (wall) time, there are recently three possible ways. At first, there are the classical supercomputers. These are basically standard computer components like CPUs and main memory with some kind of fast interconnect. The problem is then split into small pieces and computed on the CPUs in parallel. The available supercomputers belong in principle to two different types.

Shared-memory computers

The first type are so called shared-memory computers. They have several CPUs connected in a way, that they can access the same memory. Such the processes running in parallel on the CPUs can work on the same data simultaneously. Recent shared-memory supercomputers like the SGI ALTIX UV provide up to 4096 CPUs accessing up to 32 TB of shared main memory (PSC, n.d.). Programs for these types of supercomputers are easier to parallelise because all processes have the same data. It is possible to parallelise loops over big arrays in a way, that the parallel processes work on different regions of the arrays, e.g. process 1 of p processes computes array entry 1 to N/p, process 2 computes array entry $N/p+1$ to $2N/p$ and so on. With modern programming techniques like OpenMP the parallelisation of the most expensive loops in the program can normally be achieved within hours.

[1] The monopole moment has $l = 0$, the dipole moments $l = 1$, and so on.

Distributed-memory computers

The other type are the distributed-memory computer. These are in principle several shared-memory computers, called nodes, connected through a very fast network. The processes running on one nodes can only access the data in their local shared-memory and if they need information from a process running on another node they have to retrieve this information via the network. One widely used programming technique for this type of computers is MPI (Message Passing Interface). It provides functions to send data, e.g. particle coordinates, to another process. The programmer has to take care of the correct collection of the data from the local arrays of the source process, as well as to include the received data into the local arrays of the target process. Because the bandwidth of the network communication is limited and it is affected by latency, every communication has to be well-considered and optimised. On the other hand the local memory of the nodes is limited, so that it is in general not possible to communicate all information to all processes. Hence it can be quite challenging to write a program, that scales good on huge distributed-memory computers. But reward for the higher effort it a program, that in principle can benefit from a higher number of processors and more memory.

Hardware accelerators

The alternative to supercomputers for fast computing is the use of hardware accelerators like those developed from recent high-end graphic-cards. These cards have their own memory and a processor capable of computing large sets of data in parallel. When a problem is small enough to fit in the memory of an accelerator and the computational expensive parts of the problem can be transformed in a way that it can be handled by the accelerator, the performance gain can be very large. When the problem does not fit into the memory of one accelerator it is possible to combine the accelerators with the classical shared- or distributed-memory supercomputers. In this case data exchange between two accelerators via the shared-memory or via the network can be quite time consuming compared to the computation time. This can nullify the performance gain.

The type of code developed in this work requires a lot of communication (see Sec. 5.3.3). Therefore it is expected that this code would benefit from accelerator cards.

Scaling

When a program runs on a parallel computer with p processes in parallel, it is in general not exactly p-times faster. As first described by Amdahl (1967) the speedup of a parallel program is limited by these serial components. The speedup of a program running on a distributed-memory computer is often limited by the communication. Additionally, the run-time of most algorithms does not increase linearly with problem size. The dependence of the program run-time on the problem size or number of parallel processes is called scaling.

4.2.2 Parallel tree code - PEPC

During this work the parallel tree-code PEPC was extended with a SPH-module for hydrodynamics. PEPC is parallelised with MPI for distributed memory computers, In the following the parts most important for this work will be described in detail.

PEPC is a so-called hashed-oct-tree based tree-code written by Gibbon (2003) for plasma-simulations. It follows the implementation described by Warren & Salmon (1995) and is parallelised with MPI to optimise it for distributed-memory-computers. As noted in Gibbon et al. (2010a) it is capable of simulating more than 10^8 particles with up to ≈ 8000 parallel processes. However, recent changes made PEPC capable of running on all 73 728 compute-nodes (with 294 912 cores) of Jugene (see Winkel et al., 2011). Because Jugene is still the supercomputer with the highest number of nodes in the world[2], this makes PEPC the highest scaling tree-code. In Fig. 4.4, the scaling of the latest version for homogeneous particle distributions with different particle numbers N can be seen. The solid lines represent the total run-times. A linear decrease of run-time with increasing number of cores is ideal. Note the near ideal decrease of run-time with increasing number of cores of the blue solid line, almost up to the full machine.

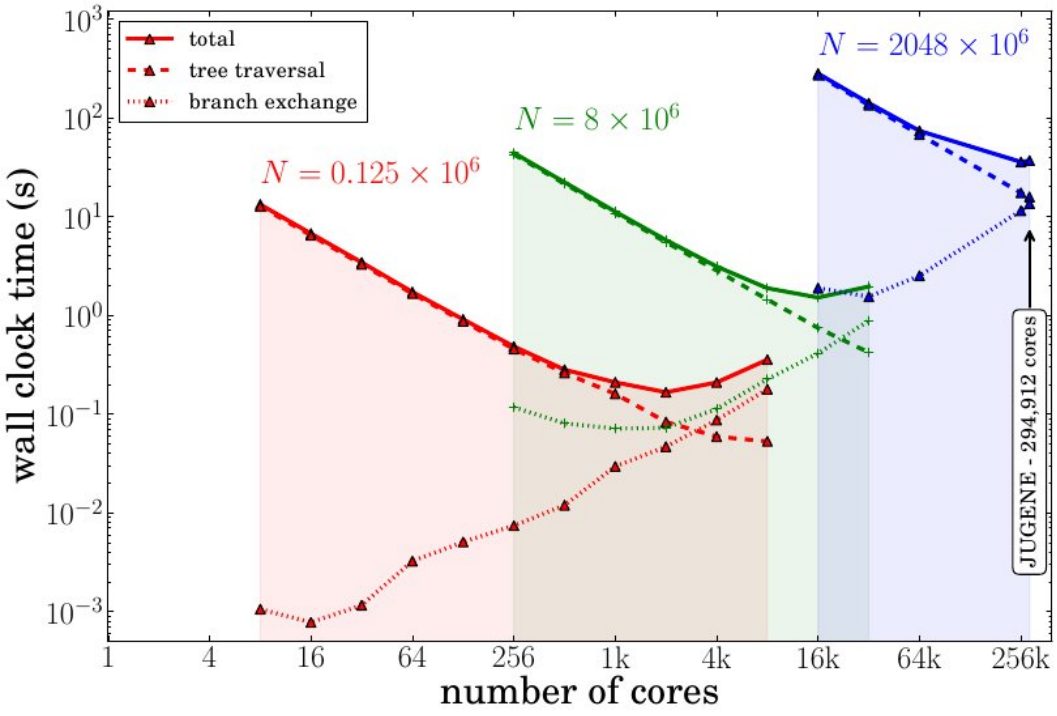

Figure 4.4: Runtime of PEPC for simulations of homogeneous particle distributions with different particle numbers N. The solid lines represent the total run-times. A linear decrease of run-time with increasing number of cores is ideal. Note the near ideal scaling for huge particle numbers (blue line). (Image courtesy of Winkel et al., 2011)

For any parallel program on a distributed memory machine, it is necessary to distribute the data somehow to the processes. To reduce communication to a minimum, frequently used date should be

[2]For the full list of the fastest computers see `http://www.top500.org`.

kept together. For particles distributed homogeneously in the simulation box this means, that the data of one process preferentially cover a contiguous space region. The distribution of simulated space across the main-memory of the processors is called domain decomposition.

4.2.3 Domain decomposition and tree construction

To identify particles and tree-nodes over process-boundaries, unique identifier are needed, provided by coordinate-based keys.

Key construction

To construct these keys, first the maximum extension of the three dimensions *boxsize* is determined. With a maximum number of n_{lev} tree-levels, the bisection of every dimension per level leads to the box-length[3] s on the finest level with $s = boxsize/2^{n_{lev}}$. The finest level can be filled with up to $2^{n_{lev}}$ tree-nodes per dimension. Then the three floating-point coordinates of the particles are transformed into three integers giving the number of finest-level-box-lengths s which fit between the lower simulation-box-border and the particle with

$$ix = (x - xmin)/s,$$

where *xmin* is the minimum x-coordinate of all particles (*iy* and *iz* likewise). From the binary representations of these integers, the oct-tree can now be derived with simple arithmetic operations because each of these integers already has the form of a one-dimensional binary tree: If the most significant bit[4] equals 0, the particle is located in the left half of the simulation-box, if it equals 1 the particle is located in the right half. If the n-th most significant bit equals 0 the particle is located in the left half of the n-th level box, if it equals 1 the particle is located in the right half of that box.

Figure 4.5 shows how in the 2D-case a quad-tree is constructed from the integer-coordinates. The x-coordinate of a column can be found by reading the digits written under this column from the outer to the inner one as a binary number. For example, for the column with the red 2 one finds 1010_2, where the subscript denotes the numeral system. As denoted above, this equals the number of columns left of that box, here 10_{10}. The leading 1 (in binary notation) tells that the box lies in the right half of the top-level box. Similarly one obtains 1001_2 for the y-coordinate of this box, which corresponds to 9_{10}. Here the leading 1 tells that the box lies in the upper half of the top level box. Enumerating the four quarter-boxes on every level with 0 to 3, one can transform the bits of the integer-coordinates directly into the digit sequence called key k via

$$k = p + \sum_{j=0}^{n_b-1} 4^j (2 \cdot bit(i_y, j) + bit(i_x, j)), \tag{4.2}$$

[3] The edge length of a tree-node.
[4] The most significant bit is the bit with the highest value.

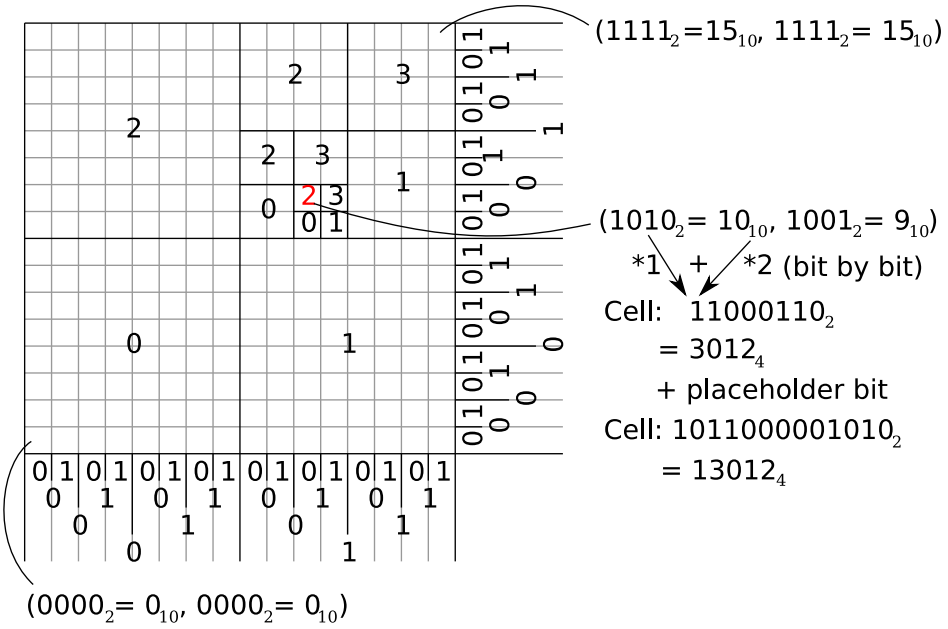

Figure 4.5: Sketch for the key construction from integer coordinates. The box with the red 2 is the 11$^{\text{th}}$ box from left and the 10$^{\text{th}}$ box from bottom. Because counting starts from 0 here, the box has the integer coordinates (10,9) as shown in the sketch (the subscript denote the numeral system). The bits from these coordinates binary representation are then interleaved to construct the cells key.

where the function $bit(i, j)$ selects the j-th bit of the integer coordinate i, n_b is the total number of bits (here 4) and p is a placeholder to distinguish between the keys 00_2 and 000_2 which are written as 100_2 and 1000_2 with the placeholder bit (see Gibbon et al., 2010b). So for the red 2, one obtains the key $k = 1011000001010_2 = 13012_4$. Conversely one can learn from that key that the box is a fourth-level box (five digits minus one for the placeholder), in the top-level-box it is located in the top-right quarter (enumerated with 3), in this quarter it is located in the bottom-left quarter (enumerated with 0) and so on.

For the 3D-code, the keys are computed as

$$k = p + \sum_{j=0}^{n_b-1} 8^j (4 \cdot bit(i_z, j) + 2 \cdot bit(i_y, j) + bit(i_x, j)). \tag{4.3}$$

By this means a single integer-key can be constructed from the three floating-point coordinates to identify each particle. In PEPC the three 8-byte floating-point coordinates are transformed into one 8-byte integer. Without the placeholder-bit, 63 bits of the integer are used for position-information instead of the $3 \cdot 64 = 192$ bits of the three real numbers. Because only 21 of these 63 bits can be used per dimension, the spatial resolution of the key-space is limited to $2^{21} \approx 2 \cdot 10^6$ finest-level-boxes per dimension within the simulation-box. Therefore, the maximum particle-density contrast is limited by the code-design.

Domain decomposition and hashing

For the domain-decomposition, the particles are sorted according to the integer-values of the keys (see e.g. Gibbon et al., 2010b). This 1D particle distribution is divided into p (p is the number of processes) equal-load chunks by weighting the particles with their computational effort, determined in the previous time-step. Each of these particle chunks is assigned to one process and the particles and their properties are sent to their associated processes. Figure 4.6 shows the resulting space-filling-curve in case of a fully filled finest level and the assignation to five domains.

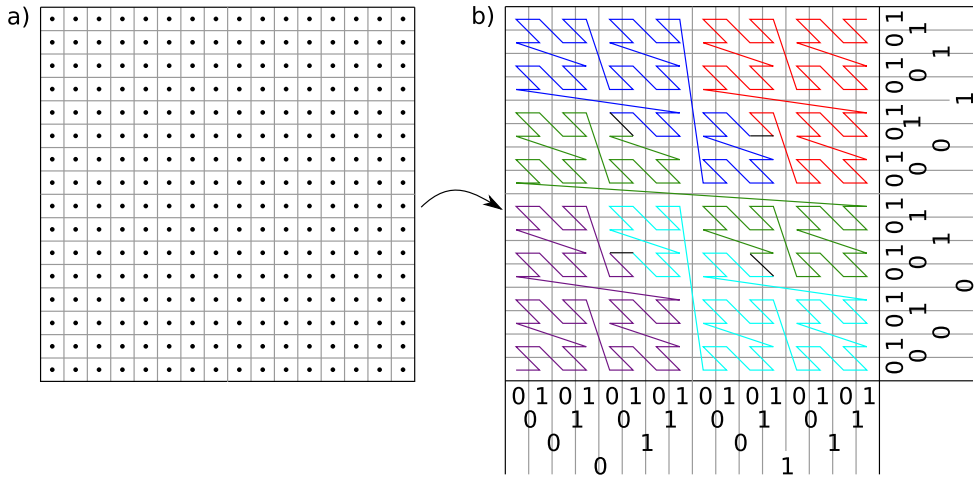

Figure 4.6: Quad-tree with particles filling the finest tree-level (a) and corresponding space-filling-curve (b). The colours denote five different domains over which the curve is distributed.

