Smoothed Particle Hydrodynamics

for astronomical model demonstration (2010)

Thomas A. Grønneløv*

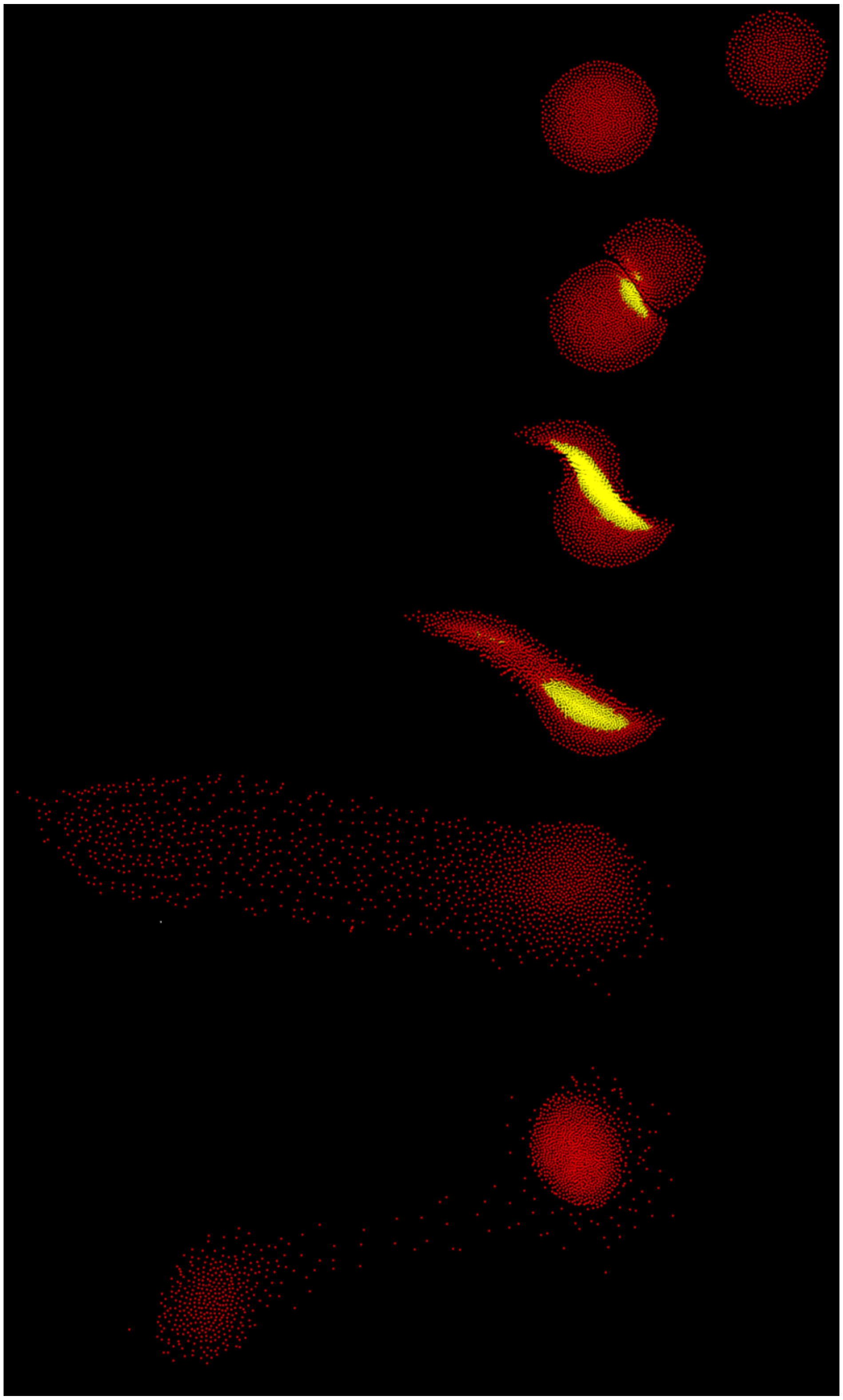

Figure 1: Collision between protoplanets

*e-mail:Tag@Greenleaf.Dk

Abstract

The theories explaining many astronomical observations come from an iterative process of forming a general theory, designing a model, testing and refining. This project aims to retrace the steps leading to some of the astronomical theories which are now considered fact.

The focus will not be on state of the art and very complex astronomical models and the maximal degree of realism. Instead it will be on creating the simplest possible models which still realistically demonstrate the mechanisms at work at a level of accuracy that would make them relevant for introductionary astronomy courses.

The SPH method will be used to create models for some astronomical problems. The models will be described, implemented and tested to provide a comparison between the observations and the models.

This project aims to demonstrate those models and do it in such a way that it can be considered a dynamic lab experiment where the effect of changing the parameters can be observed live. Motivation

In the astronomical field various very complex SPH models are being put to everyday use forming new theories, but the focus is naturally not on old previously solved problems. This means that there is a lack of illustrative physical simulations of those problems.

Contents

1 Notation and symbols

This section serves as a small collection of symbols and units used throughout the text.

P	Pressure	N/m^2
ρ	Density	kg/m^3
$\delta(\alpha,\beta)$	Kronecker delta function	unitless
a	acceleration	m/s^2
v	Velocity	m/s
m	Mass	kg
ϕ_i	Gravity potential at particle i	N
c	Speed of sound	m/s

2 Fluid dynamics

In astronomy the objects being simulated are generally fluid like objects. That may be gas clouds or fluid objects such as stars or even galaxies which behave like inviscid[1] fluids [FLU]. When simulating more rigid objects such as planets or moons, the fluid assumption still generally holds for gas planets and rock planets with a molten core. Under extreme stress, such as during interplanetary collisions, even solid rock planets will behave like fluid [Liu and Liu 2003].

For this reason, the behavior of fluid is relevant to the simulations in this paper and it will be briefly described.

2.1 Navier-Stokes fluid equations

The Navier-Stokes equations describe how the velocity field of a fluid changes over time based on the velocity field itself, the resulting pressure and viscosity as well as external forces. The equations will not be derived here, but its derivation in relation to the Smoothed Particle Hydrodynamics method can be found in detail in [Liu and Liu 2003].

[1]Without internal friction known as viscosity

The equations are based on the conservation of three physical properties of a fluid. These are conservation of mass, momentum and energy.

Conservation of mass For a Lagrangian fluid (explained in detail in section 3) one deals with the actual fluid particles which each have constant mass. This means that we are sure that mas is neither lost nor gaines, but in general fluid dynamics this has to be explicitly required as written in (1) where v is velocity and ρ is density.

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot v \tag{1}$$

Conservation of momentum Momentum is mass times velocity and for a closes system the momentum should remain constant. This is written as (2) where μ is the viscosity quotient which we will not use (see section 4.1.2), P is pressure, v is velocity and ρ is density while F is external forces. The forces will in our case only be gravity.

$$\rho\left(\frac{\partial}{\partial t} + v\cdot\nabla\right)u = -\nabla P + \mu\nabla\cdot(\nabla v) + F \tag{2}$$

Conservation of energy The energy is also a constant in a closed system (3), though it may be either in the form of macroscopic ordered kinetic energy or microscopic unordered heat energy.

$$\rho\left(\frac{\partial\varepsilon}{\partial t} + v\cdot\nabla\varepsilon\right) - \nabla\cdot(K\nabla T) + P\nabla\cdot v = 0 \tag{3}$$

3 Smoothed Particle Hydrodynamics

Smoothed Particle Hydrodynamics, hereafter SPH, is a so called Lagrangian method as opposed to an Eulerian method. Where an Eulerian method defines the space in which a material moves and does not track any part of the substance in particular, but rather keeps track of how much is located at a particular position in space and what the properties is of the material currently at that position in space; a Lagrangian method deals explicitly with a part of the material and tracks this material through space.

A good analogy comes from meteorology where we can either track a property, such as air temperature, in two distinctly different methods. The Eulerian method would be to place weather stations at fixed positions on the ground and let them track air temperature. They will then report on the current temperature of the air at their specific position. They will not track any part of the air in particular. The alternative Lagrangian method would be a weather balloon which is launched and then move with the air. It will constantly measure the same sample of the air and track *that* air as it moves through space.

Particle SPH is a Lagrangian method and it works by partitioning a material into a number of particles. Each particle *is* a part of the material and this subset of the material, or the particle, is tracked through space over time and this particle composition is the reason for the term ”particle” in the name.

Smoothed The term ”smoothed” comes from the fact that partitioning a material into a number of point sized particles will both be physically unrealistic and it will give problems when defining gradients of various properties inside the material. The smoothing is also what lets the method define a dynamic neighborhood relationship among near particles and this is the equivalent of a mesh in this otherwise mesh free method.

Hydrodynamics The method is not restricted to simulating fluids, but it is in this area that it is most powerful when compared to other methods. Where most other methods either describe a material through a mesh, which is not easily updated, or describe space in an Eulerian fashion, SPH does not rely on fixed meshes or spatial partitioning since it dynamically define an redefine particle interrelations. At first an Eulerian grid based method may seem to be good enough, and it is in fact used a lot in fluid dynamics. The method fails however when the potential space in which a material may move is very large since even the empty areas will have to be included in the grid and thereby cost storage and computational resources. In astronomical simulations there is a lot more empty space than there is material and for this reason a Lagrangian method which tracks the actual material rather than the space is preferred.

3.1 General idea

Given a particular partial differential equation PDE which is to be solved for a material the problem domain, the material (assume a fluid for clarity for now), is first discretized by partitioning it into a number of particles which each carry with it part of the material, that being mass, temperature, velocity and position. After this partitioning one needs a method to calculate the field function describing the material at any point in space. This continuous field function is used to estimate derivatives of the field itself at the particle positions. As an example consider a pressure force which is the derivative of the pressure field. Given the field and its derivatives one can rewrite the PDE into a set of ordinary differential equations ODE which can be integrated over time to advance the simulation of the material.

Smoothing If we have a number of particles distributed in space and each particle have a temperature T associated with it, then what is the second derivative of T $\nabla^2 T$ at the particles location? This value is needed if we want to figure out how each particles temperature will change over time. One particle in itself cannot define this second

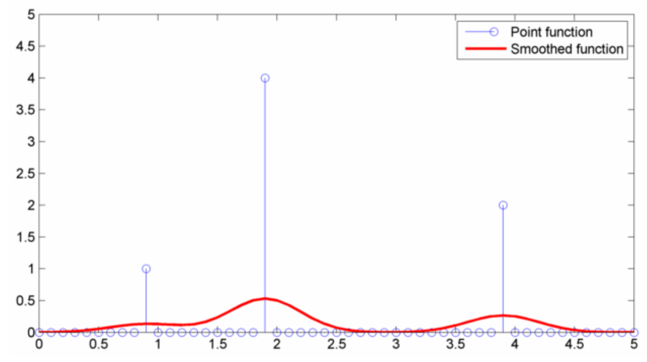

Figure 2: Particles can be defined as discontinous points in space or theycan be smoothed into a continous function

derivative, or Laplacian since it will depend on the neighbors as well.

When looking at the particles isolated, we can consider them as points in space. The field is defined only at the particles position and is considered zero elsewhere. This is not a differentiabel function and this view does not let us calculate derivatives of the field. When they are instead smoothed as in Figure 2, we obtain a smooth function which *is* differentiable. The figure is the result of a 1D convolution of the point values with a Gaussian function.

Smoothing can be viewed conceptually different ways. Either the particle is actually smeared out over space as a soft blob and thereby touches nearby particles and influence them, or we can view the smoothing as a probability density function, as in quantum mechanics, where the position is not an absolute. Two particles may influence each other if their probability densities overlap somewhere and the higher the probability of the particles being there, the higher probability of influence. Averaging this influence will weight it by the probability and we have the same result as in the other view. Personally however I prefer to view the smoothing as a weighted averaging among nearby particles. This view seems to make more sense when dealing with the derivatives of the field.

3.2 Integral formulation

The more mathematical basis for SPH is explained herein.

The integral formulation is based on the identity (4) where $\delta(x)$ is the Dirac delta function which is 1 for x=0 and else 0. The expression (4) simply states that the value of a function at x is equal to the integral of this function over the entire domain x' multiplied with $\delta(x-'x)$ which is only non zero at x=x'. In other words, we throw out all values of the integral except the sample at the exact location we are interested in. This formulation is identical to the point based view seen in Figure 2.

$$f(x) = \int_\Omega f(x')\delta(x-x')dx' \tag{4}$$

If we now exchange the $\delta(x-x')$ with a more smooth function $W(x-x',h$ which has its smoothness defined by the extra parameter h, then we can rewrite to (5). This expression is however only an approximation since we are essentially blurring the image to remove the points. For this reason smoothed values are said to have $O(h^2)$ accuracy. This error measure is derived in [Liu and Liu 2003]

$$f(x) \approx \int_\Omega f(x')W(x-x',h)dx' \tag{5}$$

In SPH W is designated ”smoothing kernel” and this is the function which is used through out to go from the discretized point based view to a continuous field. Here h is the smoothing length which defines the influence area of the particles when describing the field.

3.3 Field approximation

The particles have an influence radius and their influence is weighted according to the kernel function based on their proximity. More concretely this can be by rewriting the integral to a summation over all particles as seen in (7) by assuming a particle volume of ΔV. Now x_j means the position of the j'th particle.

$$f(x) = \int_{\Omega} f(x')W(x-x',h)dx' \tag{6}$$
$$f(x) \approx \sum_j f(x_j)W(x-x_j,h)\Delta V$$

The volume can be discretized in terms of mass m and density ρ since density is equal to mass divided by volume $\rho = kg/m^3$. Both mass and density are quantities which make sense are generally needed in a fluid simulation.

$$\begin{aligned} f(x) &\approx \sum_j f(x_j)W(x-x_j,h)\frac{1}{\rho_j}(\rho_j\Delta V) \\ &= \sum_j f(x_j)W(x-x_j,h)\frac{m_j}{\rho_j} \end{aligned} \tag{7}$$

The function f(x) can now be *any* quantity which is known for the particles such as temperature, velocity, pressure or something else, and the above expression lets us find both the field values and the derivatives of the field from only a particle definition.

Number of support particles When a particle is left without any neighbors inside its support radius, then the field value at its position will be equal to the particles own value. This makes sense since it is an average of one value. It is however then essentially a point sampling of the field and at this point a supposedly fluid material is approximated by an infinitely rigid particle moving about on its own. While atoms may be considered small and very rigid, the particles in fluid dynamics often represent larger collections of atoms and those large collections are not rigid objects. If they were, they would describe solids and not fluid. This is therefore a bad approximation therefore the solution is inaccurate if the number of support particles is very small. A recommended number of support particles is 5 for 1D, 21 for 2D and 57 for 3D [Liu and Liu 2003]. The values may vary slightly from model to model depending on the level of smoothness desired. More about this in section 7.1.

It is interesting to note that if the number of support particles should be kept at a fixed value and the particles are very very tightly packed, then the smoothing length h should be very very small. This would mean that the kernel would approach the delta function at which point there is no smoothing but rather an *accurate* point sampling. In other words, if the particle discretization is reversed by packing the particles so tight that they start to form a continuum, then the smoothing method will transform into the initial continuum method (4) using the Dirac delta operator.

3.4 Derivatives of the field

The spatial derivative of the smoothed function is obtained simply by replacing W with $\nabla \cdot W$. This is seen from the following rewrite of (5) with $\nabla \cdot$ added to the equation.

$$\nabla \cdot f(x) = \int_{\Omega} [\nabla \cdot f(x')]W(x-x',h)dx' \tag{8}$$

Differentiating f(x') multiplied with W can be done using the chain rule which gives

$$[\nabla \cdot f(x')]W(x-x',h) = \tag{9}$$
$$\nabla \cdot [f(x')W(x-x',h)] - f(x') \cdot \nabla W(x-x',h) \tag{10}$$

Combining (8) and (10) gives

$$\nabla \cdot f(x) = \int_{\Omega} \nabla \cdot [f(x')W(x-x',h)]dx' - \tag{11}$$
$$\int_{\Omega} f(x') \cdot \nabla W(x-x',h)dx' \tag{12}$$

Gauss' divergence theorem lets the first integral term be rewritten to a surface integral rather than a volume in 3D or an area in 2D as it is now. The vector n is the normal at the surface boundary.

$$\nabla \cdot f(x) = \int_S f(x')W(x-x',h)] \cdot nds - \quad (13)$$

$$\int_\Omega f(x') \cdot \nabla W(x-x',h)dx' \quad (14)$$

For a kernel with compact support, the W(x-x') will be zero from the surface and outwards. This means that it can be dropped from the expression since we assume compact support. This lets us rewrite the expression into its final form (15). As mentioned in [Liu and Liu 2003] artificial boundaries may influence the actual surface in a way that make the surface integral non zero, but in this project there are no such things as boundaries since the entire space is the scene for the simulations.

$$\nabla \cdot f(x) = -\int_\Omega f(x') \cdot \nabla W(x-x',h)dx' \quad (15)$$

3.5 Kernels

A kernel is a function of a n-dimensional distance vector r and a smoothing length h. This function is used to scale the contribution from particles inside the support radius of a certain point for a n-dimensional domain.

For a kernel to give a reasonable smoothing of a particle sampled field, it has to fulfill some general requirements, and in some cases it has to be designed to fit a specific purpose as described in the following. It should be noted that the following requirements are sometimes violated for specific purposes if, for some reason, it actually makes sense, in the problem being described, to have an uneven smoothing function where particles in one direction influence the field value more than particles at other directions.

Normalized The kernel function must be normalized, meaning that it should integrate to 1. This is an obvious requirement since otherwise the function being smoothed would lose or gain in value. If the function was for example a mass distribution, then a non normalized kernel would change the mass of the material. To not break the laws of conservation, the integral must be 1.

$$\int_\Omega W(x-x',h)dx' = 1 \quad (16)$$

Approach δ for small h The kernel should approach the Dirac delta function when h approaches zero. This stands to reason since a h of zero means there is no smoothing and that we have returned to the initial discontinuous function from (4).

$$\lim_{h \to 0} W(r,h) = \delta(r) \quad (17)$$

Compactness The kernel should vanish when r is outside the support radius. Since the support radius is the radius of the smoothing, no particle outside this radius should influence the result. This is primarily to ensure that computational resources is not wasted on calculating the minuscule influence of a far far away particle, but apart from gravity, there generally is no very long range forces between particles and gravity is not handled by smoothing kernels as described in section 6.2.

$$\lim_{r \to h} W(r,h) = 0 \quad (18)$$

Positive The function should be positive inside its support. This is a requirement which is based on physics in that a particle carrying a positive quantity should never add negatively to the direct summation over the supporting particles. It would not make sense for a positive quantity to add negatively.

$$W(r,h) \geq 0 \text{ for } ||r|| \leq h \quad (19)$$

Monotonous decreasing as $|x - x'|$ grows The weighting of nearby particles should be such that particles close by will count more than particles far away due to the assumption that any force working between them will diminish with distance and not grow. It is also preferred if the derivatives of the kernel all go towards zero at the boundary since this will improve numerical stability to not have a sharp edge at the boundary.

$$W(r,h) \geq W(r+\Delta r,h) \text{ for } \Delta r \geq 0 \tag{20}$$

Even An even kernel function is one for which (21) always holds. Again this seems reasonable in that the contribution from two particles on opposite sides from a given position should influence the point at the position with equal weight. From a physics perspective, the forces acting on the particles should be the same, so the direction from one particle to the other should not influence the weighting.

$$W(x-x',h) = W(x'-x,h) \tag{21}$$

Smooth A smoothing kernel should be a sufficiently smooth in itself. This is required for numerical stability since numerical integration works badly with discontinuous forces. An exception to this rule is the spiky kernel (section 3.5.2) which is actually discontinuous around 0.

3.5.1 Kernel for density

Often density is smoothed using a function which looks like a Gaussian. In [Gingold and J. 1977] they state that the physically most reasonable kernel is a Gausian. A Gaussian distribution is what is commonly seen in nature and it resembles a normal distribution of probability density function. A Gaussian is however not compact since it never becomes zero. For this reason the function can either be truncated at a distance of h or it can be replaced

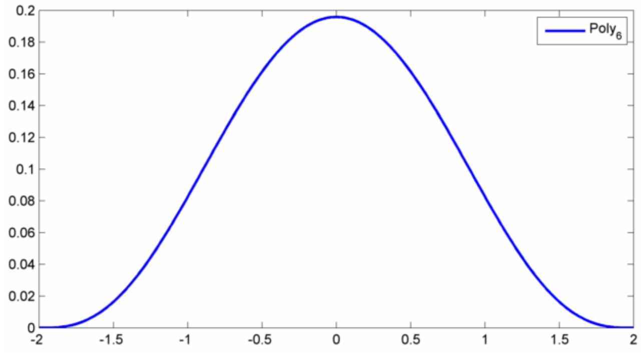

Figure 3: The $poly_6$ kernel which approximates a Gaussian inside the support radius which is here 2

by an approximation that looks like the Gaussian and which is compact.

One such function is known as poly6 which is defines as in (22) and this is the function we have used in the simulator. The ”problem” for some with the $poly_6$ is mainly in its derivatives since it is not infinitely differentiable as the Gaussian is. We do however not need the derivatives of the density so it does not in our case.

$$W_{poly6}(r,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - ||r||^2)^3 & 0 \leq ||r|| \leq h \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

The fraction in the equation is the normalization constant which is pre-computed for a given smoothing length. This means that runtime we need only calculate the $(h^2 - r^2)^3$ and here it is interesting that r^2 is used since the squared distance between two points is calculated very quickly and we void taking the more costly square root to find r

3.5.2 Kernel for pressure forces

While the poly6 kernel is easy to evaluate it has the same ”problem” as the Gaussian which it emulates. Its first derivative approaches zero when $||r||$ approaches zero. This means that as two particles move closer and closer together, the pressure gra-

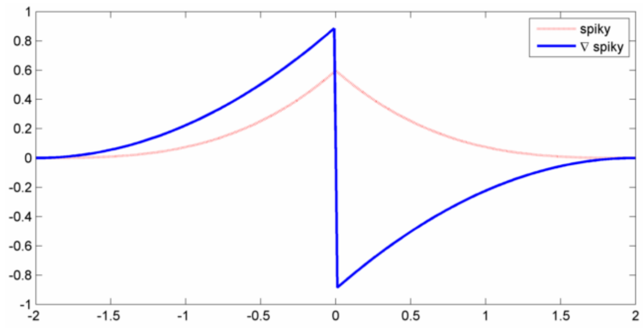

Figure 4: Spiky kernel and its gradient for a support radius h of 2

dient will actually decrease and for $||r|| = 0$ it will be zero. To remedy this problem the so called Desbruns spiky kernel (23) was designed and is seen along with its gradient in Figure 4.

$$W_{spiky}(r,h) = \frac{15}{\pi h^6} \begin{cases} (h-||r||)^3 & 0 \le ||r|| \le h \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

This kernel is discontinuous in its first derivative and is shaped such that pressure forces will increase exponentially as two particles approach each other and as they pass each other the sign of the pressure force will invert.

For pressure calculations the gradient of the spiky kernel (24) is used.

$$\nabla W_{spiky}(r,h) = -\frac{45}{\pi h^6} \begin{cases} \frac{r}{||r||}(h-||r||)^2 & 0 \le ||r|| \le h \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

3.5.3 Kernel for viscosity

We have no kernel for viscosity here. In astronomy the ordinary physical viscosity is not used (see section 4.1.2). Instead we have an artificial viscosity which is primarily used to describe shocks and to some smaller extent shear viscosity.

4 Particle rate of change

The physical parameters carried by a particle, and describing it, are the position vector x, the velocity vector v, its mass m and its internal energy ε.

Mass is a constant and need not be considered.

The rate of change for position is the velocity which is known $\frac{dx}{dt} = v$, but the rate of change of the velocity, the acceleration, $\frac{dv}{dt} = a$ has to be calculated from pressure forces, viscosity forces and gravity forces as well as from the particles mass. The rate of change of internal (heat) energy is calculated as a side effect from viscosity and can additionally be calculated from heat conduction and radiation between the particles.

4.1 Acceleration

When calculating the forces controlling a fluid, it is commonly gravity, pressure, viscosity and surface tension which are used. We will ignore surface tension since its influence is very small on the large scale. While it has a huge influence on a small drop of water, it is irrelevant in describing a ball of molten rock thousands of kilometer in diameter. Here gravity will shape the object along with pressure forces and viscosity. For rotational gas clouds, and especially for accretion disks, which is not simulated in this project, magnetic forces are also relevant, but we will ignore that force here since we will primarily deal with models where the effect is not visible. The magnetically induced jets of tiny charged particles from the poles of a star and the magnetically induced turbulence in gas clouds will be at a scale that is beyond the focus of this text.