In general, the keys of particles relatively close to each other differ mainly in the least significant bits. The keys of particles from opposite parts of the whole simulation box differ mainly in the most significant bits. For example, all particles in the red domain in Fig. 4.6 have 11_2 as most significant bits. This provides a method for rapid access of the information associated with a tree-node. Because the particles of one domain are in general relatively close to each other (not true for all particles of the green domain in Fig. 4.6), their keys are equal in the most significant bits. By removing part of the most significant bits the keys can be mapped to memory addresses within a previously allocated memory area. With this technique, a tree-node can be accessed very quickly compared to an indirect addressing scheme. Because of this method, called 'hashing', and the subdivision of a tree-node into eight child-nodes, this type of code is called 'hashed-oct-tree'.

Tree construction

After the domain decomposition, each process constructs its local tree. During this process a parent-node's key is obtained by shifting the child-node's key by the number of dimension bits in the direction of the least significant bit. Contrary, the child-node's keys can be constructed from the parent-node's key by attaching the binary representation of the numbers $0-7$. At each node, information about the present child-nodes are stored. If a tree-node contains only one particle this is attached to

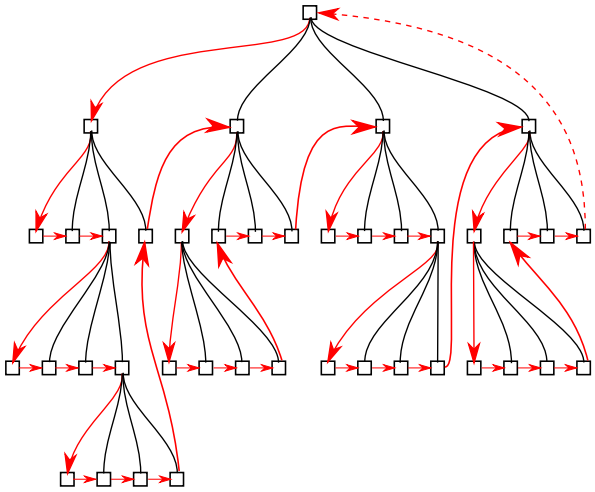

Figure 4.7: The tree from Fig. 4.2 where each node has a link to the "next" node. Following the red line each node is visited exactly once.

the node, else the node is subdivided into eight child-nodes (see Gibbon et al., 2010b). Additionally, at each node a link to the "next" node is stored to line up the nodes (see Fig. 4.7).

Then the multipole moments are computed for each local tree-node.

4.2.4 Tree-walk and force summation

When the preparations are done, for each local particle a list of pseudoparticles fulfilling the criterion (4.1) is built for the force-summation. Because these lists can contain several hundreds of pseudoparticles, storing the lists for all local particles can be very memory-consuming. To avoid this problem, the present version of PEPC creates interaction-lists for chunks of local particles at a time. After the force summation for each chunk the interaction-lists for the next chunk of particles is built and stored in the same memory area. Because the routine building the interaction-lists 'walks' through the tree from node to node, basically according to the order shown in Fig. 4.7, this routine is called tree-walk.

After computing the forces for all particles, the particles' velocities and positions are advanced by the integrator. Here a leapfrog integrator is used. This is a second order accurate integration scheme, which conserves momentum and energy accurate. Here the velocity is evaluated for a point in time between two coordinate evaluations.

$$x_{i+1} = x_i + u_{i+1/2} \cdot \Delta t$$
$$u_{i+1/2} = u_{i-1/2} + a(x_i) \cdot \Delta t,$$

where x is the coordinate, u the velocity, a the acceleration and the subscript denotes the time-step.

After the coordinates and velocities have been updated, the code proceeds with the next time-step. For a simplified work scheme of PEPC see Lst. 4.1.

```
1  for each time-step
2    do domain decomposition
3    build tree
4    compute multipole moments
5
6    for each chunk
7      find pseudoparticles for gravitational interaction
8      sum forces
9    end chunks
10
11   compute new velocities
12   compute new positions
13 end time-step
```

Listing 4.1: Simplified work scheme of PEPC

4.3 The tree-walk in detail

The newly implemented neighbour-search algorithm for SPH is based on PEPCs tree-walk. The neighbour search itself will be described later, but first the tree-walk will be described here in more detail.

Because PEPC is designed for distributed-memory computers, the particles needed for the force summation are not necessarily stored in the memory of the process doing the actual force summation. Thus, this information has to be retrieved from another process and copied into the local tree. In a typical run, a lot of multipole moments have to be imported by a process, so that this communication is very sensitive to network latency. To minimise the impact on the total runtime, PEPC uses a latency-hiding scheme (see Warren & Salmon, 1993) for searching and retrieving the information needed for the force evaluation. It is based on an alternating execution of communication code and the local search (see Lst. 4.2).

The interaction lists for long-range interactions can become quite long depending on the particle distribution and the parameter θ chosen for the acceptance criterion (Eq. 4.1). Therefore, the interaction-lists are not built for all particles at the same time. For a given number of particles, an array is filled with the indexes of the actual particles to be processed. The entries of this array named `pshort` (see Lst. 4.2 and Fig. 4.8) point to the actual particle coordinates, masses etc. For example the x-coordinate of the first particle in this list can now be accessed with `x(pshort(1))`.

At the beginning of a local walk a separate array called `plist` is filled with the indexes of the `pshort` array. Now, the x-coordinate of the first particle in this list can be accessed with `x(pshort(plist(1)))` (see Lst. 4.2 and Fig. 4.8).

For all particles `i` in `plist` the local walk walks through the tree searching for interaction partners

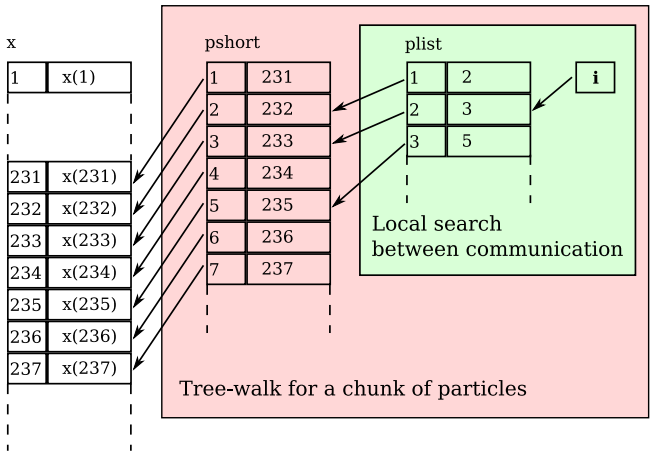

Figure 4.8: PEPCs tree-walk uses arrays containing indexes of other arrays for the book-keeping. Here the most important arrays are shown. `x` is an array containing for example the x-coordinates of the particles but could be another array containing velocities or masses as well. This array is global for the whole tree-code. The red rectangle denotes the whole tree-walk for one chunk of particles. The green rectangle represents the inner loop (see Lst. 4.2).

for the particle `pshort(plist(i))` starting at the root-node and and following the order formerly described (see Fig. 4.7). In the innermost loop for all particles in `plist` only one node is checked. When a tree-node fulfils the acceptance criterion (4.1), it is added to the interaction list of `plist(i)` otherwise in the next iteration the walk proceeds with the next node. When the walk reaches a tree-node where the acceptance criterion is not fulfilled and tries to access a child-node which is not locally present, this child-node is appended to a 'nodes-to-get'-list for later retrieval and the particle is removed from `plist`. This is repeated until `plist` is empty, which means that the walk cannot proceed without node-information from other processes.

This is acquired via the following steps:

1. All processes notify the other processes they need information from.
2. Buffers are created for receiving requests from other processes.
3. The requests are collected by the owner-process and sent to the requesting process.
4. Buffers are created for receiving requested data.
5. The data requested by the other processes are packed and sent.
6. The processes wait for incoming data.
7. When data are received in the buffers they are integrated into the local trees.

After the communication is finished for all particles in `pshort`, the walk status is tested. If the walk is not finished for a particle it is added to the new `plist`. Then the walk is resumed with `plist`, which will become shorter and shorter every iteration. It is important to note that the

interaction lists are associated with `pshort`, so `plist` only contains pointers to the unfinished particles.

When the walk is finished for all particles, `plist` does not contain any more particles. Then for all particles in `pshort`, pseudoparticles for the gravitational interaction are stored in lists. These lists are used by the force summation. All pseudoparticles with their multipole information needed for this force summation for particles of the actual chunk have already been imported to the local tree.

For a simplified work scheme of the tree-walk see Lst. 4.2.

```
1  IN: pshort   ! indices of actual particles in local arrays
2  IN: npshort = #pshort
3  maxactive = MPI_MAX(npshort)
4  plist = indices in pshort
5  nlist = #plist
6
7    DO WHILE (maxactive > 0)
8      DO WHILE (nlist>0 )
9        DO i=1,nlist
10         local_index = pshort( plist( i ) )
11
12           IF (mac ok) use this node
13           ELSE resolve node
14           END IF
15
16       END DO
17       plist = indices of not finished particles in pshort
18       nlist = #plist
19     END DO
20     get remote data
21     plist = indices of not finished particles in pshort
22     nlist = #plist
23     maxactive = MPI_MAX(nlist)
24   END DO
```

Listing 4.2: Simplified work scheme of PEPCs tree-walk.

5 PEPC with SPH

In this work, the existing hierarchical tree-code PEPC has been extended to include the treatment of a hydrodynamical component with a smoothed particle hydrodynamics algorithm. When combining a tree-code with SPH, it is natural to identify the particles used by the tree-code with the interpolation points used for SPH. In the tree-code, the particles already have coordinates x, y, z, velocities v_x, v_y, v_z and masses m. As SPH-particles contain as well intrinsic properties like density ρ and temperature T or energy u, additional information have to be introduced. Depending on how the code is designed, a smoothing-length h - fixed or individual - is required. Together with the force F or acceleration a affecting the particles, the temperature change has to be returned for later integration.

5.1 Smoothed Particle Hydrodynamics equations

Before implementing the equations derived in Chap. 3, some more adaptions are needed. To simulate gas-discs around stars, temperature is more important than internal energy, because it is used to compute pressure and, as output, can be compared to observations. Therefore, the temperature is used as one of the independent variables. The SPH-equations have to be adjusted for explicit dependence on temperature rather than other physical quantities.

From the equation of state for an ideal gas follows

$$P = \frac{\rho}{m_M} k_B T, \tag{5.1}$$

where P is the pressure, ρ is the density, m_M is the mass of one molecule of the gas, k_B is the Boltzmann constant and T the temperature. Combining this with (3.14), one obtains

$$\mathbf{F}_a = -m_a \sum_b m_b \left(\frac{k_B T_b}{m_M \rho_b} + \frac{k_B T_a}{m_M \rho_a} + \Pi_{ab} \right) \nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{5.2}$$

With the energy per particle with Z degrees of freedom of $u = \frac{Z}{2} k_B T$ and the relation between heat capacity ratio γ and the degrees of freedom of $\gamma = \frac{Z+2}{Z}$, one obtains the relation between temperature T and energy per particle u as

$$k_B T = (\gamma - 1) u.$$

Together with (3.15) and the transformation above, one obtains for the temperature change

$$\frac{dT}{dt} = -(\gamma-1)\frac{1}{2}\sum_b m_b \left(\frac{T_a}{m_M \rho_a} + \frac{T_b}{m_M \rho_b} + \frac{\Pi_{ab}}{k_B}\right)(\mathbf{v}_b - \mathbf{v}_a)\nabla W(\mathbf{r}_a - \mathbf{r}_b, h). \tag{5.3}$$

Equation 3.6 for the density can be implemented as derived earlier.

5.2 Implementation details

The SPH-routines are implemented as a module in PEPC which can be included easily into the rest of the code. The entire SPH-part is basically started by just one function call inside PEPC. However, some changes to parts of the existing code were still necessary: Particle properties like density, temperature and temperature change have to be included. Also several new physical constants have be introduced.

The additional program components were implemented as shown in Lst. 5.1 (for comparison see Lst. 4.1). The gravitational force summation results in an array with the forces for all particles. In the SPH-force-summation, the thermal forces are just added to the gravitational forces in that array. Additionally, the temperature changes are written to another array. These arrays are passed back to the integrators for the particle coordinates and the temperature.

Since the equations 5.2 and 5.3 need the density ρ, it is computed first, which however needs the neighbours of the particles to be determined first. This results in the necessary sequence:

1. search neighbours
2. compute density
3. compute force

The temperature change is computed together with the force, as properties already calculated for the force can be reused. Because PEPC is a distributed-memory program, the computed density of a particle has to be sent to all processes which have requested this particle during the neighbour search. This has necessarily to be done between the density computation and the force computation which results in the flow seen in Lst. 5.1.

The summations according to the equations 3.6, 5.2 and 5.3 for a fixed number of neighbours has a complexity of $\mathcal{O}(N \cdot N_{nn})$. The most expensive part of the SPH-method is the neighbour-search. In the simulation code developed in this work, a neighbour-search exploiting the already present oct-tree is used with a complexity of $\mathcal{O}(N \log N)$.

```
1  for each time-step
2    do domain decomposition
3    build tree
4    compute multipole moments
5
6    for each chunk
7      find pseudoparticles for gravitational interaction
8      sum forces
9    end chunks
10   BEGIN SPH
11
12     for each chunk
13       search neighbours
14       (validate neighbours)
15       compute density
16     end chunks
17
18     update remote density
19
20     for each chunk
21       compute sph force
22     end chunks
23
24   END SPH
25   compute new velocities
26   compute new positions
27   compute new temperature
28 end time-step
```

Listing 5.1: Simplified work scheme of PEPC with SPH

5.3 The neighbour search

The implemented neighbour-search algorithm follows the idea described by Warren & Salmon (1995). It corresponds basically to a modified version of the tree-walk as described in Sec. 4.3.

```
1  search_neighbours_of_particle_i(r) {
2
3    walk through tree from root to leaves
4      1) particles within r put on next neighbour list
5      2) ignore nodes/particles outside r
6      3) for nodes with overlap with r
7         if children locally present, resolve
8         else get children from remote process
9    end
10 }
```

Listing 5.2: The first version of the neighbour search algorithm. It is just capable of finding neighbour particles within a fixed radius around the particle i.

The implementation was done in several steps. The first step is to find all particles within a given radius around a particle (see Lst. 5.2).

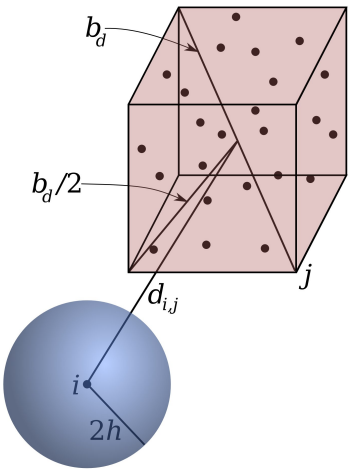

Figure 5.1: Sketch for the neighbour acceptance criterion. The node j (here illustrated as red cube) can contain neighbour particles for the particle i inside the blue sphere, if the sphere with radius $b_d/2$ around the centre of the cube and the blue sphere overlap. This is the case, when (5.4) is fulfilled.