The total rate of change in velocity is given by

$$\frac{dv_i}{dt} = a_i^{pressure} + a_i^{viscosity} + a_i^{gravity} \tag{25}$$

4.1.1 Pressure

Objects are pushed away from areas of high*er* pressure towards areas with lower*er* pressure. Therefor knowing the pressure field P, or rather its gradient ∇P, lets us calculate the pressure force. The force is then the negative gradient (26). The pressure is first calculated for each particle from the density ρ, the temperature T and a suitable Equation Of State as described in section 5. Then the pressure gradient is calculated through direct summation.

$$F_{pressure} = -\nabla P \tag{26}$$

$$F_i^{pressure} = -\rho_i \sum_{j \neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W(r_{ij}, h_{ij}) \tag{27}$$

which gives the acceleration

$$a_i^{pressure} = -\sum_{j \neq i} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla W(r_{ij}, h_{ij}) \tag{28}$$

4.1.2 Viscosity

Viscosity describes the internal friction among the particles in a substance. There are two general types which are bulk viscosity and shear viscosity. Bulk viscosity describes viscosity between particles being pushed into each other by collision while shear viscosity is viscosity from particles moving past each other. It transforms ordered macroscopic kinetic energy into random microscopic kinetic energy.

In astronomy viscosity is generally dropped from simulations [Flebbe et al. 1994]. The scale of both time and space is such that the ”ordinary” physical viscosity often has very little influence. Shocks from impacts are however quite important and must still be taken into account since they are not described by the pressure forces. The momentum lost from pressure forces, when particles collide, is returned as soon as the particle have come to a stop and start to move away from each other again. Momentum lost to shocks is not returned and while the total momentum for all particles is constant, the momentum for the individual shocked particle is irreversibly converted into heat. In other words, particles bounce elastically off of each other when controlled only by pressure forces while viscosity better described an inelastic collision. Pressure forces alone also rarely prevent particle inter penetration.

The acceleration from viscosity is given by

$$a_i^{viscosity} = \frac{1}{2} \sum_{j=1}^{N} m_j \prod_{ij} \nabla W(r_{ij}, h_{ij}) \tag{29}$$

Here Π is artificial viscosity tensor which was first defined by Monaghan in 1989 [Monaghan 1989] and which is used to describe shocks in colliding material. For this reason it is only active for particles moving towards each other while it is zero for particles moving away from each other. The artificial viscosity tensor for two particles i and j is

$$\prod_{ij} = \frac{-0.5\alpha\mu_{ij}(c_i + c_j) + \beta\mu_{ij}^2}{0.5(\rho_i + \rho_j)} \tag{30}$$

where the parameters α is a combined shear and bulk viscosity that dampens post-shock oscillation and is dominant at low relative velocities. β is used to get a Neumann-Richtmeyer viscosity which is also used in other simulation methods such as a grid based finite element simulation. This term is dominant for high relative velocities.

In many simulations, the parameters are set to $\alpha \approx 1$ and $\beta \approx 2$.

$$\mu_{ij} = \begin{cases} \frac{v_{ij} \cdot r_{ij}}{h_{ij}(|r_{ij}|^2/h_{ij}^2 + \eta^2)} & for\ v_{ij} \cdot r_{ij} < 0 \\ 0 & otherwise \end{cases} \tag{31}$$

Here c_i is the speed of sound at particle i as described in section 5. η^2 is a stabilizing term generally set to 0.01. v_{ij} is the difference in velocity vectors between particles i and j.

4.1.3 Gravity

Gravity is given special treatment in section 6.

4.1.4 Total momentum equation

Combining and rewriting all of the above into one momentum equation gives

$$\frac{dv_i}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + \Pi_{ij} \right) \nabla W(r_{ij}, hij) - \nabla \phi_i \tag{32}$$

The gravity potential at particle i is represented by ϕ_i.

4.2 Change in internal energy

If temperature is relevant in a simulation then internal energy needs to be included. We deal primarily with the internal energy of a particle and not its temperature. The relation is given through the specific heat capacity c, having the unit $kJ/(kgK)$, of the material so that $T = \frac{\varepsilon}{mT}$

The internal energy ε changes due to the loss of kinetic energy from artificial viscosity and due to heat diffusion between neighboring particles as well as due to energy being radiated from particles. In our models the primary cause of a change in energy is from artificial viscosity and compression while heat diffusion and radiated heat work too slowly and too weakly to have any significant impact on the results. All three will be described here, but only the energy from artificial viscosity will be implemented in the simulator.

This means that momentum can be transfered into heat due to viscosity and compression. While the energy from compression can be transfered back to momentum, the heat from viscosity will never leave the particle which received it since it can neither be diffused away nor radiated away and it will not transfer into momentum again.

Heat from viscosity The motion energy lost from artificial viscosity is converted into internal energy through the following relation

$$\frac{d\varepsilon_i^{viscosity}}{dt} = \sum_{j=1}^{N} \frac{m_j}{2} \prod_{ij} v_{ij} \cdot \nabla W(r_{ij}, h_{ij}) \tag{33}$$

Heat from compression As a gas is compressed, the temperature increases and so does the pressure. This is due to the simple fact that work has been done to the gas in order to compress it. This work is converted into heat. If the gas later expands then the pressure inside it will do negative work and the gas will cool off again. This is described as

$$\frac{d\varepsilon_i^{pressure}}{dt} = \sum_{j=1}^{N} \frac{P_i}{\rho_i^2} m_j v_{ij} \cdot \nabla W(r_{ij}, h_{ij}) \tag{34}$$

Heat diffusion In[Bodenheimer et al. 2007] the following equation is given to describe heat diffusion

$$\frac{d\varepsilon_i}{dt} = -\sum_{j=1}^{N} m_j \frac{(k_j + k_i)(T_i - T_j)(\nabla W(r_{ij}, h)}{(\rho_i \rho_j)||r_{ij}^2||}$$

Here k is the heat conduction quotient and T is the absolute temperature. It is seen that the energy changes faster, the larger the heat conduction quotient, and the larger the difference in temperature, but it is also seen that even for pure aluminum with a relatively heat conduction quotient of around W/(mK), heat "moves" very slowly and it would take a long time for it to flow over the thousands of kilometers which are the scale of our simulations. It does however flow, and in some models it should not be ignored if for example an object is left for a very long time so that it can equalize its temperature, but it is irrelevant for us right now.

Heat radiation Heat radiation is described in detail in [Bodenheimer et al. 2007] and is not repeated here. Radiative transfer is very important for collapsing gas clouds but has little or no relevance for other models unless they should run for a very very long time. The solidification of the Earths crust would have to be described through radiation of heat but this is a very slow phenomenon which we do not deal with here.

5 Equation of state

The relation between the physical properties, density ρ, pressure P and temperature$T$ is described through what is known as an Equation Of State EOS. Knowing two of the properties lets us calculate the last one using the EOS for the particular material in question. A true EOS which accurately describes the relationship among the three variables over all values and all transitions is very complicated as it has to take into account various phase changes (changes between solid, fluid, gas and plasma) and for mixed materials it is even more complicated.

The current state of the art when talking about EOS is known as ANEOS which is an acronym for Analytical Equation Of State and which is a computer program in itself and not a single equation. This is often the EOS used in astrophysical simulations when the thermal processes are the describing factor in the simulations. In the simulations relevant to this paper, the actual EOS is known to have very little impact on the results since the controlling factor is mainly gravitational. We do still need some measure of how pressure, temperature and density relates to each other, but it does not have to be super accurate.

5.1 Polytropes

A polytrope (35) is a model describing how pressure relates to density for a fluid being compressed under its own gravity. Planets and stars and even spherical interstellar gas clouds are just that - spheres of fluid held together by their own gravity. The equation is derived by balancing the pressure with the gravitational force. The pressure force at all points inside the sphere should be in balance with the gravitational force.

$$P = K\rho^{(n+1)/n} \tag{35}$$

P is pressure, K is a tuning constant derived from the ideal gas equation, ρ is density and n is the polytropic index.

K is calculated from the following equation where N_A is Avogadros number, k_B is Boltzmann's constant, T is temperature in Kelvin and μ is the atomic weight of the particles. ρ is density and n is again the polytropic index.

$$K = \frac{N_A k_B T}{\mu}\rho^{-1/n} \tag{36}$$

The polytropic index essentially describes the compactness or how strongly pressure increases with increasing density. A low index of 0.5 to 1 is describing a neutron star well while an ”ordinary” star as the Sun may have an index of 3 while a gas giant like Jupiter is described by an index of 2. It should be noted that a planet such as the current Earth with a thin atmosphere, a solidified crust and a fluid center consisting of different materials with an inner iron core, is not described by a single polytrope.

The missing part in the above equations is temperature. A polytropic model describing both density, pressure and temperature is found in [Wada and

Kokubo]. It is tuned to be used in a simulation of the moon forming impact on proto Earth.

$$P = (\gamma - 1)\rho\varepsilon + C\left(\frac{n}{3} + 1 - \gamma\right)\rho^{n/3+1} \qquad (37)$$

Again P is pressure, ε is the internal energy and C is a constant defined to be 1 in this specific version of the EOS. γ is the specific heat capacity which is defined to be 1.01. The first term on the right hand side is just the ideal gas equation while the last term is describing the temperature independent pressure based on the solidity of the material.

The only thing missing now is to relate the viscosity of a fluid to the pressure, temperature and density, but this is generally not included in astrophysical simulations since the importance is neglectable. There is an article describing the relations in some detail in [Nature] but we will ignore the effect here.

The paper [Wada and Kokubo] cited previously is an actual astronomically relevant simulation of the moon forming impact which is carried out with this much simpler EOS. For this reason it seems perfectly reasonable that our much simpler simulations are done with an EOS which is no more complicated than the one used there.

We will use the above EOS throughout all experiments for simplicity and since research [Wada and Kokubo] has shown that the EOS is primarily responsible for the smaller details in a model.

5.2 Speed of sound

Many equations require the speed of sound c to be known. It depends on stiffness and mass of the medium in which the wave travels and this can be described in terms of change of pressure over change in density (38) which is again obtained from the EOS.

$$c^2 = \frac{\partial P}{\partial \rho} \qquad (38)$$

For an EOS describing P as a function of ρ, the speed of sound is given by differentiating this EOS with respect to ρ which gives the final form for the speed of sound

$$c^2 = \frac{\partial P}{\partial \rho} = \frac{n+1}{n} K \rho^{(n+1)/n-1} \qquad (39)$$

6 Gravity

While gravity can for smaller simulations be considered a simple force pulling evenly at every particle and where particle-particle gravitational pull is irrelevant, things are different when dealing with an astronomical simulation. Here the force of gravity is the overall governing force and it therefore needs to be dealt with in an appropriate way. There are two problems related to the force of gravity. Firstly it has infinite range and secondly it goes towards infinity for point masses when their distance goes towards zero.

The infinite gravity at zero distance is dealt with through gravitational smoothing in which the point gravity source is replaced with a source with a non zero radius. Obtaining the gravitational potential is then changed from a point sample to what is in essence an integral of the sources mass distribution over a volume.

The problem with the infinite range of gravity is dealt with exactly the opposite way around through hardening. Here a distributed collection of gravity sources will be approximated by a point source rather than as the sum of individual potentials.

Both methods rely on the same basic reasoning which is described in the following.

6.1 Infinite gravity at zero distance

The force of gravity is defined as (40) where G is the gravitational constant which is approximately 6.67E-11, m are the two point masses exerting a gravitational pull on each other and $|r_i - r_j|$ is the distance between the masses.

$$F_{ij} = G\frac{m_i m_j}{|r_i - r_j|^2} \tag{40}$$

It is evident that for a distance of zero $|r_i - r_j| = 0$, the force will be infinite. In the real world there will never be a distance of zero between any two particles, except possibly for two black holes colliding. In a SPH simulation there may well be radii of zero or other very small values and the problem needs to be addressed.

The first thing to note is that any SPH particle will represent a collection or real world particles and those particles are not assumed to be located at the exact same point in space. Just as the kernel smooths the contribution of a SPH particle the true particles are more or less smoothed, or rather distributed, in space. This means that the distance to *all* the real particles inside the SPH particle can never be zero from any position in space as seen in Figure 5

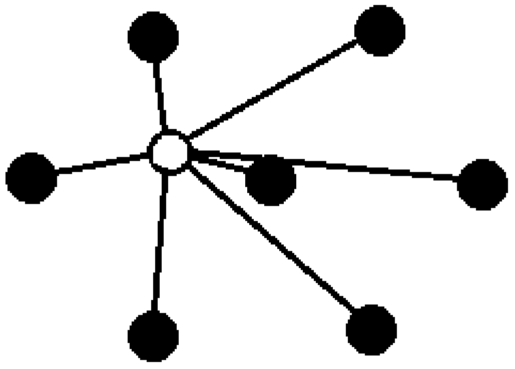
Figure 5: The solid particles are contained inside the same SPH particle but their real world positions are not the same. Therefore the hollow particle can never have a distance of zero from all of them at the same time

The point is that even though SPH particles are points and can be exactly on top of each other, the same is never true for a group of real particles ”contained” inside the SPH particles. The question is what the overall gravitational pull will be at a position inside a particle group such as that on Figure 5. At a glance we see that the hollow particle is closest to the tree particles pulling it to the left but at the same time there are four particles pulling it to the right and one of those is really close. For this setup the resulting force will be small even though the hollow particle is located close to the center of mass for the particle group.

Another way to think about this force not approaching infinity in the real world is a tunnel through the center of Earth. If an object is thrown into the hole, then it will initially be attracted with a force of approximately $m9.82m/s^2$ Newton, but unlike for point masses the gravitational pull does not increase as the object gets closer to the center of Earth. Instead it diminishes and becomes zero at the exact Earth center. This is because at this position there is an equal mass pulling it from all directions since all of Earth is ”above” the object now.

If the falling object is replaced with another planet and if Earth and the other planet can pass through each other without colliding, then we could perform the same experiment again. The gravitational pull is maximal just when the two planets touch each other and then it diminishes as they move their centers closer and closer to each other. When the centers coincide then the resulting force is again zero.

This shows us that real world objects, which are not point masses, should be dealt with differently than point masses since non point-mass objects have zero and not infinite gravity when their distance is zero.

Gravitational softening A softening method for gravity was proposed by Aarseth in 1963. It uses a softening factor ε which to some extent describes the radius of the smoothed masses, as opposed to the previous hard point masses.

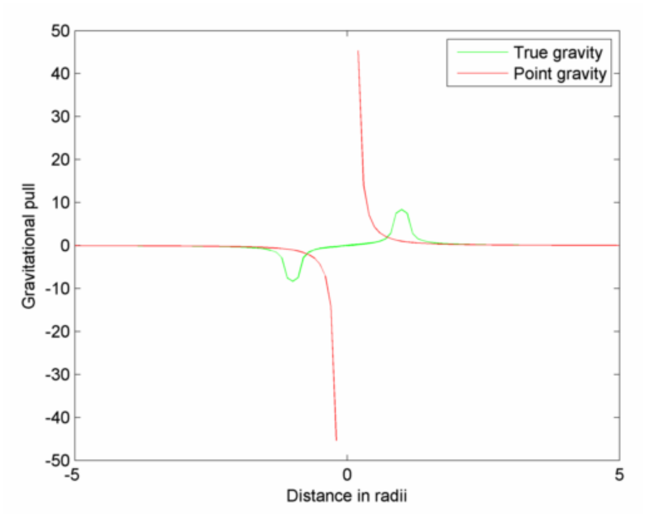

Figure 6: Gravity calculated both as an integral over an object and as a point source approximation

$$F_{ij} = \frac{Gm_i m_j(r_j - r_i)}{(\varepsilon^2 + |r_i - r_j|^2)^{3/2})} \tag{41}$$

This method has a maximum at a distance of $\frac{1}{\sqrt{2}}\varepsilon$ where the magnitude of the gravitational acceleration is given by (42)

$$a_j = \frac{2Gm_i}{3^{3/2}\varepsilon^2} \tag{42}$$

It should be noted that the expression for point particle gravity and for smoothed particles are identical at an infinite distance at which point the ε term vanishes which is also what we see for the 1D example in Figure 6. Here the object exerting the gravitational pull is defined by its density function as $\rho(x) = \frac{1}{2}$ for $-1 \leq x \leq 1$ which integrates to a mass of one over that interval. The gravitational pull felt at *position* is then $F(position) = \int_{-1}^{1} \frac{0.5}{(x-position)^2} dx$.

Smoothing factor and kernel smoothing length When this smoothed gravity is used in SPH simulations one generally set ε equal to the smoothing length h of the SPH kernels. This is mainly to ensure that the numerical resolution of the gravity force matches that of pressure, viscosity and what other forces might be included in the SPH simulation. In the case of collapsing clouds they must be same length since gravity and pressure forces are the two forces controlling collapse and expansion. if one force is given an advantage over the other, then that can be the single factor triggering or preventing the collapse.

6.2 Gravity's infinite range

While other forces in a SPH simulation are generally limited by the compact support of the kernels, the same can not be done for gravity. In astronomy gravity is the dominant force because of its infinite range and because it is always attractive. Gravity is the weakest of the four fundamental forces, those being gravity, electromagnetism and the weak and the strong nuclear force, but while the nuclear forces both have a very limited range gravity and electromagnetism have an infinite range. The reason the very strong force of electromagnetism is not overpowering gravity is that gravity always attracts masses to each other while electromagnetism both attracts and repels charged objects. On the large scale the attractive and repulsive force of electromagnetism cancels each other out which is not the case with gravity.

This infinite range and large influence of gravity means that an N-body simulation will have to perform $\frac{1}{2}N^2$ gravity force calculations to find the resulting force acting on every particle in a group of N. For large N this is a severe limitation.

Where there is no way around the N^2 complexity for the absolutely precise theoretical gravity calculations, one can perform accurate *enough* calculations at a complexity of $O(n\ log\ n)$ using a spatial partitioning such as an oct-tree for a 3D scene.

As it was noted for the smoothed gravity, a collection of gravity sources will sum to the same as a single gravity source when the distance is infinite. This means that at an infinite distance the resulting gravity from a group of gravity sources will

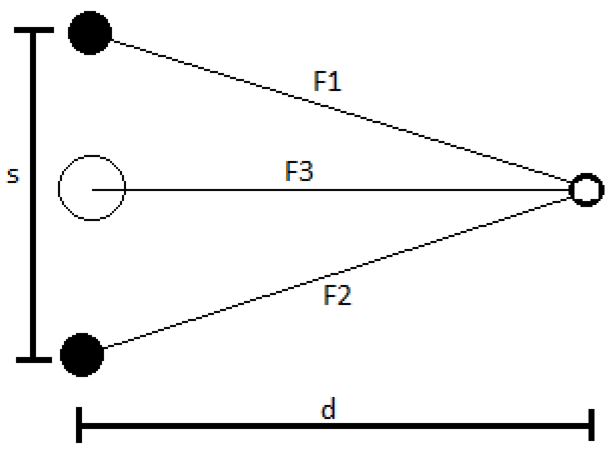

Figure 7: Gravity from the two solid objects is the sum of F1 and F2 but can be approximated by F3

be the same whether it is calculated as a sum of sources or as a single source located at the center of mass and with a mass being equal to the sum of all masses. When the distance is not infinite, the two methods will give different results and the simplification with a single point mass will be increasingly inaccurate as the distance shrinks or as the particles spread out.

As seen in Figure 7 the gravitational pull, at a position to the right, from the two solid objects can be calculated as the sum of each objects pull F1 and F2. This is a correct calculation. The resulting pull can also be *approximated* by F3 which is the pull from the hollow, virtual, object located at the center of mass of the solid objects and having a mass equal to the combined mass of the solid objects. It is a discretization since F3 is only approximately equal to F1+F2. There are two reasons for this. One is the fact that both F1 and F2 are 2D vectors and their vertical components are opposite to each other and therefore sum to zero. This leaves only the horizontal components which means that F1+F2 is always less than F3 calculated as if from a single gravitational source. The other is that the distance to the two particles is not equal to the distance to the center of mass. This means that even when the particles are located on a single line towards the position, for which the gravity is calculated, the approximated force will not be the same as the true force.

As the fraction s/d approaches zero, the error will approach zero as well since the relative vertical force will go towards zero and so will the relative difference in distance between the each particle and center of mass. This is the reasoning behind the Barnes-Hutt simulation method which uses the fact that the error, when approximating a group of gravity sources by a single combined source, will be very small for groups which are either very compact or very far away.

The Barnes Hutt method If 3D space is recursively partitioned into increasingly smaller cubes using an oct-tree until the cubes are either empty or contain only a single particle, as seen in Figure 8, then the nested cubes can be stored in a tree structure where each node contains either other nodes or a single particle. The largest node containing all others is designated the root node and the nodes deepest in the tree which contain only a single particle are designated leaf nodes.

The root node represents the entire 3D space of the scene and each child node the represents increasingly smaller parts of this space.

When building the tree, one have to partition a cell into potentially (in 3D) 8 new cells whenever it contains more than a single particle. This is an O(n log n) operation for the entire tree. Whenever a new node is created, the center of gravity and the combined mass of all particles contained within the bounds of the node, are stored. The size and location of each node is implicitly given by its parent node since its three sides will each be half the length of the parents sides.

When the tree is constructed, one, one can for any node in constant time O(1) acquire the combined gravitational pull from all particles within its bounds. This regardless of the number of particles actually contained herein, by looking only at the center of mass and combined mass inside the node. This changes the O(n^2) computational cost of calculating N-body gravity to O(n log n), but at

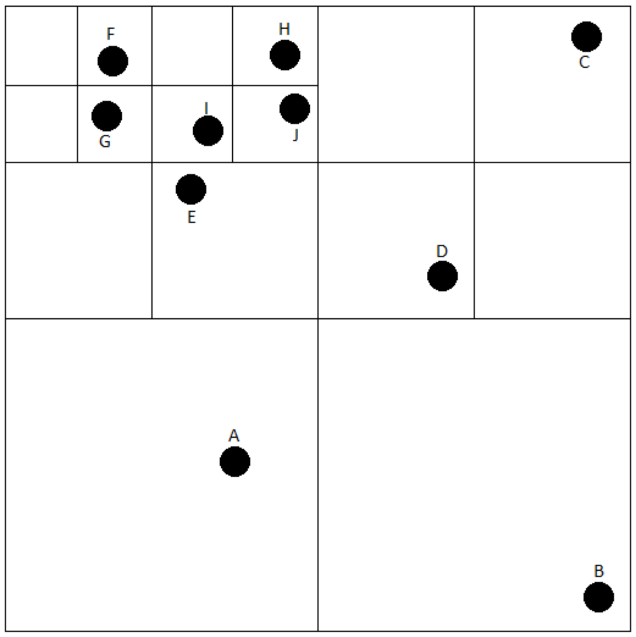

Figure 8: A 2D quad tree partitioning space until the cells are either empty or contain only one particle

a cost. As mentioned previously it is not correct to approximate the gravitational pull from a distributed mass as if it was concentrated in a single particle located at the center of mass. The error depends on how distributed the mass really is and how far away it is.

The way the Barnes-Hutt method deals with the point approximation error is by calculating a value for particle spread divided by distance. The smaller this value is, the smaller the error is. When the value approached zero, at infinite distance or when a number of particles truly are all located at the same point, then that error measure also vanishes. One then defines a maximal acceptable error measure *theta* and when calculating gravitational pull from a region of space, contained in a tree node, this measure is compared with the cube size and cube distance since the spread of a number of particles in a cube cannot exceed the size of the cube. Therefore the size of the cube places an upper bound on how spread out the particles in it can be.

When the error from point approximation exceeds the allowable value θ then the approximation is abandoned and instead the gravitational, point based, pull from that region is replaced with the sum of the pull from each of the eight child nodes. This recursion ends when the child node is small enough or far enough away or when it contains only a single particle at which point the point assumption is no longer inaccurate.

The following procedure is used to calculate the gravity at a position P from a node N. This is initially called with N=root.

```
1. function barnesHuttGravity(P, Node)
2. gravity = 0
3. foreach node N in Node
4. calculate distance d between N.CenterOfGravity and P
5. calculate error as N.sideLength / d
6. if error< $\theta$ then gravity = gravity + pointForce(P,N.CenterOfGravity, N.CombinedMass)
7. else gravity = gravity + barnesHuttGravity(P,N)
8. return gravity
```

To provide a better estimate for the error by approximating a group of particles, all inside the same node, one could calculate measures such as the variance for a normal distribution describing the particles and then use this instead of the cubes side lengths, but while it would indeed be a more correct estimate, using the side lengths do in fact give the upper bound on the error which is the important thing in this approximation. Calculating a variance for the particles, and there may be a lot of particles in the early nodes in the tree, is much more costly and the benefit from doing so is very little. It may save the gravity calculation from going down a few steps deeper into the tree here and there when the particles are tightly packed, but that does little good compared to the guaranteed extra cost when building the tree. For this reason the error is always calculated from only the distance to a cells center of mass and the cells side lengths.

A little perspective It should be noted that even though this method is considered an approximation used to speed up the computation, there are really never a situation where such approximations are not used. When simulating the gravity from the sun on Earth, both objects are approximated by point masses and we do not try to calculate the gravity from each individual particle in the Sun and Earth. Unless the calculations are done at a sub atomic level, approximations are used. Approximating a huge galaxy millions of light years away by a point will give a smaller error than doing the same approximation for the Moon a few hundred thousand kilometers away.

Another factor worth keeping in mind is that when a force is calculated as a sum of a very large number of smaller forces, then roundoff errors will be building up and eventually this supposedly correct summation of forces will result in an inaccurate value. This means that with finite precision numbers, there is a limit as to how good an accuracy is obtainable; with or without point mass approximations.