For every particle i, the algorithm walks through the tree from the root down to the leaves as described before. The acceptance criterion for a node is changed from the original node-size-distance-ratio (Eq. 4.1) to a criterion taking the distance and both smoothing lengths into account. Here only particles within the radius $2h$, which is here equal for all particles, are accepted as neighbours, because the kernel vanishes outside (see Sec. 3.2.2). This means that a node j with body diagonal b_d can contain potential neighbour particles for particle i, if

$$d_{i,j} \leq 2h + b_d/2, \tag{5.4}$$

where $d_{i,j}$ is the distance between particle i and geometrical centre of node j (see Fig. 5.1). In this

case, the child-nodes have to be checked with the same criterion. Like in the normal tree-walk, the non-local data has to be requested from other processes. If a node is a leaf and the acceptance criterion is positive, the associated particle is added to the neighbour list of particle i.

The validity of the implementation was tested as described in Sec. 5.3.1. Figure 5.2 shows a sample particle distribution where the neighbours of one particle were removed for better contrast.

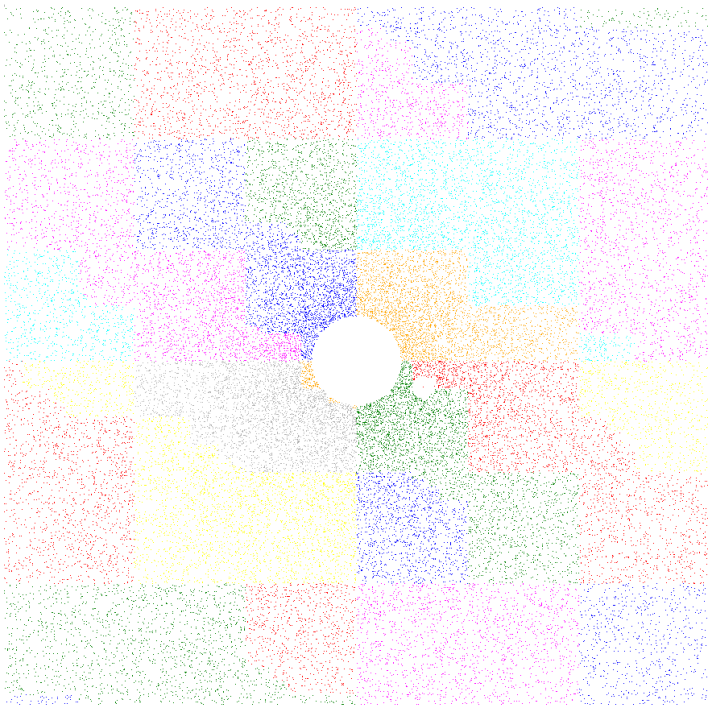

Figure 5.2: Visualisation of the results of the next neighbour search with 100 000 particles distributed in a two dimensional disc with a r^{-2} density run with 32 processes. This is a zoom on the inner part. The different colours represent the process domains. In the big white circle in the centre are no particles to avoid to high particle densities caused by the r^{-2} density distribution. In the other white circle the next neighbours within the radius $2h$ around the particle in the centre of this circle were removed to have a better contrast.

```
1  while there are particles with less than N_nn found next neighbours
2    search_neighbours_of_particle_i(r_i)
3
4    if found next neighbours < N_nn
5      increase r_i for this particle
6      put particle on list to search neighbours again
7    end
8  end
9
10 for all particles
11   while n_found > N_nn
12     move furthest particle to the end of the list
13     shorten list by one
14   end
15
16   move furthest particle to the end of the list
17   new search radius r_i is 1.1 * distance to this particle
18 end
```

Listing 5.3: Modified neighbour search to find exactly N_{nn} neighbours.

Next the search was refined to obtain *exactly* N_{nn} neighbours for a particle i. This is done by first searching neighbours within a sphere with a given radius r_i around particle i. For the first time-step, these radii are set to an initial value specified in the configuration file. For all other time-steps, the value determined in the previous time-step is used. After all particles in this sphere are found, they are counted to check whether at least N_{nn} neighbours are found. If not, the search radius is increased by 10% for this particle and the search is repeated (see Lst. 5.3).

When the number of located neighbours N_f fulfils $N_f \geq N_{nn}$, the furthest neighbour is moved to the end of the list. While $N_f > N_{nn}$, the list is then shortened by one and the next furthest particle moved to the end of the list, until exactly N_{nn} neighbours are left. When this is finished the furthest among the N_{nn} neighbours is at the end of the list. The separation between this furthest neighbour and particle i is used as the new individual search radius for particle i during the next time-step.

This version is basically one more loop around the first version, that is executed while there are still particles with less than N_{nn} found neighbours.

To test the SPH-code, it was necessary to make simulations of a periodic medium possible. Switches in the configuration-file allow one, two or three dimensional periodicity or open boundaries[1] to be selected. The periodic boundaries have been implemented by modifying the neighbour search and the integrator. When the particles are moved, it is tested whether the particles are still inside the

[1]The periodicities of all three dimensions can be chosen individually. However, because in case of reduced dimensionality first the x-, then the y- and then the z-dimension is used, the periodicity has to be enabled according to the choice of dimensionality.

simulation-box. Therefore, the minimum x-, y- and z-coordinate and the x-, y- and z-box-length have to be specified in the configuration-file. When a particles coordinate is smaller (bigger) than the corresponding minimum (maximum) coordinate the particle is shifted by the box-length so that it is inside the box again.

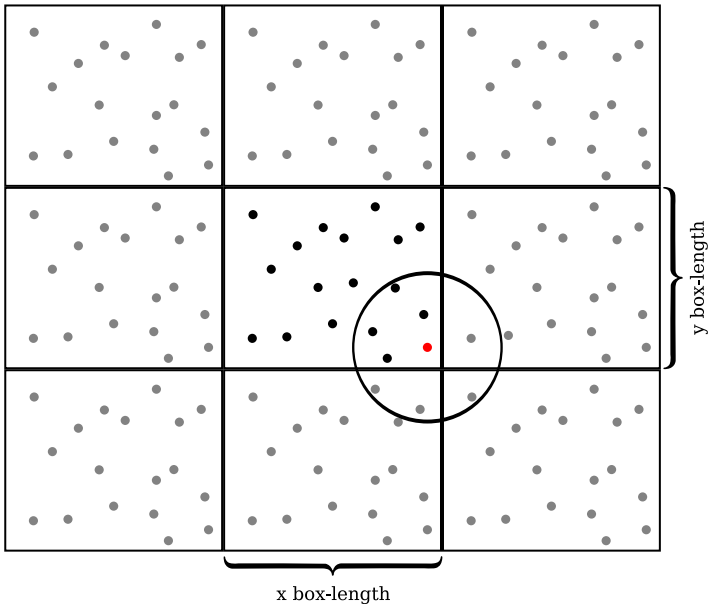

Figure 5.3: To find all neighbours of the red particle in the circle in case of 2D-periodicity also the virtually shifted simulation boxes (gray particles) have to be tested.

In the case of a periodic boundary problem, the neighbour search has to look for neighbouring particles also in simulation boxes virtually shifted by the box-lengths (see Fig. 5.3). In 1D 3 boxes, in 2D 9 boxes, and in 3D 27 boxes respectively have to be searched for neighbours. Therefore, according to the periodicity, shift-vectors for the virtual simulation boxes are constructed[2]. Then for all shift-vectors the overlap of search-sphere and particle-coordinate shifted by the shift-vector is tested. In case a neighbour is found, the neighbours node-number and the shift-vector are stored in a list. The particles are again virtually shifted by the shift-vectors when density and forces are computed.

The search in all virtual boxes is realised as an additional loop over all copies of the simulation box. Searching for neighbours in up to 27 copies of the simulation box needs up to 27 times the normal run-time. This is very slow and could be solved way better, but since it is just for tests and not for real simulations this is not necessary here.

A visualisation of the neighbours of a particle for a 3D particle distribution with 3D-periodicity can be found in Fig. 5.4.

The final neighbour search algorithm is implemented as shown in Lst. 5.4. The loop over the periodic copies is the innermost loop and done for every single comparison. Here lies potential for improvement. When a comparison is negative for one periodic copy of a node and positive for another one, the test for the child-nodes is only necessary for the children of that copy where the test

[2]The first shift-vector is (0,0,0) for the real simulation-box, another one could be (- x-box-length, 0, z-box-length).

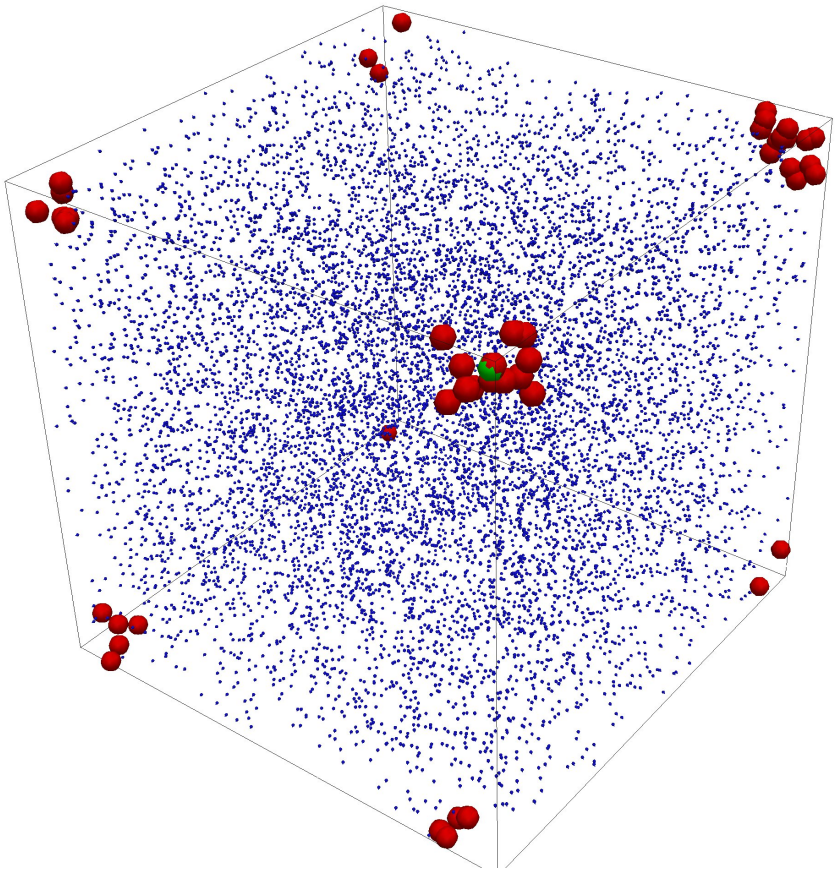

Figure 5.4: Neighbours (red) of a particle (green) in a three dimensional periodic simulation-volume. Because the green particle is located in one corner of the simulation box and the periodicity in all three dimensions is activated, the neighbours lie in all corners.

was positive. This will be improved in the next version.

```
1  IN: pshort   ! indices of actual particles in local arrays
2  IN: npshort = #pshort
3
4  not_enough_nn = indices in pshort, n_not_enough_nn = #not_enough_nn
5  plist = indices in not_enough_nn, nlist = #plist
6  max_n_not_enough_nn = MPI_MAX(n_not_enough_nn)
7  DO WHILE (max_n_not_enough_nn > 0)
8    maxactive = MPI_MAX( n_not_enough_nn )
9
10   DO WHILE (maxactive > 0)
11     DO WHILE (nlist>0 )
12       DO i=1,nlist
13
14         local_index = pshort( not_enough_nn( plist( i ) ) )
15         DO periodic_shift = 1, num_periodic_shifts
16
17           IF (mac ok)
18             IF ( node is particle) use this particle
19             ELSE resolve node
20             END IF
21           ELSE ignore this node
22           END IF
23
24         END DO    ! periodic
25
26       END DO    ! do nlist
27       plist = indices of not finished particles in not_enough_nn
28       nlist = #plist
29     END DO    ! while nlist
30     get remote data
31     plist = indices of not finished particles in not_enough_nn
32     nlist = #plist
33     maxactive = MPI_MAX(nlist)
34   END DO    ! maxactive
35
36   not_enough_nn = indices of particles with less than nn neighbours
       in pshort
37   n_not_enough_nn = # not_enough_nn
38   plist = indices in not_enough_nn, nlist = #plist
39   max_n_not_enough_nn = MPI_MAX(n_not_enough_nn)
40 END DO    ! not enough
```

Listing 5.4: The final neighbour search algorithm with periodicity.

The previously described reiteration of the neighbour search for particles with $N_f < N_{nn}$ is realised similar to the loops over a subset of the particle chunks in the original tree-walk. One more array containing indexes of another array is introduced called `not_enough_nn` with its length `n_not_enough_nn`. The entries of this array point to the entries of `pshort` (see Fig. 5.5). All particles with `pshort(not_enough_nn(i))`, with `i` between `1` and `n_not_enough_nn` fulfil $N_f < N_{nn}$. For example, during the inner loop, the coordinates of the first particle can be accessed with `x(pshort(not_enough_nn(plist(1))))`.

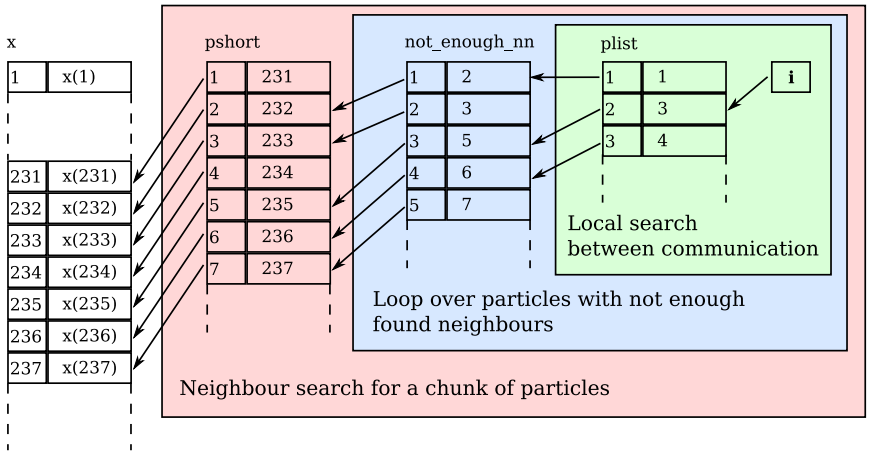

Figure 5.5: Also the new implemented neighbour search algorithm uses arrays containing indexes of other arrays for the book-keeping. Here the most important arrays are shown. `x` is an array containing for example the x-coordinates of the particles but could be another array containing velocities or masses as well. This array is global for the whole tree-code. The red rectangle denotes the whole neighbour search for one chunk of particles. The blue rectangle shows the reiteration of particles with $N_f < N_{nn}$. The green rectangle represents the inner loop (see Lst. 5.4).

As shown in Lst. 5.4, the array `not_enough_nn` is filled with all particles in line 4. In the 'maxactive'-loop from line 10 to line 34, the search is performed with fixed search radii for all particles in `not_enough_nn`. In line 36 `not_enough_nn` is filled again with the particles fulfilling $N_f < N_{nn}$. For those particles the search radius is increased as described before. Then the search is started again for these particles.

5.3.1 Validation of the neighbour search

To check the correctness of the neighbour lists for all particles, a validation subroutine was implemented. This routine computes all $N-1$ distances between one particle and all others and stores them in a list. This list is then sorted and the closest N_{nn} neighbours are compared to the results from the tree-based neighbour search.

Storing the distances from all particles to all other particles needs $\mathcal{O}\left(N^2\right)$ memory. For huge particles numbers this is not possible. Therefore, the validation is performed in chunks similar to the neighbour search. All processes send their particles to all other processes one after the other. The processes compute the distances for all particles in the actual chunk to the received remote particles.

The distances are sorted and only the closest N_{nn} neighbours are stored. When the distances to the particles from the next process are computed, these N_{nn} closest neighbours are appended to the list before sorting it. In the end the closest N_{nn} neighbours among all particles are known.

The result from this routine is compared to the result from the tree-based neighbour search and the number of differences, if any, is written out to a log-file. With this routine, the validation of periodic neighbour lists is also possible.

Final tests of the periodic neighbour search with random particle distributions with up to 10^6 particles and all combinations of dimensionalities and periodicities worked without errors.

5.3.2 Symmetric neighbour search

As already noted above (see Sec. 3.2.3), the neighbours have to be chosen according to the symmetric neighbour criterion (3.17) for a better momentum conservation. Since the density computation is done with the neighbours based on simple neighbour criterion (3.16), this makes another neighbour search necessary just before the force computation. To implement the symmetric criterion, the tree-construction has to be modified. To allow the comparison with

$$max(h_i, h_j),$$

the maximum h beneath each node has to be stored as a node-property - a refinement beyond the scope of this work which will be included in future.

5.3.3 Scaling

An analytical estimate

Assuming that the density is constant within the whole simulation volume, one can estimate the number of particles to be fetched by one process (Breslau, 2010). A total number of particles N in the volume V result in a mean-density of $\rho = N/V$. If simulated on p processes, the average volume of one domain is $V_p = V/p$. Assuming that the domain is spherical, its radius can be expressed as

$$R_{\text{domain}} = \left(\frac{3\pi V_p}{4}\right)^{\frac{1}{3}}.$$

With the mean-density, the radius of the sphere containing all N_{nn} next neighbours of a particle can be written as

$$r_{\text{search}} = \left(\frac{3N_{nn}}{4\pi\rho}\right)^{\frac{1}{3}}.$$

As shown in Fig. 5.6, the search spheres of all particles close to the domain border form a shell around the domain, in which next neighbours of particles from the domain can be found. All these

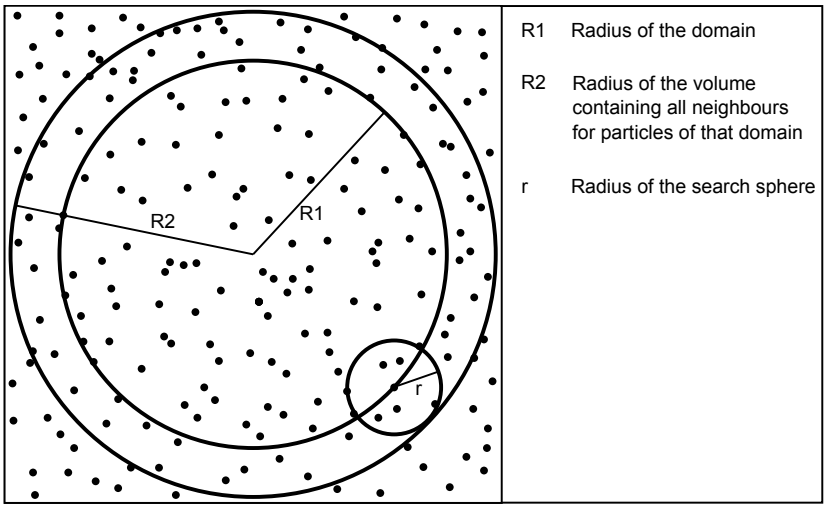

Figure 5.6: Sketch for the estimation. The inner big circle with the radius $R1$ represents one process domain. If there are many particles near the domain border, the search spheres with radius r of these particles overlap and fill the area between the two bigger circles. All particles in this area have to be fetched from other processes.

particles have to be fetched during the next neighbour search and stored in the local memory. Their number can be calculated as

$$\begin{aligned} N_{\text{fetch}} &= \frac{4}{3}\pi\rho\left[(R+r)^3 - R^3\right] \\ &= \left(27\frac{N_{nn}N^2}{p^2} + 27\frac{N_{nn}^2 N}{p} + N_{nn}^3\right)^{\frac{1}{3}}. \end{aligned}$$

It is easy to see that this leads to the asymptotic complexities

$$\mathcal{O}(N_{nn}), \quad \mathcal{O}(N^{2/3}), \quad \mathcal{O}(p^{-2/3}), \tag{5.5}$$

for large N_{nn}, N and p respectively.

In case of a fixed number of particles per process N_p, one finds

$$N_{\text{fetch}} = \left(27N_{nn}N_p^2 + 27N_{nn}^2N_p + N_{nn}^3\right)^{\frac{1}{3}},$$

which leads to the complexities

$$\mathcal{O}(N_{nn}), \quad \mathcal{O}(N_p^{2/3}). \tag{5.6}$$

However, this is only for the communication effort and the memory space needed locally to store the fetched particles. Additionally requested memory can lead to problems when simulating with too few particles per process, because the size of the locally allocated memory is initially calculated depending on the number of local particles. As shown in Tab. 5.1, the fraction of additional memory needed, increases with decreasing particles per process. This can lead to out-of-memory problems

even though the memory is almost unused.

N_{nn}	N_p	N_{fetch}	$N_{\text{fetch}}[\%]$
50	10000	6000	61
50	50000	17000	33
50	200000	40000	20

Table 5.1: Estimated number of particles to fetch from remote processes depending on the number of particles per process.

Benchmarks

For benchmarking the new algorithm, several sets of time measurements have been performed each with one configuration parameter varying. In each case the time needed by the neighbour search algorithm for the local walk ('nn walk'), the time to fetch data from other processes ('nn fetch') and the time needed by the rest of the program ('rest') was measured.

The measurements were performed without periodicity because the periodic search is known to be slow and is just for test scenarios and not for real simulations.

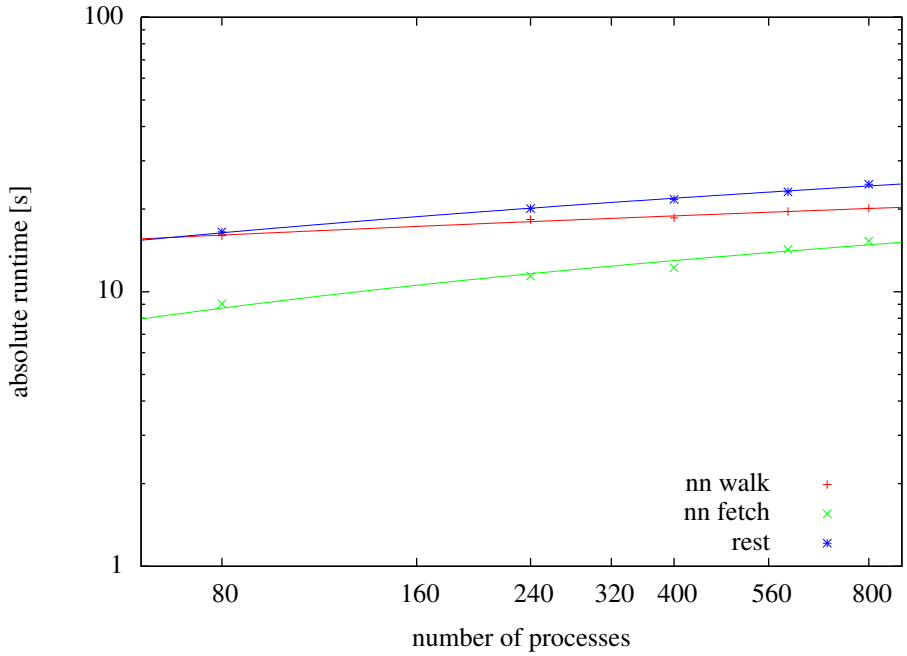

Figure 5.7: Weak scaling on Juropa using 150000 particles per process and 50 next neighbours to find. The data were fitted with functions of the form $f(x) = a\log(bx) + c$, where c is a constant overhead, b is an overhead per particle and a is the cost-factor for each particle.

- For the weak scaling, the number of particles per process was fixed to 150000 and the number of neighbours required set to 50. The number of processes was varied from 80 to 800, which

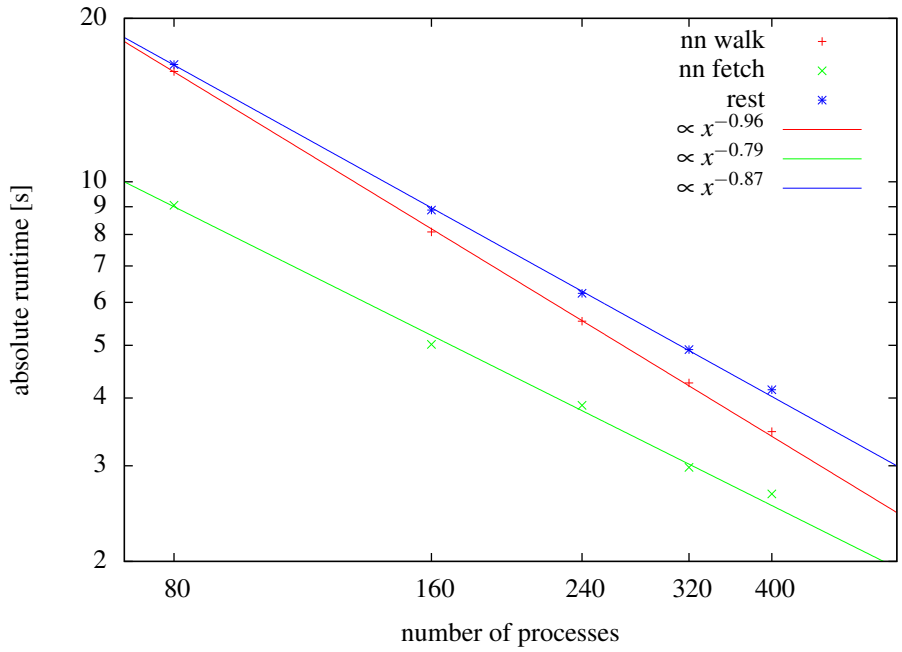

Figure 5.8: Strong scaling on Juropa using 12 M particles and 50 next neighbours to find.

results in $1.2 \cdot 10^7$ to $1.2 \cdot 10^8$ particles in total. For the log-log plot of the absolute run-time see Fig. 5.7. Due to the theoretical $\mathcal{O}(N \log N)$ scaling of the tree-algorithm, the weak scaling, where $N = N_p \cdot p$, should be close to $\mathcal{O}(\log(N_p p))$. Therefore the data were fitted with functions of the form $f(x) = a \log(bx) + c$. Fits with a function of the form $f(x) \propto x^a$, according to what is expected from the analytical estimate (Eq. 5.6) resulted in values for a below 0.25. That means that the time for the communication is here not dominated by the number of fetched particles.

- For the strong scaling, the total number of particles was fixed to $1.2 \cdot 10^7$ and the number of neighbours set to 50. The number of processes was varied from 80 to 400, which results in 30000 to 150000 particles per process. For the log-log plot of the absolute run-time see Fig. 5.8. Ideal strong scaling means that if the number of processes is multiplied by a factor a the run-time is divided by a, which results in a p^{-1} dependence of the run-time, where p is the number of processes. As can be seen in the figure, the local next neighbour search ('nn walk') scales almost ideal with a $p^{-0.96}$ dependence. The 'nn fetch', which is the time for the next neighbour communication, scales like $p^{-0.79}$. This is close to the estimated $p^{-2/3}$ (Eq. 5.5).

- For the N scaling, the number of processes was fixed to 240 and the number of neighbours set to 50. The total number of particles was varied from $6 \cdot 10^6$ to $3.6 \cdot 10^7$, which results in 25000 to 150000 particles per process. For the log-log plot of the absolute run-time see Fig. 5.9. The fitted functions show that all three measured program parts scale like $\mathcal{O}(\log(N_p p))$, which is in excellent agreement with the expectations.

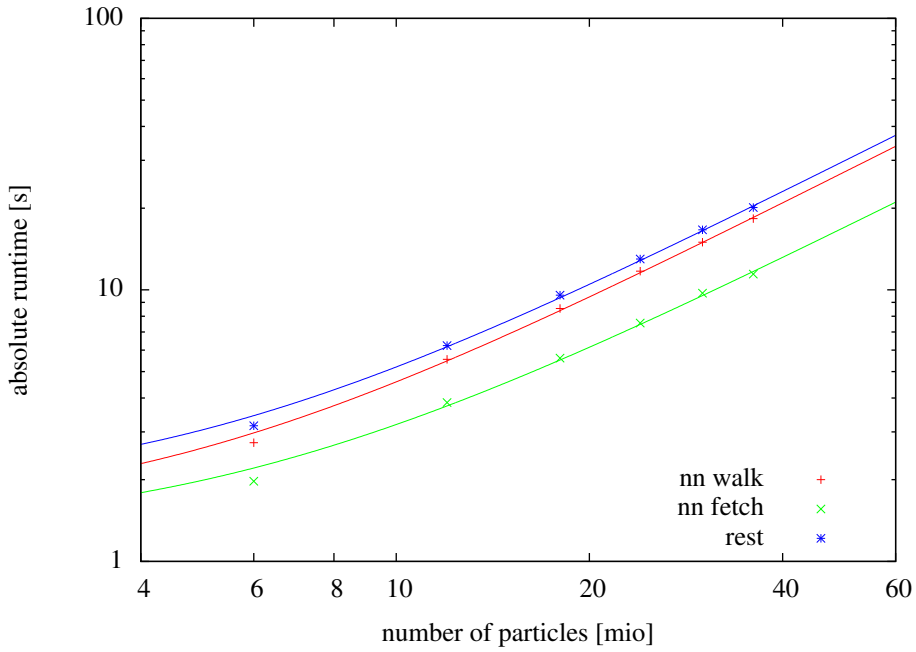

Figure 5.9: N scaling on Juropa using 240 processes and 50 next neighbours to find. The data were fitted with functions of the form $f(x) = ax\log(x) + b$, where b is a constant overhead and a is the cost-factor for each particle.

- For the N_{nn} scaling, the number of particles per process was fixed to 50 000 and the number of processes set to 80. The number of next neighbours to find was varied from 25 to 150. For the log-log plot of the absolute run-time see Fig. 5.10. According to the expected linear dependence on N_{nn} (Eq. 5.6 and Eq. 5.5) the data were fitted with functions of the form $f(x) = ax^b$. There is a good agreement for the 'nn fetch'-time. The walk is even better. The run-time increase of the rest of the program is probably due to a fuller hash-table.