When an object is not a point but is blurred, then some parts of it will be closer to the observer and others will be farther away. Due to the fact that gravity depends on the distance squared, this means that gravity is more influenced by the part of the object which is closer than the object center, than it is by the part that is farther away. In other words; moving one half closer will increase gravity more than what is lost from pushing one half farther away. This means that the more an objects mass is distributed, the stronger the actual gravity will be. This is illustrated in Figure 9 where it is seen that the gravity contribution from the closest (to the right) parts of the object is larger than the contribution from the far side.

A perhaps better way to see that this is true is by considering an object with mass m at a distance r from an observer. We now split this object into two halves and move one a distance δ towards the observer and the other δ away from the observer. Initially the force F from the object was proportional to F_1 as seen in (47). After the move the force is now F_2. If we subtract the forces from each other then we see that F_2 is in fact larger than F_1 when it is assumed that δ is strictly less than r. Otherwise one object will be displaced to the other side of the observer.

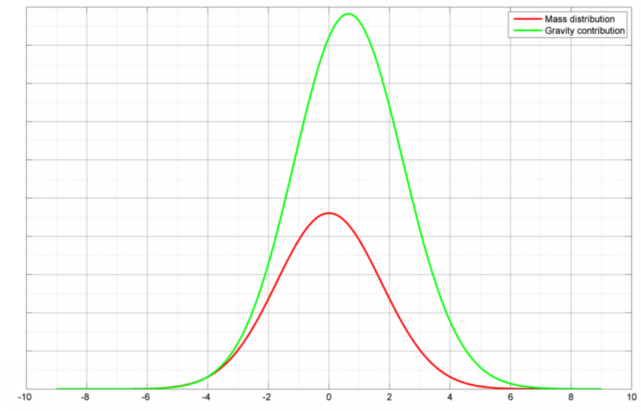

Figure 9: It is seen that the mass closest to the detector, at the right hand side, contributes more to the gravitational pull than the mass to the left.

$$F_1 = \frac{m}{r^2} \tag{43}$$

$$F_2 = \frac{\frac{1}{2}m}{(r+\delta)^2} + \frac{\frac{1}{2}m}{(r-\delta)^2} \tag{44}$$

$$F_2 - F_1 = \frac{\frac{1}{2}m}{(r+\delta)^2} + \frac{\frac{1}{2}m}{(r-\delta)^2} - \frac{m}{r^2} \tag{45}$$

$$F_2 - F_1 = \frac{-\delta^2 m(\delta^2 - 3r^2)}{r^2(\delta^2 - r^2)^2} \tag{46}$$

$$\tag{47}$$

7 Level of scale in time and space

Computer scientists often know SPH from simulating water or some other earthly fluid. In astronomy there are different issues to deal with. For

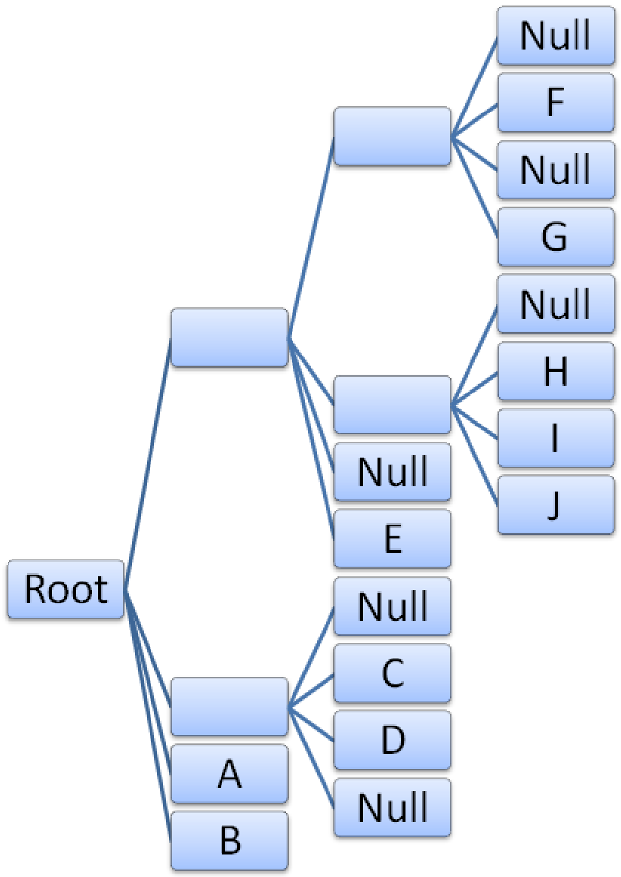

Figure 10: A tree structure describing the 2D spatial partitioning seen in Figure 8. The leftmost node is the root which contains all other nodes. A child node with no text is just a container of other tree nodes while a node with text is a leaf containing a single particle. Null nodes are nodes which neither contain other nodes nor contains particles. They are generally not created but are shown here for completeness.

one, the simulated material is generally compressible and can vary from a thin gas to the dense center inside a star. This will be a problem if the smoothing length is constant and the same for all particles. This can be solved using different smoothing lengths or using sink particles as described in section 7.3.

Another issue is that the forces acting on the particles will be vastly different in a simulation where some particles may be involved in a planetary collision while other particles are slowly orbiting the collision area. This will cause huge slow downs if all particles are advanced in time by the same step-sizes, which leads to the use of individual and even adaptive time-steps.

Yet another issue which is relevant in astronomy is the level of scale. Simulating the solar system with at least one particle for Earth, would require the larger bodies such as Jupiter and the sun to consist of an enormous number of particles if all particles have equal mass. This is generally solved by avoiding such scenarios since it is rare that it is required to simulate the large scale and the small scale simultaneously. The solution can instead be to use large scale simulations to find initial states for small scale simulations.

7.1 Individual adaptive smoothing lengths

As described in 3.3 there is an optimal number of support particles. Any smaller that this value and the solution becomes too coarsely sampled and inaccurate. Any more support particles and the solution becomes too smoothed and inaccurate in addition to it being more costly to evaluate.

When simulating an astronomical scene, the variation in density among particles can be enormous if the simulation tries to deal with more than a small subset of a problem. SPH models have been used to simulate the formation of solar systems as well as the collapse of interstellar gas clouds to the point where stars are formed in the denser regions.

Both problems span from very thin gas clouds to quite dense regions which contain enough material to form stars or planets. For this reason it seems obvious that if all particles have the same support radius then the particles in the denser regions will have a larger number of supporting neighbor particles than the particles in the thin regions have.

The solution is to initialize the support radius of a particle to a value which gives it a good number of support particles and then let this radius change over time along with the density. This lets a particle travel from a region with thin gas, having a large support radius, to a more dense region and have its support radius contract accordingly so the number of supporting neighbor particles remains almost constant over time as described in detail in [Bodenheimer et al. 2007].

Individual smoothing lengths Each particle i should have its own smoothing length h_i which is optimal for that particle. When the simulation starts, each particle will be initialized to the same smoothing length h. Then a relaxation of each h_i is performed until the lengths give a reasonable number of support particles. This is not a lengthy procedure and it is performed only once before the simulation starts.

Adjusting h_i runtime There are two simple ways to do this. If the change in density ρ_i over time is known, then the change in h_i can be based on that as (48).

$$\frac{dh_i}{dt} = -\frac{h_i}{3\rho_i}\frac{d\rho_i}{dt} \tag{48}$$

We will however not be using the change in density over time, but will instead calculate density through direct summation, so $\frac{d\rho}{dt}$ is not readily available. For this reason the other method (49) will be used. It updates smoothing length based on how well that length performed during the last time step.

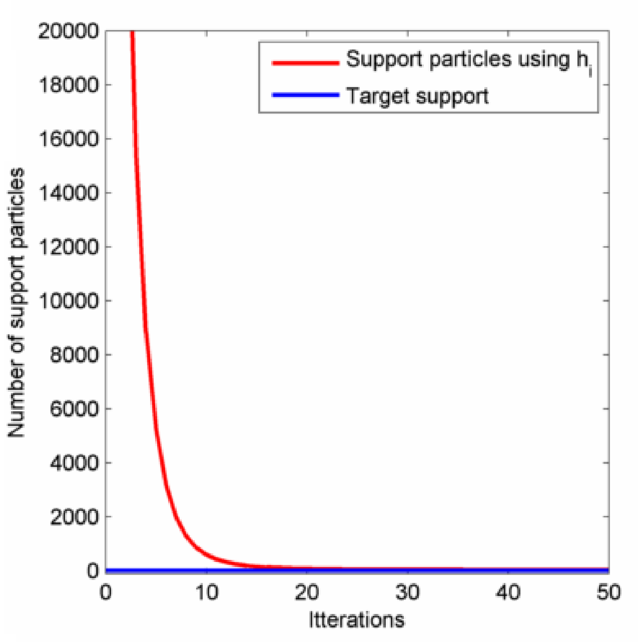

Figure 11: The convergence of the relaxation of the variable smoothing length

As seen in Figure 11 the relaxation method quickly changes h_i to match the desired number of support particles. The graph shows the convergence with an initial support which is a thousand times too large as can very well be the case when initializing the particles to a global smoothing length.

$$h_i^n = h_i^{n-1}\frac{1}{2}\left[1+\left(\frac{N_{target}}{N_i^{n-1}}\right)^{1/3}\right] \tag{49}$$

Hand tuning vs. relaxation Hand tuning a smoothing length get optimal number of interactions will result in the histogram shown in Figure 12. While most particle have the desired number of interactions, many do not. Further more, this will change over time as the particles move and the smoothing length will have to be constantly updated which is hard with hand tuning.

As an alternative the histogram of a simulation which dynamically adapts the smoothing lengths is displayed. This shows very much better results and the physics frame rate remained unchanged due to the simplicity of the method.

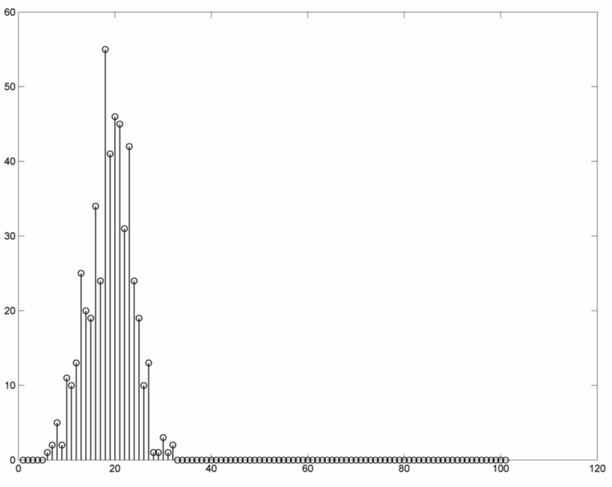

Figure 12: Histogram showing the number of interactions per particle with every particle having the same smoothing length. The target was 25

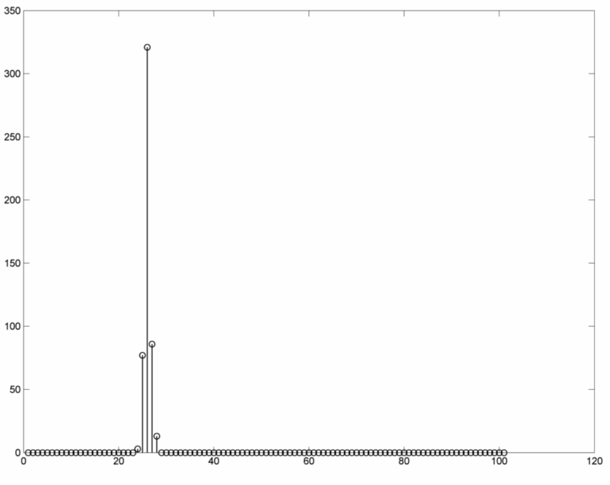

Figure 13: Histogram showing the number of interactions per particle with the particles having individually adapted smoothing lengths. The target was set to 25

It should be noted that some particles may eventually become very isolated and in that case, the relaxation scheme will expand their smoothing lengths far beyond what is reasonable. A very isolated particle should be isolated rather than smoothed beyond recognition. For this reason a maximal smoothing length will often make sence. Additionally it may be a good idea to use a minimal smoothing length to avoid numerical instabilities and/or very small time-steps when particles for one reason or another become very compressed.

Symmetric interactions One obvious problem with different smoothing lengths is that there may be a situation in which one particle has the other inside its smoothing radius while the opposite is not true. Newtons third law of motion states that whenever an object exerts a force F on another body then that other body exerts a force -F on the first body. It is all about action and reaction. If one particle has the other inside its support and the opposite is not the case, then the first particle will feel a force but the other particle will not. This is a problem if it is not fixed. A gentler version of this problem is two particles which are each in the others support radius but have different smoothing lengths and therefore scale the contribution of the other particle differently.