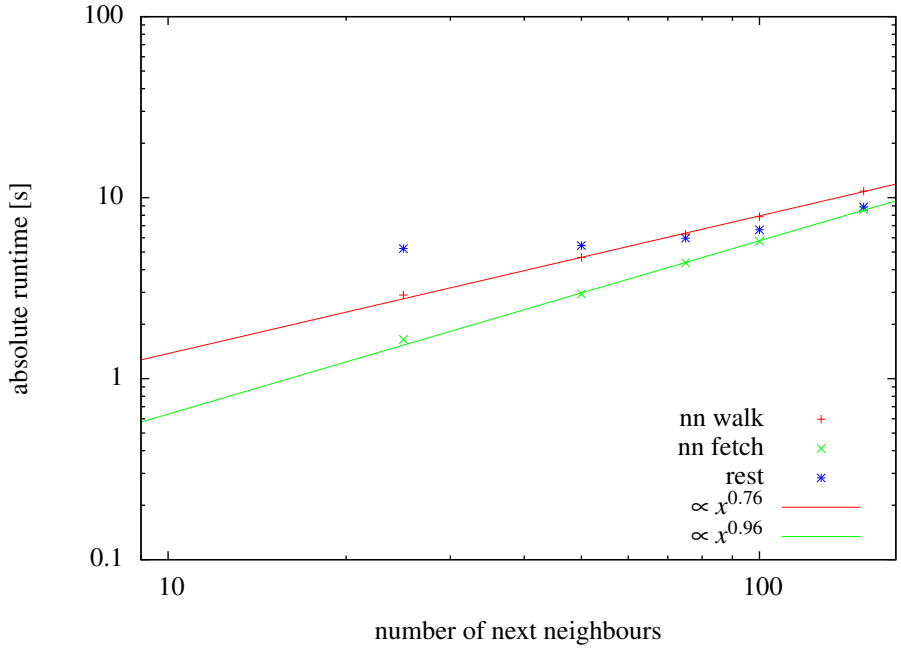

Figure 5.10: N_{nn} scaling on Juropa using 50 000 particles per process and 80 processes.

6 Testing the new code

Several tests were performed to verify the correctness of the code. Among them are one-dimensional sound-waves propagating in a periodic simulation box. After these tests succeeded, further features like the artificial viscosity were implemented. Its validity was tested with one-dimensional shock problems. Finally, a test involving gravity and pressure was performed to show that both components are working correctly together.

Generating start configurations

Before starting the test series, one has to solve the problem of generating start conditions. There are in principle two methods to setup a density distribution with SPH. The first one is to distribute particles with different masses equally-spaced in the simulation box. This is straight forward but destroys the intrinsic resolution adaptiveness. The other method is to use equal-mass particles and distribute them with variable distances. This method is more complex but the resolution adaptiveness is preserved. Therefore, this method is preferred.

For one-dimensional setups the density $\rho(x)$ can be integrated over the whole length L of the simulation box to obtain the mass M in the box

$$M = \int_L \rho(x)dx.$$

For a given number of particles N in the simulation box the mass of an individual particle m is given by $m = M/N$. To place the first particles in the simulation box, the density-function $\rho(x)$ is integrated again with the upper limit x_1 fulfilling

$$\int_0^{x_1} \rho(x)dx = m.$$

The particle is then placed in the middle between 0 and x_1. For the other particles i the integration has to fulfil

$$\int_0^{x_i} \rho(x)dx = i \cdot m.$$

The particle i is placed in the middle between x_i and x_{i-1}. With this method start-conditions for any

density-distribution in one dimension $\rho(x)$ can be obtained.

6.1 1D sound wave

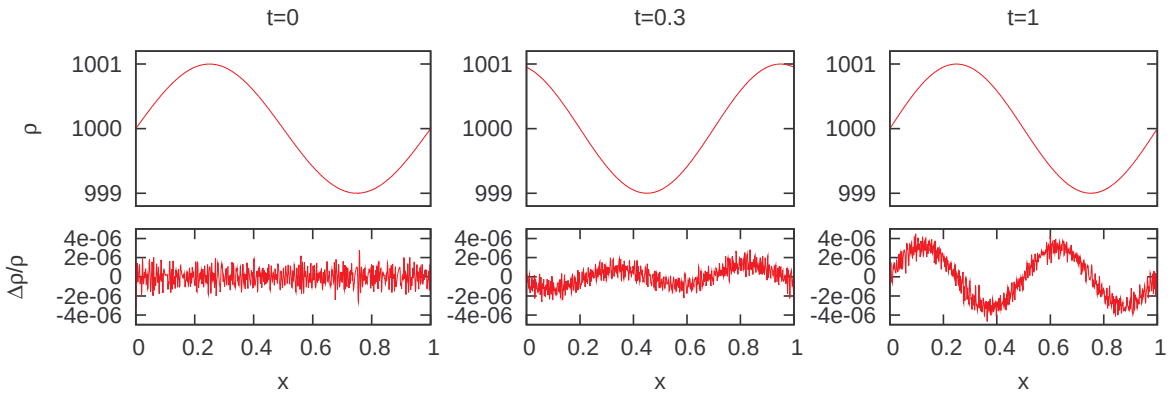

Figure 6.1: 1D sound-wave propagating in a periodic simulation box of length $L = 1$. The plots show the simulated density and the deviation from the analytical density for $t = 0$, $t = 0.3$ and $t = 1$.

In fluids, sound waves are small density disturbances that obey the (here one-dimensional) wave-equation

$$\frac{\partial^2 \rho}{\partial t^2} = c^2 \frac{\partial^2 \rho}{\partial x^2}, \tag{6.1}$$

where c is the sound-speed. Besides the wave equation, sound waves have to satisfy the one-dimensional Euler equations, which are given by

$$\frac{\partial \rho}{\partial t} = -\frac{\partial(\rho v)}{\partial x},$$
$$\frac{\partial v}{\partial t} = -\frac{1}{\rho}\frac{\partial P}{\partial x}$$

with the velocity v of the medium and the pressure P. For small disturbances, $\rho_1 << \rho_0$, and with the equation of state for an ideal gas $P = k_B T \rho$ (here the mass of the molecules m_M is included in k_B) they can be transformed to

$$\frac{\partial \rho}{\partial t} = -\rho_0 \frac{\partial v}{\partial x},$$
$$\frac{\partial v}{\partial t} = -\frac{k_B T}{\rho_0}\frac{\partial \rho}{\partial x}.$$

It can be easily shown that

$$\rho(x,t) = \rho_0 + \rho_1 \sin\left(2\pi\frac{x}{\lambda} + \omega t\right) \tag{6.2}$$

$$v(x,t) = -c\frac{\rho_1}{\rho_0} \sin\left(2\pi\frac{x}{\lambda} + \omega t\right) \tag{6.3}$$

solve these equations and the wave-equation (6.1), with $c = \omega\lambda/2\pi$ and $c^2 = k_B T$. Here ω is the angular frequency and λ the wave-length of the wave.

For the time $t = 0$, the start parameters for a simulation are

$$\rho(x) = \rho_0 + \rho_1 \sin\left(2\pi\frac{x}{\lambda}\right), \tag{6.4}$$

$$v(x) = -c\frac{\rho_1}{\rho_0} \sin\left(2\pi\frac{x}{\lambda}\right). \tag{6.5}$$

The box-length L has to be chosen as a multiple of the wave-length λ as $L = k\lambda$, with the wave-number k, because of the periodicity of the simulation box.

A one-dimensional wave was set up with wave-number $k = 1$. For the here performed tests, the simulation box of length $L = 1$ contained $N = 1000$ particles (see Fig. 6.1, left plot). The sound-speed was chosen as $c = 1$. This setup was advanced with a time-step size of $\Delta t = 0.0001$ for a total time $t = 1$. Figure 6.1 shows a comparison of the results at $t = 0.3$ and $t = 1$ together with the analytical solutions (see Eq. 6.2). The upper plots show the densities obtained from the simulation. The lower plots show the relative deviation from the analytical density $(\rho_{sim} - \rho_{\text{analytic}})/\rho_{\text{sim}}$. The relative error increases with time, but after 10000 time-steps it is still below $4 \cdot 10^{-6}$ which is satisfying. Further tests could measure the dependence of the error of the number of particles and the size of the time-step.

6.2 1D sound wave in 2D volume

The sound wave test was also performed in a two-dimensional periodic simulation volume. Therefore, the particles distributed along the x-axis from the previous test were replicated in y-direction to fill the x-y-plane. The start velocities were obtained as in the one-dimensional case. Figure 6.2 shows the results. The wave is only stable until $t \approx 0.32$, corresponding to time-step 32 for the time-step size $\Delta t = 0.01$ in the case visualised here. With smaller time-steps the result is almost the same. This problem is still not fully understood, but the wave collapses probably because the whole setup is very sensitive to perturbations. The particle positions deviate from a regular lattice with spacing d only by a factor of $0.001d$.

However, the code presented here is not designed for solving exact problems. The stability of the wave until $t = 0.32$ can be interpreted as a good result.

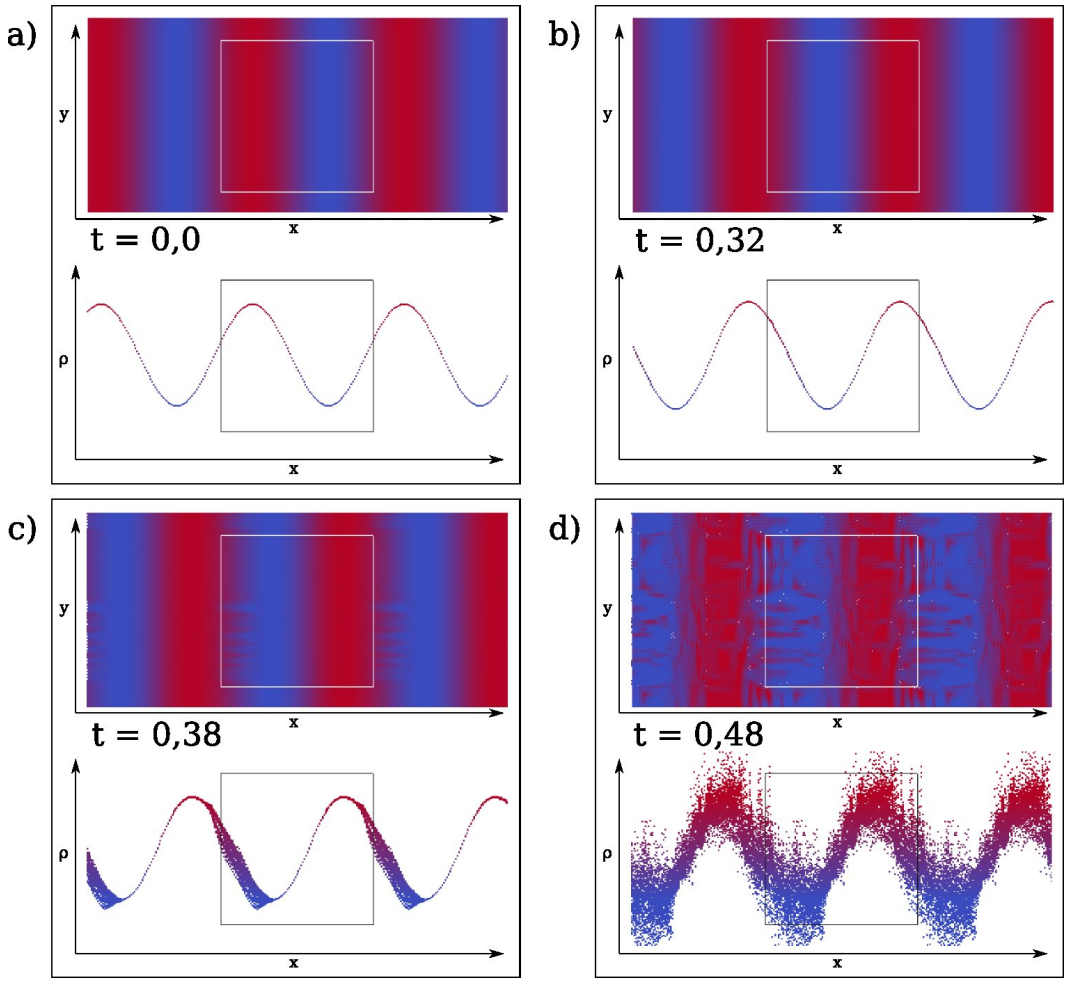

Figure 6.2: One-dimensional sound wave propagating in a two-dimensional periodic simulation volume. The upper plots show the view on the x-y-plane. The colour denote the density, where blue is low and red is high. The lower plots show the density versus x-coordinate, colours as in the upper plots. The squares delimit the real simulation volume, the rest are shifted copies to show the periodicity. The graphics show the initial conditions (a), for $t = 0.32$ (b), $t = 0.38$ (c) $t = 0.48$ (d). The wave propagates in the negative x-direction.

6.3 1D shock-tubes

In the next step, it should be tested in how far the code is capable to resolve shocks. As already mentioned, shocks occur, when the fluid properties, like temperature, density of pressure, change by huge amounts. This results in large accelerations. The tests presented here were introduced by Sod (1978). Toro (1997) described them in detail and presented numerical solutions obtained with several different solvers. Springel (2010) and Hubber et al. (2011) performed similar tests with their SPH-codes. In Springel (2010) three different problems were investigated. In Tab. 6.1 the initial conditions are shown. All three test problems are one dimensional, hence shock tubes, with a discontinuity in density, pressure or velocity. Figure 6.3 shows the results from that paper.

Here only the problems 1 and 3 are used. Problem 1 is investigated with the two alternative setup methods described in Sec. 6. Figure 6.4 shows the results. They can be compared to Fig. 6.3. Results

Test	ρ_L	v_L	P_L	ρ_R	v_R	P_R
Problem 1	1.0	0.0	1.0	0.125	0.0	0.1
Problem 2	1.0	−2.0	0.4	1.0	2.0	0.4
Problem 3	1.0	0.0	1000	1.0	0.0	0.01

Table 6.1: Initial conditions for shock tests from Springel (2010).

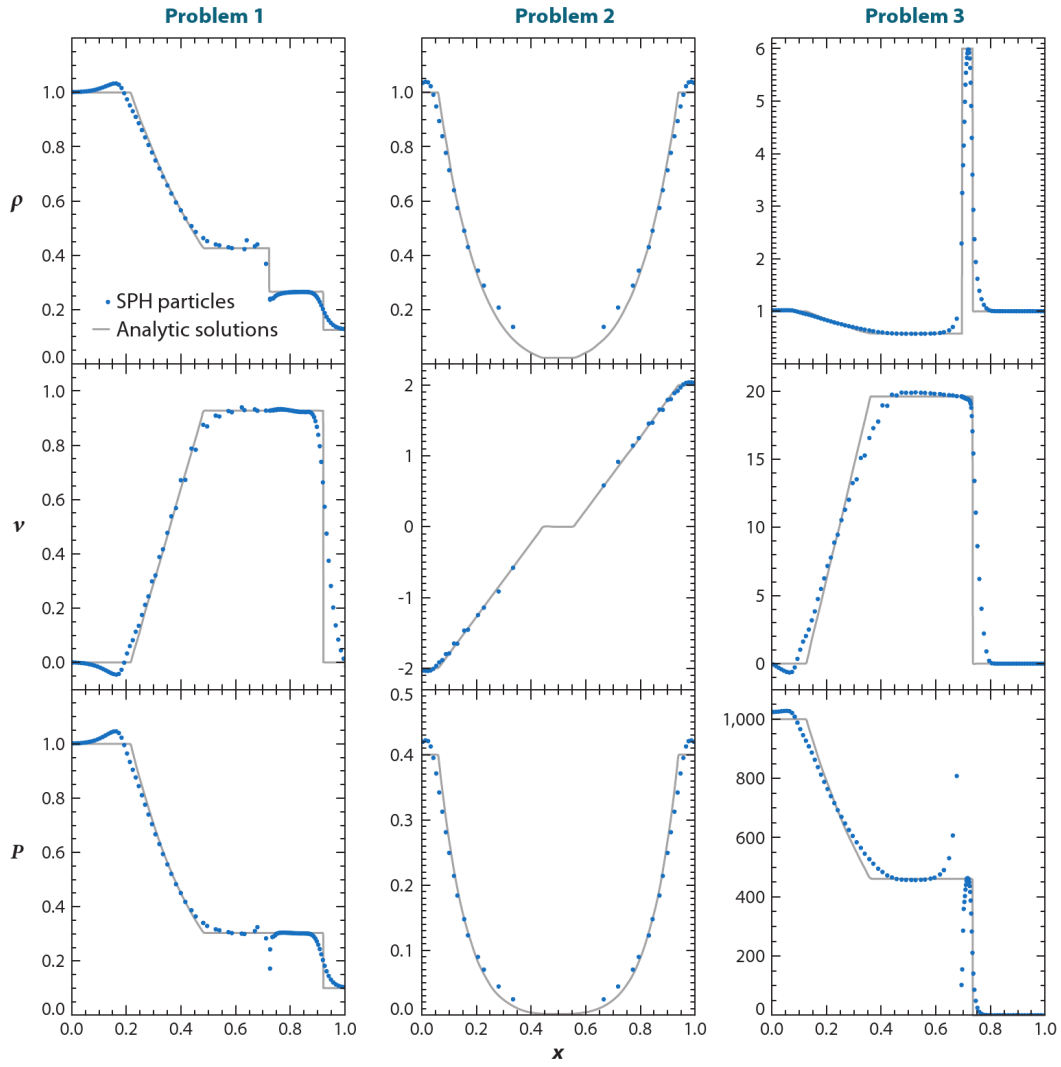

Figure 6.3: Results of the 1D shock tests with SPH from Springel (2010) for comparison.

for similar tests can for example be found in Hubber et al. (2011) or Merlin et al. (2010).

- Test 1a (see Fig. 6.4, left) corresponds to Problem 1 from Tab. 6.1, set up with equal mass particles distributed according to above described method. The initially higher pressure and density in the left half of the box causes a shock wave propagating to the right and a rarefaction wave propagating to the left. In the rarefaction front the particles are accelerated to the right, what can be seen in the velocity plot. Due to the simple neighbour search which does not produce

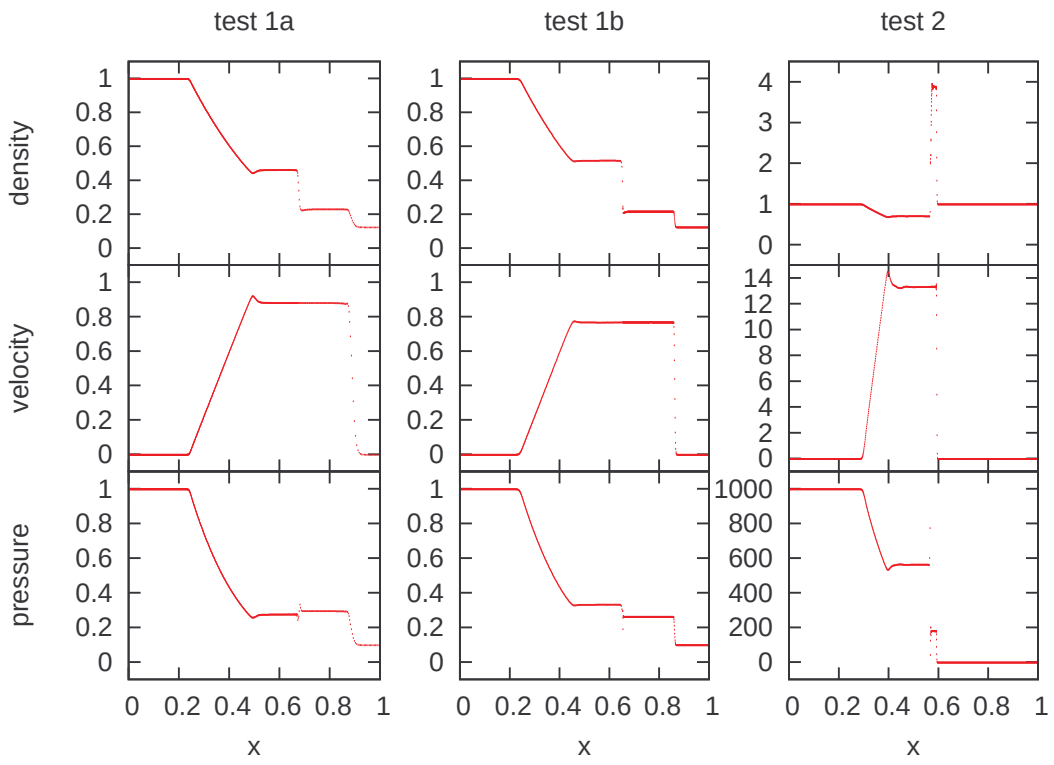

Figure 6.4: Results for all three performed shock tests. Test 1a and test 1b correspond to Problem 1 in Tab. 6.1. Test 1a is set up with equal mass particles and test 1b with equidistant distributed particles. Test 2 corresponds to Problem 3 set up with equidistant distributed particles. (Explanation see text.)

neighbour lists for a symmetric force computation the momentum is not fully conserved. This is probably the cause of the difference of the results in Fig. 6.4 to the results by Springel (see Fig. 6.3).

- Test 1b (see Fig. 6.4 middle) corresponds to Problem 1 from Tab. 6.1, set up with equidistant distributed non-equal mass particles. The result is basically the same as in test 1a with some small differences in the density, pressure and velocity behind the shock wave. Additionally there is a discontinuity in the pressure at $x \approx 0.65$ that is not shown by the analytical solution in Fig. 6.3.

- Test 2 (see Fig. 6.4 right) corresponds to Problem 3 from Tab. 6.1, set up with equidistant distributed non-equal mass particles. In the initial conditions there was only a huge discontinuity in the pressure, here corresponding to temperature for the continuous density. This leads to a rightward propagating peaked shock front and a leftward propagating rarefaction wave. The rarefaction wave propagates faster because it is moving in the hotter medium, which has a higher sound speed.

- All three tests were also performed with lower resolution. The results showed basically the same features but not as sharp as the high resolution simulations.

In principle, the equal mass particle distribution is to prefer for example for the simulation of protoplanetary discs, because of the intrinsic resolution adaptiveness. But here the discontinuity in the density in test 1a leads to serious problems caused by the simple neighbour criterion (see Eq. 3.16) explained in the following.

Figure 6.5 a) shows a sample distribution of equal mass particles with a discontinuity in the density and the effect on the neighbour lists. In the upper part of the image a 1D density distribution with a jump in the density by a factor of 3 is shown. The lower part shows the corresponding particle distribution with a jump in the particle distances by a factor of $1/3$. The circles have radii corresponding to the local smoothing length for $N_{nn} = 4$. The particles 1 to 8 and 12 to 20 have two left and two right neighbours within the kernel. Because the simple neighbour criterion from Eq. 3.16 is used the 4 closest particles are used as neighbours. This produces problems for the particles 9, 10 and 11. These particles have one left and 3 right neighbours.

Figure 6.5 b) shows the corresponding neighbour lists. For example a force from particle 14 is effecting particle 11 but not vice versa. This asymmetry destroys the momentum conservation. For the shock tests the jump in density is a lot higher. Therefore, for test 1b and test 2 equidistant distributed particles are used.

This problem with the simple neighbour criterion shows that further improvements of the neighbour search are needed to allow symmetric force computations. The symmetric neighbour criterion (3.17) should be implemented.

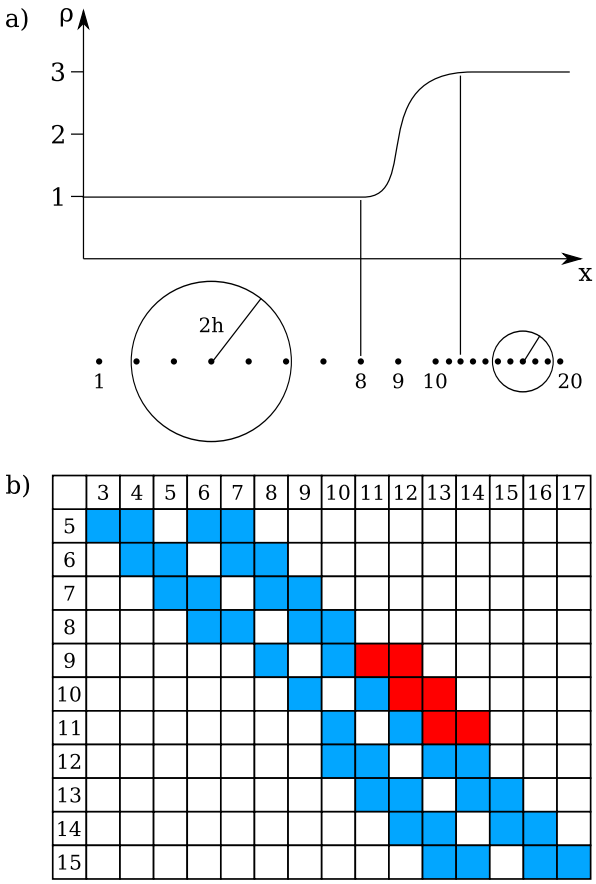

Figure 6.5: 1D sample distribution of equal mass particles with a discontinuity in the density (a) and corresponding interaction table (b). (Explanation see text.)

6.4 Sphere collapse

In the previous sections it was described, how the newly implemented SPH part of the code was tested. Next, the combination of the SPH part with the tree part has to be investigated. One possible test for this is the gravitational collapse of a uniform density sphere with pressure. This problem is analytically well understood. The analytical solution as can be found in (Truelove et al., 1998)will be summarised in the following.

A sphere with uniform density ρ_0 collapses under gravitational force and without pressure in the free-fall time t_{ff}

$$t_{\mathrm{ff}} = \sqrt{\frac{3\pi}{32G\rho_0}}, \tag{6.6}$$

with the gravitational constant G to a singularity. This process is independent of the size of the sphere.

As noted by Hubber et al. (2011) an amount of mass initially in the distance r_0 from the centre of

the sphere has moved to a distance $r < r_0$ from the centre in the time

$$t(r) = t_{\rm ff}\frac{2}{\pi}\left(\cos^{-1}\sqrt{\frac{r}{r_0}} + \sqrt{\frac{r}{r_0}}\sqrt{1-\frac{r}{r_0}}\right). \tag{6.7}$$

Particle tracks from the simulation can be compared to this analytical relation.

6.4.1 Collapse with pressure

When pressure is considered in the collapse of the sphere, the collapse will be superimposed by the thermal spreading of the sphere into the surrounding vacuum. The total energy of the sphere determines, to which extend it will dissolve. While the sphere is collapsing a rarefaction wave propagates inwards. This rarefaction wave is faster than a inward propagating sound wave because the matter itself is flowing inward. The speed of the rarefaction wave is the sum of the local flow speed and the sound speed. The time dependence of the position of the rarefaction wave can be found in (Truelove et al., 1998).

The time needed by the collapsing sphere to reach a density $\rho > \rho_0$ is given by

$$t(\rho) = t_{\rm ff}\frac{2}{\pi}\left(\eta + \frac{1}{2}\sin(2\eta)\right),$$

with

$$\cos(\eta) = \left(\frac{\rho}{\rho_0}\right)^{-1/6}.$$

The mass $M_{\rm rf}$ inside the rarefaction front is given by

$$M_{\rm rf} = M\left\{1 - 2\frac{c_s}{v_{\rm ff}}\arctan\left[\left(\frac{\rho}{\rho_0}\right)^{1/3} - 1\right]^{1/2}\right\}^3, \tag{6.8}$$

where M is the total initial mass of the sphere and $v_{\rm ff}$ the free-fall velocity

$$v_{\rm ff} = R\sqrt{\frac{8}{3}\pi G\rho_0}$$

with the initial radius R of the sphere. With

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

follows the radius $r_{\rm rf}$ of the rarefaction front

$$r_{\rm rf}(\rho) = \left(\frac{3M_{\rm rf}}{4\pi\rho}\right)^{1/3}.$$

For a given $\rho > \rho_0$ the radial position $r_{\mathrm{rf}}(\rho)$ of the rarefaction front can be calculated as well as the time $t(\rho)$ when this density is reached. So $r_{\mathrm{rf}}(t)$ can be obtained numerically and compare to the results of the simulation.

6.4.2 Test setup and results

To set up a uniform density sphere, particles are distributed on a regular three-dimensional grid with 50 particles in each dimension. From this cube a sphere is cut with a diameter of 1 AU resulting in approximately 65 000 particles. For later use with astrophysical problems, all physical constants like the gravitational constant and the Boltzmann constant are converted into the units Year, $\mathrm{M_{sun}}$, K and AU.

The masses of the particles are scaled so that the whole mass of the sphere is 1 $\mathrm{M_{sun}}$. The mean density of this sphere is $\rho \approx 1.9011 \mathrm{M_{sun}}/\mathrm{AU}^3$, which results in a free-fall time of $t_{\mathrm{ff}} \approx 0.063$ y.

For the simulation, the time-step size is chosen as $\Delta t = 1/1000 t_{\mathrm{ff}}$. The gravitational collapse of the sphere without pressure is shown in Fig. 6.6 together with the analytical solution according to (6.7). The points denote the radial positions of particles initially outside of 90%, 50% and 10% mass.

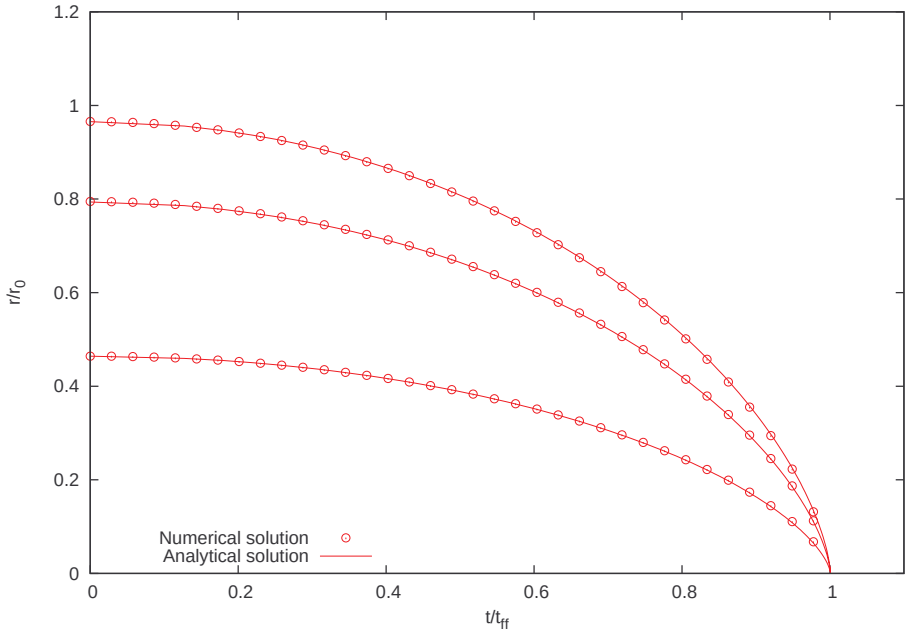

Figure 6.6: Free-fall collapse of a uniform density sphere involving only gravitational force. The points denote the radial positions of particles initially outside of 90%, 50% and 10% mass. The analytical solution is obtained with Eq. 6.7.