If one particle has the other in its support and the opposite is not the case, we still consider it a possibly interaction and will continue to the next step which is to establish the smoothing length and kernel weight.

For two particles in each others support radius but with different smoothing lengths, the symmetric way of handling this is to apply an average kernel to both particles. This can be done in several ways (50) to (53)

$$W(r_{ij},h_{ij}) = W(r_{ij},\frac{1}{2}(h_i+h_j)) \tag{50}$$

$$W(r_{ij},h_{ij}) = W(r_{ij},max(h_i,h_j)) \tag{51}$$

$$W(r_{ij},h_{ij}) = W(r_{ij},min(h_i,h_j)) \quad (52)$$

$$W(r_{ij},h_{ij}) = \frac{1}{2}(W(r_{ij},h_i)+W(r_{ji},h_j)) \quad (53)$$

According to [Liu and Liu 2003] there is no experimental results indicating if one of the methods is better and which one it would be. Obviously they are different. While (51) will tend to include more neighbors than the others and (52) will include fewer. We note that while (53) may intuitively seem more ”correct” it will also require double the number of kernel evaluations and since that is expensive, the method seems like the worse choice of the four.

We will use (51) for two reasons. First because it is cheap to evaluate and secondly because if any *one* particle ”sees” the other in its support, we are guaranteed that $|r_{ij}| \leq h_{ij}$ which means that the particles will both be inside the combined support radius. This is not always the case using (50)

When reading the earlier sections about kernels, where we were still not dealing with individual smoothing lengths, $W(r_{ij},h)$ should be replaced mentally with the symmetric kernel $W(r_{ij},h_{ij})$. In other words, the equations and the methods are the same - only the way the kernel is evaluated is slightly different when using individual smoothing lengths.

7.2 Individual adaptive time-steps

When numerically solving partial differential equations using an explicit time-stepping scheme, not all time-steps will lead to a convergent solution. A condition known as the Courant-Friederichs-Lewy condition or the CFL condition describes how the time steps must depend on the propagation of physical effects in the simulation. For a convergent solution at least the CFL condition must be satisfied. This does not ensure convergence in itself though and one generally have to satisfy additional conditions as well.

As an addition to the CFL condition (54) one generally define a force related maximal time-step (58) which ensures that particles currently undergoing violent accelerations will not take too large time steps. One may also add a velocity-based condition (59) which deals with particles already moving very fast.

Finally several methods for numerical integration allows the user to define a maximal allowable error over a time-step and then adjust the step-size to keep the error within that bound. This is supported by the commonly used Runge Kutta method or the Runge Kutta Feldberg. This actual integrator selected will be described in section 8.

The time-step used to advance the simulation will be the smallest of the ones calculated with the above mentioned methods.

7.2.1 CFL condition

The CFL condition states that the numerical domain of dependence must contain the physical domain of dependence. This can be rephrased as that the numerical propagation speed should be at least as great as the physical speed. In a grid based scheme with a distance between grid points of Δx and a time-step of Δt, information can travel one grid point every time step which is a velocity of $\Delta x/\Delta t$. If, as an example, a physical wave travels with a velocity of c, then the CFL condition states that $\Delta x/\Delta t \geq c$ since numerical information of the wave must propagate through the grid at least as fast as the wave itself.

Looking at $\Delta x/\Delta t$ it is clear that if the grid spacing Δx is small then the time-step Δt must be just as small to satisfy the CFL condition. Also if the physical wave speed c is large, then either the grid spacing must be large, or the time-steps must be small.

SPH is not a grid based scheme, so there is no fixed grid spacing. Instead there is a smoothing length which represents the ”distance” between particles. This again means that for the CFL condition to hold, a small smoothing length or a high velocity physical property will require a small time-step.

If the smoothing lengths are optimal, as described in section 7.1, then they will generally not be the same value. If additionally the particles have different velocities then both the "grid-size" and the physical signal velocity are different among the particles. This in turn means that the time-steps should also be different.

The physical signals traveling through the simulation can be pressure waves and moving particles. Neither one of those signals should travel faster than the numerical signals. This means that the CFL condition is a condition based on both particle motion v and sound speed c in the material being simulated.

Rewriting the original CFL condition to SPH for the time-steps gives

$$\Delta t_{\alpha} \leq \frac{h}{c} \tag{54}$$

which ensures that deformation waves through the medium will not travel faster than the numerical signals.

For models with artificial viscosity and fast moving particles the simple CFL condition is rewritten into (55) or (56). The first version assumes same step-size for all particles and then all particles step through time together while the other version is used for individual time-steps. We will implement both and make a comparison to demonstrate the benefits of time-steps which are limited to ensure stability but which are also optimized to individual needs.

$$\Delta t_{CFL} = min\left[\frac{h_i}{c_i + 0.6(\alpha c_i + \beta max_j|\mu_{ij}|}\right] \tag{55}$$

As described in section 4.1.2, α and β are constants defining the viscosity. c is the speed of sound and μ_{ij} is describing relative motion vectors between two particles relative to smoothing lengths and distance.

Calculating the CFL condition based time-step for particle i, rather than for all particles, is done as follows.

$$\Delta t_{CFL_i} = \frac{C_0 h_i}{h_i|\nabla \cdot v_i| + c_i + 1.2(\alpha c_i + \beta max_j|\mu_{ij}|)} \tag{56}$$

Here C_0 is the Courant number which is generally set to 0.3. The velocity divergence $\nabla \cdot v_i$ is added and is defined by (57). Note that $max_j|\mu_{ij}|$ is meant to be understood as : given a particle i, find the $|\mu_{ij}|$ for all particles j, which is maximal.

$$\nabla \cdot v = -\frac{1}{\rho_i}\sum_{j=1}^{N} m_j v_{ij} \nabla W(r_{ij}, h_{ij}) \tag{57}$$

7.2.2 Force-based limiter

The CFL condition is based on the current situation - namely the velocities and smoothing lengths. The force-based limiter is simply conserned with the acceleration of particles and is defined as

$$\Delta t_{accl} = min\left(\sqrt{\frac{h_i}{|a_i|}}\right) \tag{58}$$

7.2.3 Velocity-based limiter

The velocity-based limiter is very simple and it simply limits a particles step-size relative to its smoothing length. A particle should not move more than a fraction of its smoothing length every timestep since it would essentially move too far into the unknown. This condition is to some extent included in the viscosity-dependent CFL condition described earlier.

$$\Delta t_{v_i} = min\left(C_o \frac{h_i}{v_i}\right) \tag{59}$$

C_0 is again the Courant number, which is set to 0.3 here.

7.2.4 Handling individual time-steps

When each particle has a CFL condition-based time-step calculated, the next step could be to take the smallest time-step and advance all particles by that, but this would not be optimal. Often most of the particles are capable of taking large time-steps while a few particles are moving so fast, undergoing so strong accelerations or ate in so close proximity with other particles that they have very small time-steps. It would be silly to let all particles walk forward with baby steps because a few have to.

As described in [Saitoh and Makino], one solution could be to have a user defined largest time-step and then group the particles according to their time-steps as they best match a power of two division of the user defined time-step. If the user-defined largest time-step is 1s, then the power of two divisions will be 1s/2, 1s/4, 1s/8 etc. A particle with a time-step size of 0.3s would then be in between 1s/2 and 1s/4. The time-step size to choose should then be no greater than the step size determined through the CFL condition, so the particle would be assigned to 1s/4=0.25s which is the closest step not greater than 0.3s.

When this is done for all particles, the few particles which have very constrained time-steps will be in one group and the other particles which can take larger steps will be in a few groups with large step-sizes.

As seen in Figure 14 most of the particles have time-steps much larger than the small group with the small step-sizes. It would be wasteful, and unnessecary, to impose the smallest step-size on all particles.

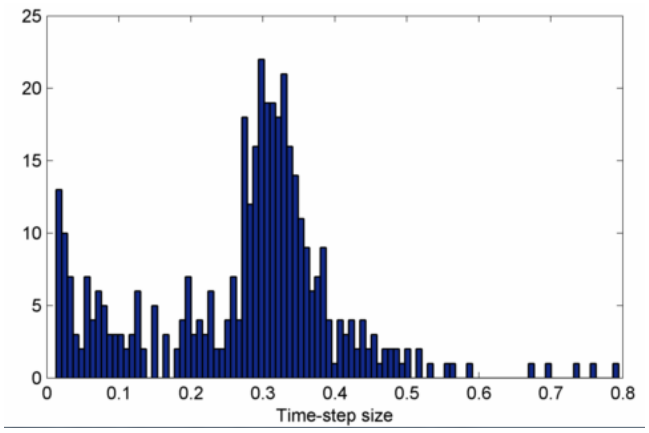

Figure 14: Histogram showing the distribution of safe time-step sizes among the particles

Particles stepping with different step-sizes In a single simulation step a couple of things happen. First all particle-particle interactions are found, next the forces from gravity and particle interactions are found and accumulated and then the particles are advanced in time to their new position. Finding interactions and calculating forces is the most expensive part. Advancing the particles is very cheap in comparison.

The way one can use the different time-step sizes to speed up computation is known as kick and drift. A particle is ”kicked” by the forces acting on it and then it ”drifts” along until kicked again. The particles with the large time-steps can drift along for a long time between kicks while the other particles need a lot of small kicks. The reasoning is that the force function $f(x,t)$ for the particles is quite constant for the large time-step particles while it changes a lot for the small time-step particles. Therefore integrating the force over a period of time requires many small sub-integrations for some particles while others can do it in one go.

Therefore the simulation starts with a kick to all particles and then they all drift to the next point in time which is $t_o + \Delta t_{smallest}$. Here the particles assigned to that step-size will locate interactions and calculate forces as well as CFL condition based time-steps and will then be kicked and drift until the next step is reached. Eventually time will have advanced enough for the particles with large time-steps to be updated and kicked as illustrated in Figure 15. It is seen that initially all four particles are kicked. Shortly after that, particle a is kicked again while the others drift. This means that over an entire user defined maximal time-step, a is kicked 8 times, b 4 times, c 2 times and d only

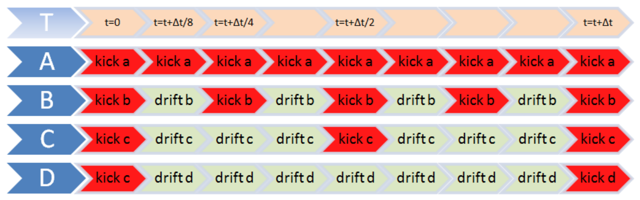

Figure 15: Four particles a,b,c and d have different time-steps and are kicked at different intervals

once. This is 15 kicks compared to the 32 kicks needed if all particles move with the total smallest time-step. Obviously the savings are greater the larger the different in step-size is.

Neighbors with different time-steps In astronomy there are situations where a group of particles will be accelerated violently in such a way that they will move far into regions with other particles which have much longer time-steps than desirable when encountering the fast moving intruding particles. one such example is a super nova explosion where the simulation contains a somewhat dense star surrounded by a thin cloud of gas. The time.steps will be smaller for particles in the star than for particles in the gas cloud, which is reasonable. When the star goes supernova, the star-particles will be pushed into the gas cloud with such a force that they will intrude far into the cloud before the cloud particles, due to their longer time-steps, even discover that something is going on. This is known as a Sedov problem, which has a known analytical solution, and the gas particles will generally have step-sizes a thousand times larger than the particles inside the star.

The solution proposed in [Saitoh and Makino] is to let step-size propagate through the particles due to neighbor relations, regardless of the forces, accelerations and velocities arising from those interactions. Briefly put, if a particle has a neighbor which has a much smaller time-step size than itself, then it should shrink its own step-size to a value not too different. Analysis of the Sedov problem have shown that good results are obtained with a limit on difference of 4.

If a particle a with a small time-step comes into contact with a long time-step particle b and the difference in step-size is larger than 4, then b red uses its step-size to no more than four times that of particle a.

in the simulations we perform here, this has little visible effect, but it is a thing which must be kept in mind for other more violent simulations and which should be considered for completeness.

7.3 Sink particles

In large scale simulations where particles are *supposed* to clump together and form objects with much higher density that the systems initial density, it is generally required to handle the high density areas in a special way. Such simulations are generally the collapse of interstellar gas clouds as it forms stars or the condensation of accretion disks around stars as they form planets.

A good example of a SPH simulation of star formation is seen in `http://www.youtube.com/watch?v=YbdwTwB8jtc` and described in [Bate 2008] where a gas cloud collapses under its own gravity and begins to expel new stars, shown as bright white particles, from the denser regions. White particles representing stars would ordinarily be a very tightly packed group of particles and this tight group would require tiny time-steps for accurate simulations as described in section 7.2.