For the collapse with pressure, the temperature was chosen so that the inward propagating rarefaction wave can reach the centre within t_{ff}. Figure 6.7 shows the results of the simulation together with the analytical solution for the rarefaction wave. The green points show the radial positions of particles initially outside of 90%, 50% and 10% mass. The red lines show the collapse without pressure as presented in Fig. 6.6. The dotted line shows the analytical solution for the inward propagation of the

rarefaction front. The good agreement of the numerical results with the analytical solution is obvious. This agreement proves, that the code can treat the gravitation as well as the thermodynamics for example of a protoplanetary disc correctly. Now the code can be applied to real physical problems.

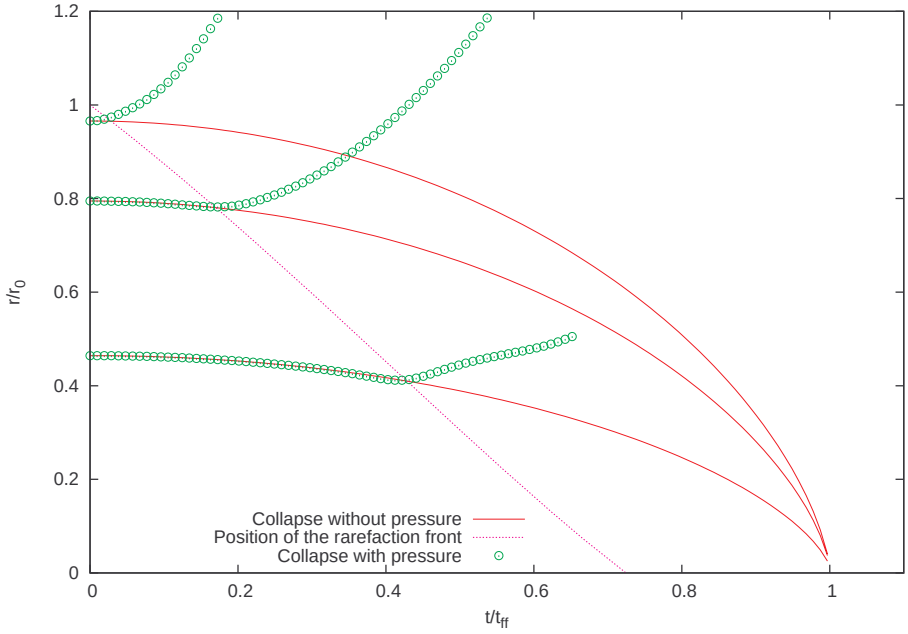

Figure 6.7: Collapse of a uniform density sphere involving gravitational and thermal forces. The green points show the radial positions of particles initially outside of 90%, 50% and 10% mass. The red lines show the collapse without pressure as presented in Fig. 6.6. The dotted line shows the analytical solution for the inward propagation of the rarefaction front.

6.5 A protoplanetary disc

After the tests were successful a first protoplanetary disc was simulated. Here a high-mass disc with $M_{\rm disc} = 0.1 M_{\rm star}$ around a solar mass star was used. As described before (see Sec. 2.3), such a disc is expected to fragment and form planets.

The density distribution $\rho(r,z)$ for the disc was chosen as

$$\rho(r,z) \propto r^{-p-1} \exp\left(-\frac{z^2 H^2}{r^2}\right), \tag{6.9}$$

where z is the height in cylindrical coordinates, $H(r)$ is the discs scale height with $H(r) = r/H_0$, and p index of the density distribution (see Sec. 2.2). Here $H_0 = 10$ AU and $p = -1$ were used, because they fit the observations.

The initial particle distribution is here obtained with a Monte-Carlo approach. The particles are placed at random coordinates with a probability proportional to the density. The so obtained physical system is not in equilibrium and has to be relaxed before used for simulations.

For the initial set up, 10^5 particles were distributed according tho the density distribution given by

(6.9) with a maximum extension of the disc $r_{\max} = 100$ AU. As the singularity in the density close to $r = 0$ would cause numerical problems, a central region of radius $r = 10$ AU was devoid of simulation particles.

For cold protoplanetary discs of low mass, self-gravitation and thermal forces can be neglected. In this case, the initial particle orbits around the star are approximately Keplerian. The Kepler velocities v_{k} are given by

$$v_{\mathrm{k}} = \sqrt{\frac{GM_{\mathrm{star}}}{r}}.$$

For high-mass discs ($M_{\mathrm{disc}} \approx 0.1 M_{\mathrm{star}}$), the self-gravitation can no longer be neglected and the orbits can no longer be approximated to be keplerian. Under the assumption, that a particle moves on a circular orbit, the centripetal force F_{c} is provided by the attracting forces of star and disc.

$$F_{\mathrm{c}} = \frac{m v_\phi^2}{r},$$

where m is the mass of the particle and v_ϕ the tangential velocity. Then the tangential velocity of a particle can be obtained with

$$v_\phi = \sqrt{\frac{(F_{\mathrm{star}} + F_{\mathrm{disc}})r}{m}}, \tag{6.10}$$

where F_{star} is here the gravitational force of the star and F_{disc} the gravitational force of the whole disc.

When as well thermal forces are taken into account, the orbits are even more complex. Due to the random placing of particles, strong numerical density fluctuations are likely to occur. The tremendous forces arising from the density gradients can accelerate particles to velocities high enough to leave the gravitational well of the system. To avoid this non-physical effect, the disc is set up with initial velocities following (6.10) and then the particle distribution is relaxed, which is described in the following. In Tab. 6.2 the setup parameters are summarised. They were chosen according to typical observed parameters (see Sec. 2.2).

Parameter	M_{star}	M_{disc}	$r_{\mathrm{disc,\,inner}}$	$r_{\mathrm{disc,\,outer}}$	H	p	N	T	Δt
Value	$1 M_{\mathrm{sun}}$	$0.1 M_{\mathrm{sun}}$	10 AU	100 AU	10 AU	-2	100 000	20 K	0.1 year

Table 6.2: Setup parameter for the protoplanetary disc.

During the relaxation process the thermal forces F_{t} are taken into account with a factor τ

$$F = F_{\mathrm{g}} + \tau F_{\mathrm{t}}$$

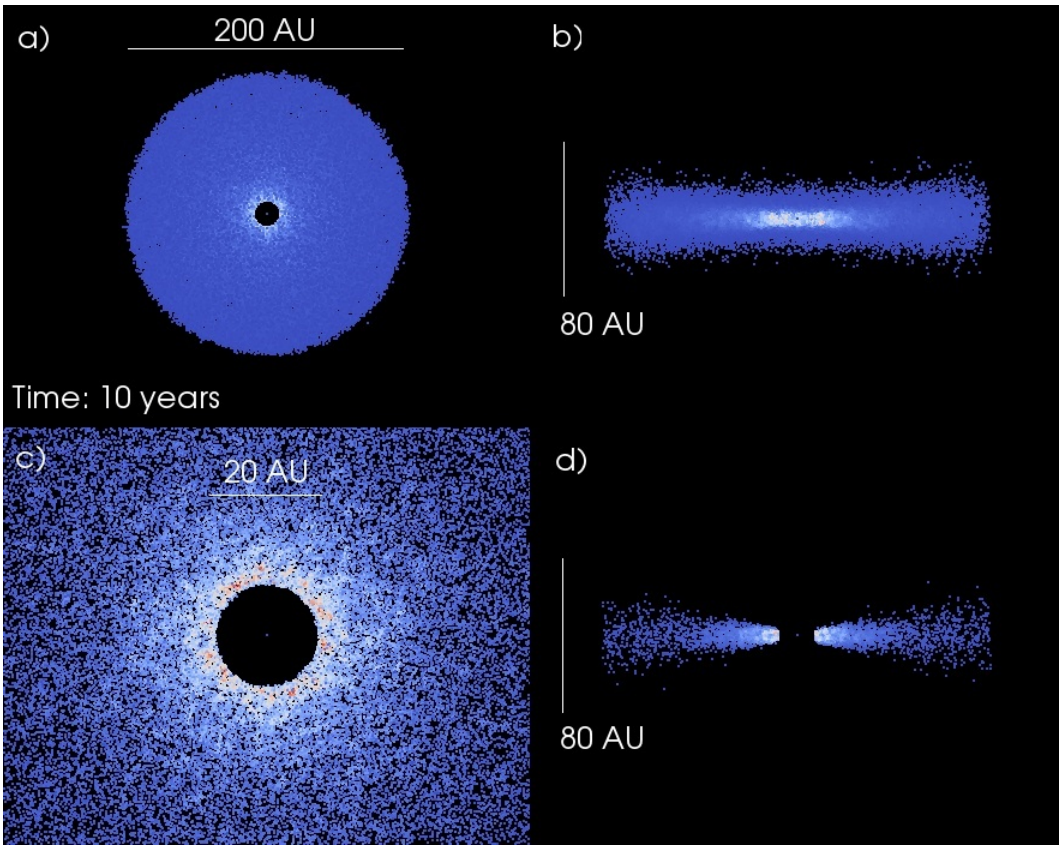

Figure 6.8: The relaxation process of a protoplanetary disc. The particles are coloured by density, from low (blue) via intermediate (white) to high density (red). a) shows the face on view on the disc. b) shows the edge-on view, where the front half of the disc is removed. c) shows a zoom on the inner part of the disc. d) shows a slice around the x-z-plane.

with

$$\tau = i/t_{\text{relax}},$$

where F is the total force, F_{g} is the gravitational force, i is the actual time-step and t_{relax} is the whole number of relaxation time-steps. That means, that the thermal forces are totally neglected at the beginning of the relaxation. Then their effect is increased linearly until they are fully considered at the end. For the relaxation 10000 time-steps with a time-step size of $\Delta t = 0.1$ years were used. The temperature increase due to viscous heating is limited by $T \leq 50$ K.

Figure 6.8 shows the disc at the beginning of the relaxation process. The particles are coloured by density, from low (blue) via intermediate (white) to high density (red). a) shows the face on view on the disc. The light blue close to the centre denotes the higher density. b) shows the edge on view with the front half removed. c) shows a zoom on the inner part of the disc. Here even the densest regions, coloured in red, can be seen. As can be distinguished from the black background, the particles are distributed inhomogeneous at small scales. d) shows a slice through the x-y-plane. Here the vertical structure of the disc can be seen.

The result of the relaxation is shown in Fig. 6.9. The four views and the colouring scheme are the same as in Fig. 6.8. In a) it can be seen, that the disc has expanded and that there is a complete light blue ring around the central hole. In c) it can be seen, that the colour gradient is much smoother than in Fig. 6.8. The particles are distributed much more homogeneous. In d) the vertical density profile can be seen. The disc is in hydrostatic equilibrium at least at the inner regions ($r \lesssim 40$ AU). The settling of particles to the discs midplane is prohibited by the pressure gradient resulting from the vertical density profile.

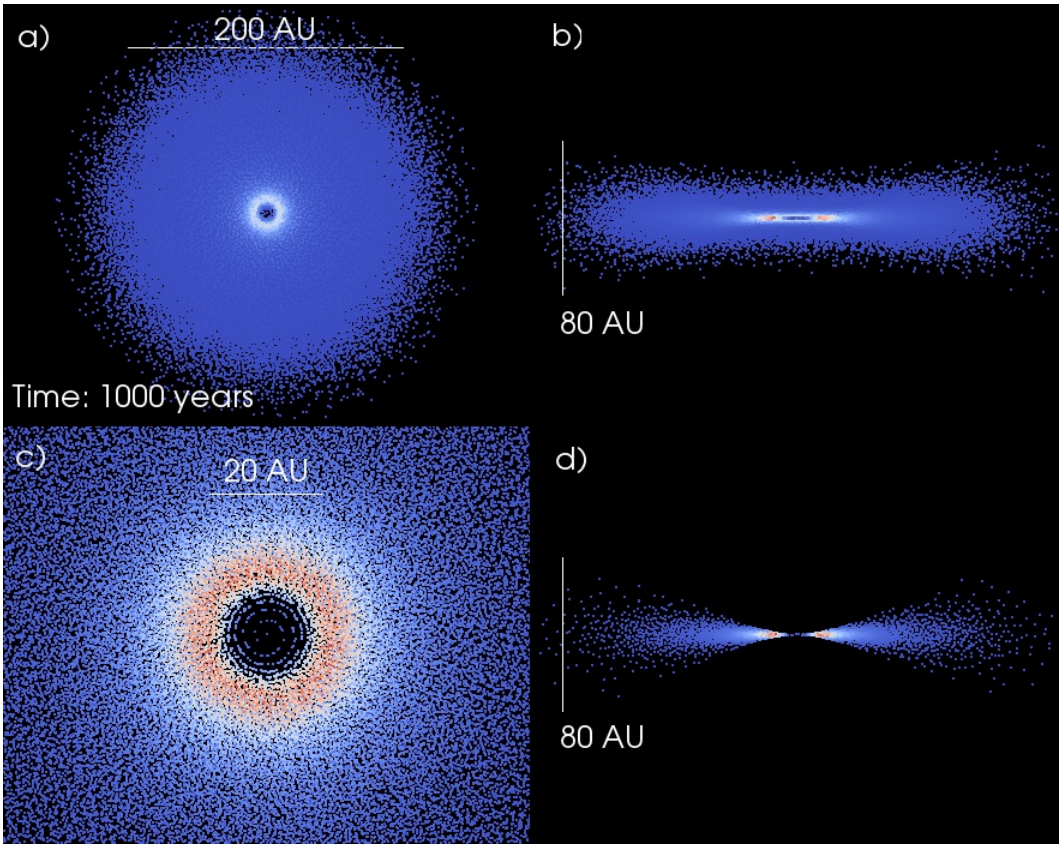

Figure 6.9: The relaxed protoplanetary disc. The particles are coloured by density, from low (blue) via intermediate (white) to high density (red). Comparison with Fig. 6.8 shows, that the particles are distributed much more homogeneous. The colour gradient is much smoother. a) shows the face on view on the disc. b) shows the edge-on view, where the front half of the disc is removed. c) shows a zoom on the inner part of the disc. d) shows a slice around the x-y-plane.

After the relaxation the disc was evolved with full consideration of the thermodynamics. Due to viscous heating the temperature in the inner parts of the disc increases. With the here chosen values for the mass of the disc, the density distribution, the time-step size and the viscosity parameters α and β (see Sec. 3.2.1) the temperature increases at the inner edge of the disc from $T = 50$ K to $T > 10\,000$ K within 10 years (or 100 time-steps) after the relaxation. These high temperatures lead to the dissolution of the disc within some 100 years. By implementing certain cooling mechanisms for example as described by Meru & Bate (2011) these inconvenient heating can be avoided.

Figure 6.10 shows the disc 100 years after the relaxation process. The four views and the colouring

scheme are the same as before. In a) and c) it can be seen, that the particles are pushed in the central hole. The density decreased compared to Fig. 6.9. In d) on can see, that the inner part of the disc is no longer in hydrostatic equilibrium as seen in Fig. 6.9 d). The increasing pressure, due to the high temperatures, accelerates the particles perpendicular to the discs midplane. 1000 years after the relaxation the disc will have lost approximately 10% of its mass due to this process. By that time the shape of the disc will have changed more to a flat torus, with a temperature of some 1 000 K inside this torus. However, the heating without cooling mechanisms or thermal conduction is not physical. For a more physical evolution of the disc further work is needed. The stable modelling and evolution of a protoplanetary disc with SPH codes is subject of ongoing research and far beyond the scope of this work.

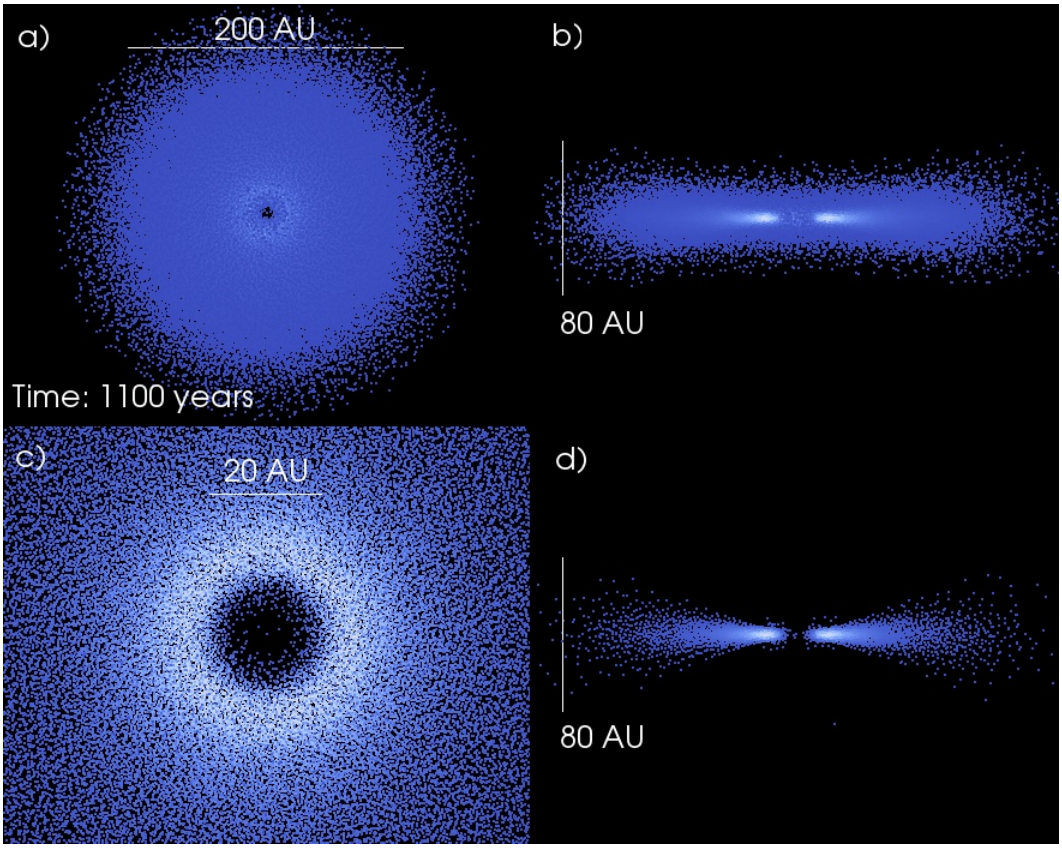

Figure 6.10: The protoplanetary disc after 100 years of evolution with full consideration of the thermodynamics. The particles are coloured by density, from low (blue) via intermediate (white) to high density (red). a) shows the face on view on the disc. b) shows the edge-on view, where the front half of the disc is removed. c) shows a zoom on the inner part of the disc. d) shows a slice around the x-y-plane. Comparison with Fig. 6.9 d) shows, that the previously thin and dense inner part of the disc is thicker and sparser. Some particles are already pushed out of the disc.

7 Conclusion

In this work, a numerical code was developed for simulations of self-gravitating protoplanetary discs on highly parallel computers. The new code, which is based on the existing code PEPC (Pretty Efficient Coulomb Solver), is expected to simulate up to $2 \cdot 10^9$ particles on highly parallel machines like the supercomputer Jugene operated by Forschungszentrum Jülich. Tests with up to 10^6 particles - our envisaged application to protoplanetary discs - were successfully performed. The capability to handle more particles has to be demonstrated in the future.

The previously existing code PEPC is mainly used for the computation of Coulomb forces in plasma physics applications. The new code has been adapted for the computation of gravitational forces. Obviously, this means an adjustment of the force constant. In contrast to the simulation of Coulomb forces, characterised by forces between two types of charged particles, for gravitational simulations the mass is the force determining property and therefore only one type of particles is used. With these modifications it is possible to simulate a self-gravitating disc in high resolution. However, thermal and viscous forces are neglected in this first version.

The main objective of this work was to model as well the fluid component of the disc. Therefore a module, implementing a basic Smoothed Particle Hydrodynamics (SPH) formulation, was developed for fluid computations. This method allows to model the hydrodynamical forces based on a particle algorithm. Here, the simulation particles have intrinsic properties like density, pressure and temperature. As most important part of the SPH module a parallel tree-based neighbour search algorithm was implemented. It has been demonstrated that its scaling is at least as good as the scaling of the code PEPC it is combined with, which is dominated by the $\mathcal{O}(N \log N)$ scaling of the force summation.

The evaluation of the density, pressure induced acceleration and change of temperature was implemented. Afterwards, tests were performed to show that the code describes all the relevant physical effects correctly. For protoplanetary discs these effects are sound waves, shocks and gravitation.

Simulating sound waves, it was demonstrated that the code can reproduce the analytical solution sufficiently. The relative deviation of the numerical solution for the density from the analytical solution was below $4 \cdot 10^{-6}$ after 10000 time-steps. This is more than sufficient for the planned applications.

It was demonstrated, that the code is capable of resolving shocks. Here, one dimensional discontinuities in density and pressure were used. The solutions for two different types of shock problems were compared to numerical solutions of other codes. It was demonstrated, that the physical phenomena can be reproduced correctly with the new code.

With a three-dimensional problem incorporating gravitation and hydrodynamics it was investigated whether the two code-parts work together correctly. The results obtained from the simulation were compared to analytical solutions, confirming the correctness of the new code.

Finally, a first test simulation of a high-mass protoplanetary disc was performed successfully. As expected, the disc dissolves due to increasing temperature caused by viscous heating in the inner parts of the disc. In a more realistic setting, cooling mechanisms have to be considered. However, during the relaxation process, where the temperature was limited, the disc evolved to hydrostatic equilibrium. The gravitational collapse perpendicular to the discs midplane was suppressed by stabilising hydrostatic pressure.

The new code is not the first of its kind. Most similar codes are parallelised for shared-memory computers, what limits applications to a few thousand CPU cores and the available memory to few 10 TB. There are only two widely used similar codes parallelised for distributed-memory computers with up to several 10^5 CPU cores and hundreds of TB memory. One of which is GADGET-2, developed by Springel (2005) and was used for the famous Millenium Simulation. At least the computation of the gravitational forces is known to be faster with PEPC than with GADGET-2 (P. Gibbon (2011), private communication). Thus, it is expected that the new code as a whole is faster, too.

The new developed simulation code can also be used for simulations of different astrophysical processes. Besides simulations of fragmenting protoplanetary discs, it can be used to investigate the role of viscosity in star-disc encounters. Also the possible recircularisation of the perturbed disc after such an encounter can be simulated. However, the code can as well be used to investigate other processes, like the formation of stars by gravitational collapse of molecular clouds (e.g. Price & Bate, 2010). The new code increases the resolution and performance of all mentioned simulations significantly and reduces the computation times. This crucial improvements are required for ongoing astrophysical research.

8 Prospect

However, even though first results were successfully produced with the new code, further improvements are possible.

From the physical point of view, first of all the energy conservation in the SPH module should be improved by the implementation of a symmetric force computation. Therefore, basically a slightly modified next neighbour search is needed. This modification should improve the treatment of strong density gradients, which occur for example during the fragmentation of a disc. Additionally, more physical processes like thermal conduction or radiation transport can be implemented for more realistic simulations. Especially radiation transport is important for a more realistic modelling of the radiative cooling of a protoplanetary disc. Thermal conduction can also provide a cooling mechanism.

From the computational point of view, the performance of the new code can be improved by changing the implementation to a hybrid scheme. This means, the usage of one MPI (Message Passing Interface) process and domain per compute-node. Within the MPI process, loops are parallelised with a shared-memory parallelisation scheme like OpenMP. When only one domain per compute-node is used, the number of domains is reduced for example by a factor of eight on the supercomputer Juropa, operated by Forschungszentrum Jülich, and the size of the domains increase by this factor. As explained in Sec. 5.3.3, this reduces the communication effort and memory usage and should result in an enormous performance gain.

Finally, it is planned to include the new code into a library, which can then be used by other simulation codes.

Bibliography

Amdahl, G. M. 1967, in Proc. of the SJCC, Vol. 30 (AFIPS), 483–485

Andrews, S. M. & Williams, J. P. 2007, ApJ, 659, 705

Andrews, S. M., Wilner, D. J., Hughes, A. M., Qi, C., & Dullemond, C. P. 2010, ApJ, 723, 1241

Bally, J. & Throop, H. n.d., Retrieved from `http://hubblesite.org/gallery/album/pr2001013b/` (26.06.2011)

Barnes, J. & Hut, P. 1986, Nature, 324, 446

Beckwith, S. V. W., Sargent, A. I., Chini, R. S., & Guesten, R. 1990, AJ, 99, 924

Boss, A. P. 1998a, Earth Moon and Planets, 81, 19

Boss, A. P. 1998b, Annual Review of Earth and Planetary Sciences, 26, 53

Breslau, A. 2010, in Proceedings 2010, JSC Guest Student Programme on Scientific Computing, ed. R. Speck, M. Winkel, 25–35

Clarke, C. J., Harper-Clark, E., Meru, F., & Lodato, G. 2008, in Astronomical Society of the Pacific Conference Series, Vol. 398, Extreme Solar Systems, ed. D. Fischer, F. A. Rasio, S. E. Thorsett, & A. Wolszczan, 341–+

Forgan, D. & Rice, K. 2009, MNRAS, 400, 2022

Gammie, C. F. 2001, ApJ, 553, 174

Ghez, A. M., Neugebauer, G., & Matthews, K. 1993, AJ, 106, 2005

Gibbon, P. 2003, PEPC: Pretty Efficient Parallel Coulomb-solver, Tech. Rep. FZJ-ZAM-IB-2003-5, Central Institute for Applied Mathematics

Gibbon, P., Speck, R., Berberich, B., et al. 2010a, in NIC Symposium 2010, ed. G. Münster, D. Wolf, M. Kremer, 383–390

Gibbon, P., Speck, R., Karmakar, A., et al. 2010b, Plasma Science, IEEE Transactions on, 38, 2367

Gingold, R. A. & Monaghan, J. J. 1977, MNRAS, 181, 375

Haisch, Jr., K. E., Lada, E. A., & Lada, C. J. 2001, ApJ, 553, L153

Hernández, J., Hartmann, L., Calvet, N., et al. 2008, ApJ, 686, 1195

Hubber, D. A., Batty, C. P., McLeod, A., & Whitworth, A. P. 2011, A&A, 529, A27+

Isella, A., Carpenter, J. M., & Sargent, A. I. 2009, ApJ, 701, 260

Levison, H. F. & Stewart, G. R. 2001, Icarus, 153, 224

Lodato, G. & Clarke, C. J. 2011, MNRAS, 413, 2735

Lodato, G. & Rice, W. K. M. 2005, MNRAS, 358, 1489

Lucy, L. B. 1977, AJ, 82, 1013

Lynden-Bell, D. & Pringle, J. E. 1974, MNRAS, 168, 603

McCaughrean, M. n.d., Retrieved from `http://www.gps.caltech.edu/~gab/astrophysics/astrophysics.html` (24.06.2011)

Merlin, E., Buonomo, U., Grassi, T., Piovan, L., & Chiosi, C. 2010, A&A, 513, A36+

Meru, F. & Bate, M. R. 2011, MNRAS, 410, 559

Monaghan, J. J. 1992, ARA&A, 30, 543

Monaghan, J. J. 2005, Reports on Progress in Physics, 68, 1703

Monaghan, J. J. & Gingold, R. A. 1983, Journal of Computational Physics, 52, 374

Monaghan, J. J. & Lattanzio, J. C. 1985, A&A, 149, 135

NASA/ESA and Feild, A. 2003, Retrieved from `http://www.spacetelescope.org/images/opo0319f/` (26.06.2011)

Pfalzner, S. & Gibbon, P. 1996, Many-body tree methods in physics (Cambridge University Press), ix + 168

Pfalzner, S., Olczak, C., & Eckart, A. 2006, A&A, 454, 811

Price, D. J. & Bate, M. R. 2010, in American Institute of Physics Conference Series, Vol. 1242, American Institute of Physics Conference Series, ed. G. Bertin, F. de Luca, G. Lodato, R. Pozzoli, & M. Romé, 205–218

Pringle, J. E. 1981, ARA&A, 19, 137

PSC. n.d., Retrieved from `http://blacklight.psc.edu/` (25.07.2011)

Rice, W. K. M., Armitage, P. J., Bate, M. R., & Bonnell, I. A. 2003, MNRAS, 339, 1025

Rice, W. K. M., Lodato, G., & Armitage, P. J. 2005, MNRAS, 364, L56

Sod, G. A. 1978, Journal of Computational Physics, 27, 1

Springel, V. 2005, MNRAS, 364, 1105

Springel, V. 2010, ARA&A, 48, 391

Springel, V. & Hernquist, L. 2002, MNRAS, 333, 649

Toomre, A. 1964, ApJ, 139, 1217

Toro, E. F. 1997, Riemann Solvers and Numerical Methods for Fluid Dynamics (Berlin Heidelberg: Springer)

Truelove, J. K., Klein, R. I., McKee, C. F., et al. 1998, ApJ, 495, 821

Unsöld, A. & Baschek, B. 2005, Der neue Kosmos – Einführung in die Astronomie und Astrophysik, 7th edn. (Berlin: Springer)

Vicente, S. M. & Alves, J. 2005, A&A, 441, 195

Warren, M. S. & Salmon, J. K. 1993, in Proceedings of the 1993 ACM/IEEE conference on Supercomputing, Supercomputing '93, 12–21

Warren, M. S. & Salmon, J. K. 1995, Computer Physics Communications, 87, 266

Williams, J. P. & Cieza, L. A. 2011, ArXiv e-prints

Winkel, M., Speck, R., Hübner, H., et al. 2011, Computer Physics Communications, submitted

Acknowledgements

First of all, I would like to thank my supervisors, Prof. Dr. Susanne Pfalzner and Dr. Paul Gibbon, for giving me the opportunity to write this thesis. Without their guidance, support and patience this work would not have been possible.

Furthermore, I want to thank all those who have been my colleagues during this time and supported this work with competent advice: Lukas Arnold, Christina Hövel, Helge Hübner, Thomas Kaczmarek, Robert Speck, Manuel Steinhausen, Mathias Winkel.

And last but not least, I want to thank my family, my girlfriend, and my friends for their support, encouragement, endurance.

Jül-4340
August 2011
ISSN 0944-2952