The tight particle groups will describe the particles inside stars, but they will do so very poorly since the resolution is no where near good enough to describe the internals of a star. Further more, the groups will require much computational power and simulations tend to come to a grinding halt as soon as the first stars begin to form as described in [Bate et al. 1995] even when using individual time-steps. It is clear that as soon as the particles have clumped together to a star, their individual behavior is no longer relevant for a simulation of star formation in a collapsing cloud.

The white dots in the simulation are however not represented by a number of tightly packed particles but by a single so-called sink particle. This one particle represents the newly formed star and it holds the combined mass, momentum and internal energy of the particles which joined together to form the star. The sink particles interact with their surroundings through ordinary boundary conditions. Any particle coming within a limiting distance from the sink particle is absorbed and its mass, momentum and internal energy is added to the sink.

If for some reason a complete simulation ranging from the collapse of the interstellar cloud to stars and solar systems with planets and moons is desired, then the solution would be to run the large scale simulation first and at some time along the way stop and use the results as initial values for a simulation at a smaller scale. The density distribution around sink particles (stars) could be used as a starting point for a detailed simulation of an early solar system and from this point on the formation of planets could be observed.

An early example of large scale collapse can be seen at

`youtube.com/watch?v=gOk8Ja6FwsO`

where we have used a SPH N-body model to simulate the collapse of a cloud showing how individual areas collapse early on and how those condensations later merge to form larger objects. When the video zooms in one one such object it is clear that it consists of a very large number of very tightly packed particles and that the behavior of those particles is of little relevance to the large scale details. This shown model was a very early implementation of the simulator.

General idea As a region becomes very dense, the particles in the region will each have a higher density. At a certain limiting density a particle may be transformed into a sink particle with a certain accretion radius. All other particles within this radius will be swallowed by the sink particle and contribute with their mass, momentum and internal energy to the sink particle.

The idea is simple, but it is complex when it has to be implemented correctly so that the physical properties of the simulation will not change due to the presence of sink particles. Particles near the sink particles will be next to an artificial boundary (the accretion volume surrounding the sink) and particles should be able to move through the outer area of the accretion volume and still escape if only their velocity is high enough.

Because sink particles are more or less a requirement for large scale accurate simulations in astronomy, they are mentioned here. Because of the complexity of implementing accurate sinks, we will not touch the subject further. The paper [Bate 2008] has a very detailed introduction to the concept.

8 Time integration

Advancing the particle states from time-step to time-step is carried out using a numerical integration method known as the Beeman algorithm. It is derived from the family of Verlet integrators and is described in detail in [Beeman 1976] where it is also compared to a number of other integrators.

The Beeman algorithm (60) is a symplectic method which means that its phase space area is constant. Points in phase space are velocity and position which for 3D gives a 6D phase space. Having a constant area in phase space means that the energy will remain constant. This is a quite important property for any N-body simulation where energy growth can result in particles being expelled from the simulation and where loss of energy will let particles collide sooner than they should.

The algorithm comes in a one step form and in a two step form as a predictor-corrector for the situations where acceleration is dependent on velocity which is the case in out simulation having a velocity dependent viscosity.

$$\begin{aligned} x(t+\Delta t) &= x(t) + v(t)\Delta t + \frac{2}{3}a(t)\Delta t^2 - \frac{1}{6}a(t-\Delta t)\Delta t^2 + O(\Delta t^4) \\ v(t+\Delta t)^{pred} &= v(t) + \frac{3}{2}a(t)\Delta t - \frac{1}{2}a(t-\Delta t)\Delta t + O(\Delta t^3) \\ v(t+\Delta t) &= v(t) + \frac{1}{3}a(t+\Delta t)\Delta t + \frac{5}{6}a(t)\Delta t - \frac{1}{6}a(t-\Delta t)\Delta t + O(\Delta t^3) \end{aligned} \tag{60}$$

To take a time-step, the next position $x(t+\Delta t)$ is first calculated based on the current position $x(t)$, velocity $v(t)$ and acceleration $a(t)$ as well as the previous steps acceleration $a(t-\Delta t)$. The same is done for the next velocity $v(t+\Delta t)$. This results in a *predicted* velocity. Based on the the new position and that predicted velocity the next steps acceleration $a(x,v)$, which is a function of both position and velocity, is calculated. This information is then finally used to correct the velocity $v(t+\Delta t)$.

This results in a fourth order accuracy for position and third order accuracy for velocity, where ordinary Verlet integration only gives second order accuracy for the velocity.

Seeing how velocity is important for accurate calculation of momentum and how we have a highly relevant velocity dependent viscosity force, we have opted for the Beeman algorithm to get its higher accuracy while still being relatively cheap compared to other methods such as the popular RK4. That method even has the disadvantage that it looses energy over time which is undesirable.

9 Astronomical models

A few models are simulated and commented on in the following. It should be noted that they all generally have unrealistic scaling of forces or time. This is to ensure that something interesting will happen quickly. It is the same "interesting" thing as would happen in the real world but much faster.

Take the formation of planets in the accretion disk surrounding a star. In reality this would take several million years for something interesting to happen. Over this period of time the disk would rotate millions of times around the star. If we want to see a planet within 5 minutes, then we would have to simulate some million revolutions of the disk which would amount to several thousand revolutions per second. That is simply not possible since each revolution would in turn require a fair amount of time-steps to be just a little bit realistic.

The solution is for us to speed up accretion and let the planets form over the duration of just a small number of revolutions. This requires a very high viscosity to make the particles lump together and therefore the simulation is no longer accurate though it still demonstrates the actual astronomical model of orbiting dust forming planets.

9.1 Collapsing gas cloud

An interstellar gas cloud may eventually reach a critical mass after which point it collapse under its own gravity. Pressure forces will attempt to make the cloud expand and gravity will try to crush the cloud into a point.

`youtube.com/watch?v=a-qTdkvbYes`

The initial cloud will have *some* internal movement and that movement will invariably give *some* rotational momentum. Otherwise every motion of every particle in the cloud would have to be balanced by the other particles exactly.

During a collapse the rotational momentum remains constant. For any particle moving relative to the clouds center of mass, it will contribute some angular momentum

$$L = r \times m\, v \tag{61}$$

where r is the vector from the center of mass to the particle, m is the particles mass and v is the particles motion vector. $\times$ is the cross product.

When r shrinks during the collapse, and L should remain constant, v has to increase. This means that as the cloud collapses, it will begin to rotate faster and faster. This will in turn add "centrifugal" force to the particles trying to pull them out again and expand the cloud. At some point in the collapse there will be balance between gravity on one side trying to compress the cloud and pressure and ”centrifugal” force trying to expand the cloud.

In addition to this, the densest central region may form a star at which point that star will begin to radiate energy and charged particles outwards. This ”sun wind” will contribute to the forces trying to expand the cloud again.

9.1.1 Jeans mass

For a gas cloud to be stabile and not expand or collapse it must be in hydrostatic equilibrium which is defined as

$$\frac{dP}{dr} = -\frac{G\rho M}{r^2} \tag{62}$$

where P is the pressure, G is the gravitational constant, ρ is the density, M is the mas of the entire cloud and r is the radius of the cloud.

This is similar to the derivation of the polytrope model section 5.1 in that it balances the pressure gradient with the gravitational gradient.

If the right hand side of the equation outweighs the left hand side, then the cloud will collapse. This can happen if either the total mass is very large, the density is very high or is the radius is very small[2] or if the pressure is small. The low pressure can be caused by a low temperature.

Jeans mass is defined as

$$M_J = \left(\frac{5R_gT}{2G\mu}\right)^{3/2}\left(\frac{4\pi\rho}{3}\right)^{-1/2} \tag{63}$$

and is derived in [Bodenheimer et al. 2007]. Its derivation is based on the speed at which particles would fall towards the center of mass, thereby compressing the cloud, compared to the time it takes pressure to build up and oppose this compression. The ”reacting time” for the pressure is related to the time it takes a compression wave to move through the gas. This again depends on the speed of sound and the size of the cloud. If gravity acts faster than pressure then the collapse is unavoidable.

[2]The limiting radius is related to another term the Jeans length

The cloud, or parts of it, will collapse under a number of different circumstances which can be induced by various processes. These processes are briefly explained but it was too time consuming to implement (even a simple one) a working simulation of all the complexities of cloud collapses.

Low temperature A hot cloud will radiate heat and if it is not too opaque then this heat energy will be lost from the cloud and the temperature will drop and a collapse can start. During the collapse heat energy will be generated from compression and some internal friction but it will still be radiated away quickly enough for the cloud to not increase its temperature. This is a an isothermic collapse until the point where the cloud becomes opaque enough to hold on to its heat.

High density A motion of the gas in an interstellar cloud is believed to be controlled by magnetic turbulence. Charged gas moves which causes magnetic fields which moves the charged gas. This will produce areas of varying density and the high density areas may collapse. Another cause of high density may be nearby stars and especially supernovas and black holes radiating particles from the poles of their accretion disks [McKinney 2006]. This radiation will create a ”wind” which sweeps gas with it and pushes the cloud together and increases its density thus initiating star formation in the collapsing regions.

9.1.2 Simulation

A plot showing the initial motion of 1000 particles with random positions in a 2D square and with

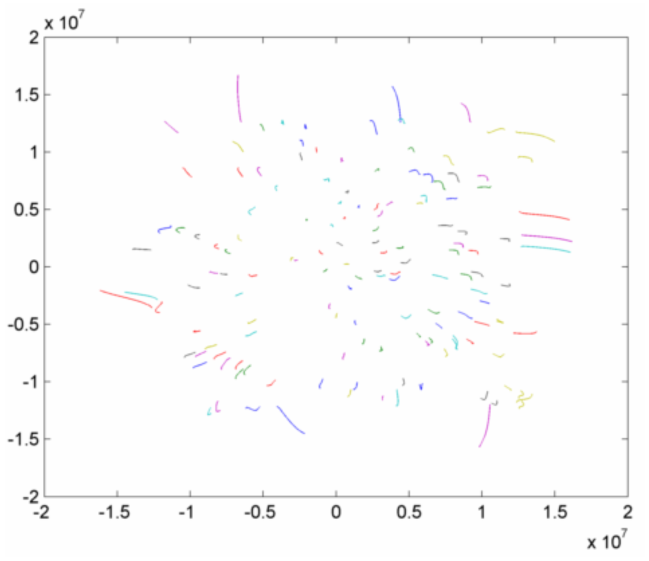

Figure 16: The initial moments of the collapse of a cloud

small random motion vectors is seen in Figure 16. The lines a traces of the particles over a few initial time-steps. During the traced time, the particles are already beginning to move towards the center of mass. The cloud was created in an unstable state where the pressure was in no way enough to carry the cloud.

Some time later Figure 17 the particles are all moving quite close to the center of mass but still there is not much rotation, but some time later the central part of the collapse shows a clear increase in rotation Figure 18.

The final rotating clump of particles as rendered in the simulator is seen in Figure 19. The colors are density colored using a hot colormap. The central part has a high density colored yellow due to the number of particles pressing in on it, just as one would expect for a core, but just outside the core is a lower density region colored red. This is caused by the rotation. The entire structure rotates with no differential motion due to the artificial viscosity which means that there can be "centrifugal" forces pulling the particles out, but *without being in balance* with the gravity as ordinarily seen in orbital motion.

The "centrifugal" force (64) grows with distance

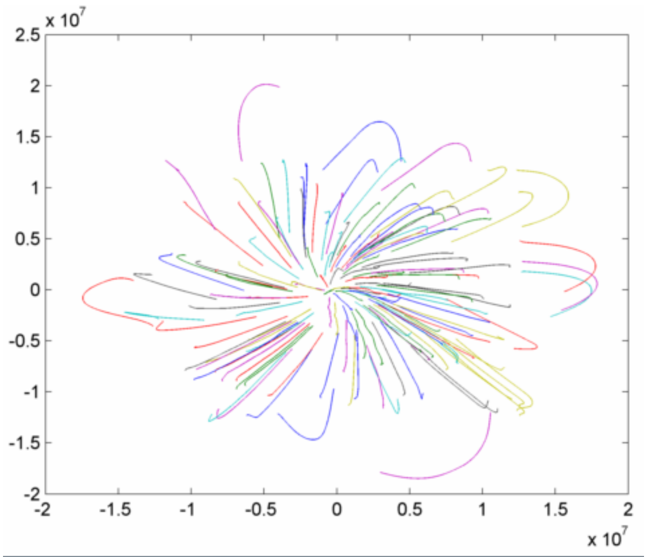

Figure 17: Particle traces during the first part of a collapse

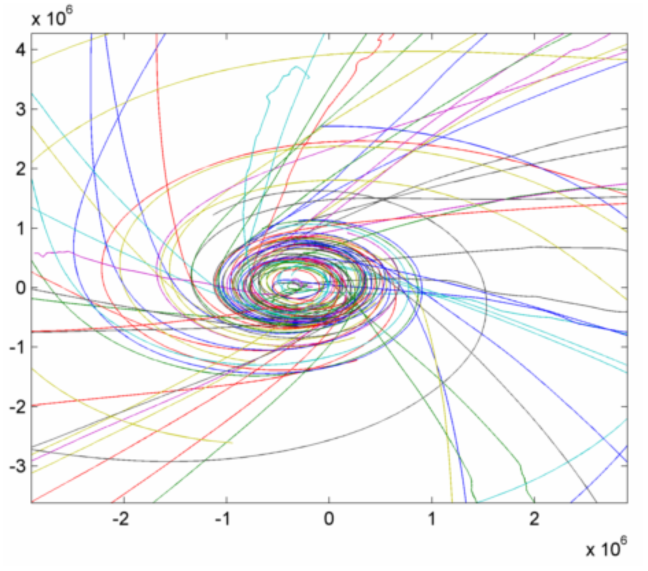

Figure 18: Zoom of particle traces during the last part of a collapse

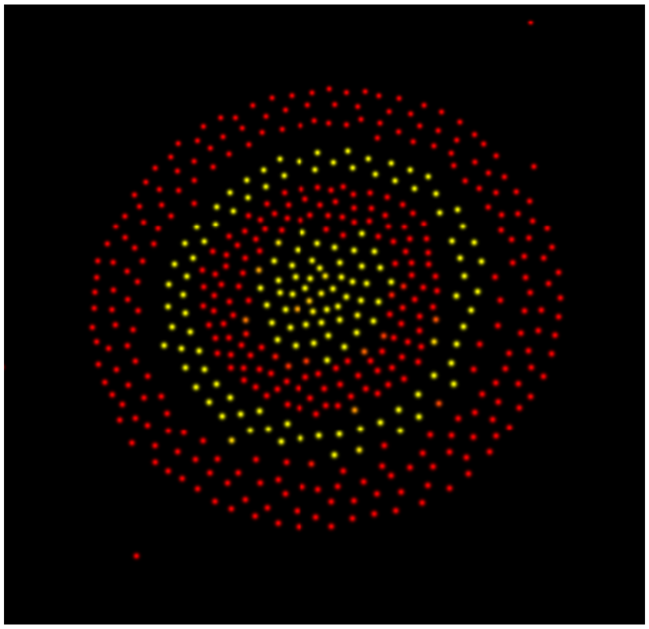

Figure 19: Zoom of particle traces during the last part of a collapse

from the center in a rigid rotating object

$$F_{centrifugal} = m\frac{v^2}{r} \tag{64}$$

since v increases with radius, but at the same time the gravity shrinks with the distance from the center. This is the reason there is a low density zone and an actually almost empty zone in the object. These are the unstable points where the forces will pull particles away. Had this been visualized as a surface plot, there would have been hills and valleys circling the center.

9.2 Origin of the Moon

The Earth Moon system have been a mystery for a long time. The Moon is very large compared to other moons and their planets. Its mass is less than a hundred times that of Earth while for comparison the two Mars moons Phoebos and Deimos are 10 million and 100 million times less than that of Mars. In other words: the Moon is so big that it seems likely that it is not just a small object being caught my the gravity of Earth. It would have been very difficult for the Earth to have caught an object of that size. The Moon is the solar systems fifth largest satellite while the Earth is a dwarf in comparison with the gas giants Jupiter, Saturn, Uranus and Neptun.

Adding to that is the fact that the angular momentum of the Earth Moon system is very much larger than that of other planet moon systems. Finally Earth has a large iron core while the Moon has next to no iron.

Prior to the discovery of the iron poor moon, the theory was that since it was so unlikely that the Moon was captured by Earth, the Earth and Moon had to have come from the same collection of dust early in the creation of the solar system and were formed close to each other. Discovering that the moon has little iron, this theory became unlikely as well.

9.2.1 The giant impact hypothesis

In 1975 Dr. William K. Hartmann and Dr. Donald R. Davis suggested the theory of a giant impact as the source of the moon [Hartmann and Davis 1975]. In the early solar system there is thought to have been a great many partially formed planets known as protoplanets. Collisions among these protoplanets would have been common and though these collisions larger planets would build.

The theory is that late in the formation of the Earth it was impacted by another protoplanet about the size of Mars. The impact should have been off center so that the system consisting of the two objects had a considerable angular momentum. A large portion of the Earths crust would have been blasted off and have entered an orbiting disk of remains from the impact. This disk would be iron poor and from it the Moon would eventually form.

Many computer simulations have later tested the theory. Both finite difference methods and SPH methods have been used in the simulations and they agree that under the right circumstances such an impact could have taken part in the creation of the moon. Some like [Wada and Kokubo] consider how complex the equation of state during and after the impact need for the moon to form as quickly as it is assumed it has. Others [Canup 2004] consider the sensitivity of the Moon formation relative to the impactor size, velocity, direction, proto Earths rotation etc. The findings of the above two articles were that a complex EOS was not required and that the impactor theory required quite specific conditions to create the Earth-Moon system we know.

9.2.2 Simulation

A video showing one of the simulations with too low impact velocity

`youtube.com/watch?v=eZkhbWzepM4`

Two planets were created by letting an unordered collection of particles collapse under their own gravity with a dampened integrator as suggested in [Bodenheimer et al. 2007]. One had 1000 particles and a total mass of 5E+24 kg which is approximately the mass of the current Earth. The other had 500 particles and a total mass of 6.4E+24 kg which is approximately the mass of Mars. The equation of state was taken from [Wada and Kokubo].

The impact velocity was ranging from 4000 m/s to 16.000 m/s and the impact location varied from 0 degrees, equator, to 80 degrees, near the pole of proto Earth.

The simulations showed that it was very difficult to make the impactor both throw out enough material and throw it far enough away for it to condensate to form a Moon. When material was actually thrown out, it was not easy to do it with just enough speed to not fall right back and to not entirely escape proto Earth.

Several simulations with variable settings was performed and images from there are shown below. Figure 20 shows the very early impact between a red impactor and a blue proto Earth while Figure 21 shows the same impact shortly after. The history unfolds on Figure 22 to Figure 25.

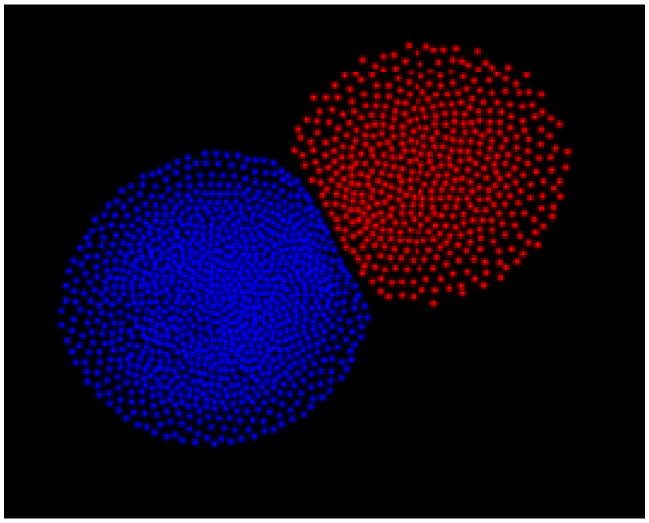

Figure 20: Red impactor just touching proto Earth around t=700s

A slow impact 7000 m/s

Too fast impact 16.000 m/s An example of an impact which was too fast. This resulted in material being blasted away with a velocity higher than the escape velocity. This does form an object, but that object has no intentions of orbiting the proto Earth any time soon. It moves away never to return leaving just a little material behind on proto Earth.

A midspeed impact 10.000 m/s This impact type was the most promising. It did actually produce an orbiting moon. The moon was too close to proto Earth though so it was eventually absorbed back into the heavier object, but it did several full revolutions before this. An interesting thing about this simulation was that the initial impact blasted material into a brief orbit after which the ”moon” fell back on earth in a secondary impact. This impact moved material out into orbit again and it was this material which formed the fully orbiting moon. This is exactly as the theory [Hartmann and Davis 1975] describes the moon forming impact - with the slight exception that the moon in that model was more long lived.

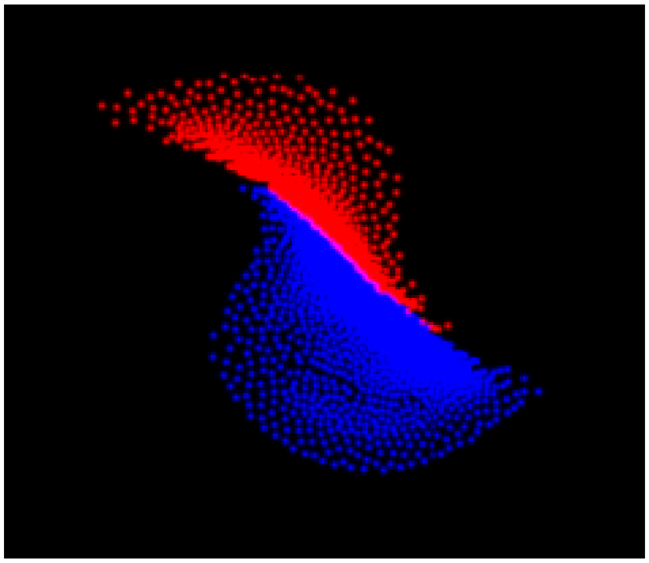
Figure 21: Impactor has flattened one side of proto Earth arround t=1050s

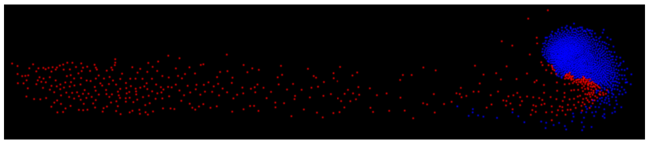
Figure 22: Material has been blown far away around t=10550s. The material is still just a loose spray

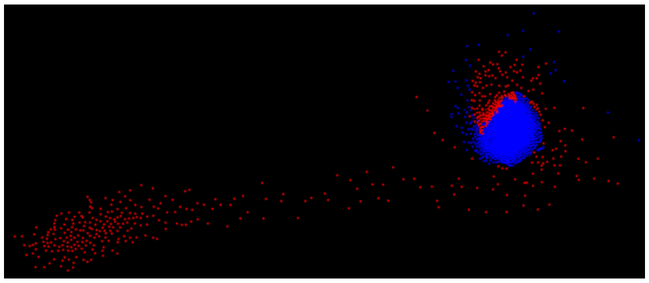
Figure 23: At t=19040s the blasted material starts to gather

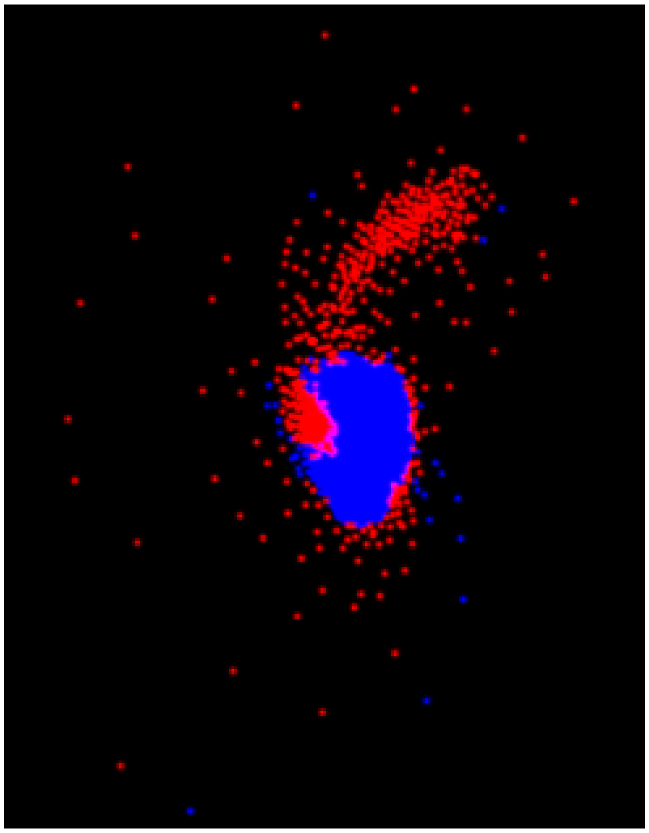
Figure 24: The material formed ”the moon” too close to proto earth and on a close pass at t=45300s the ”moon” is devoured by Earth

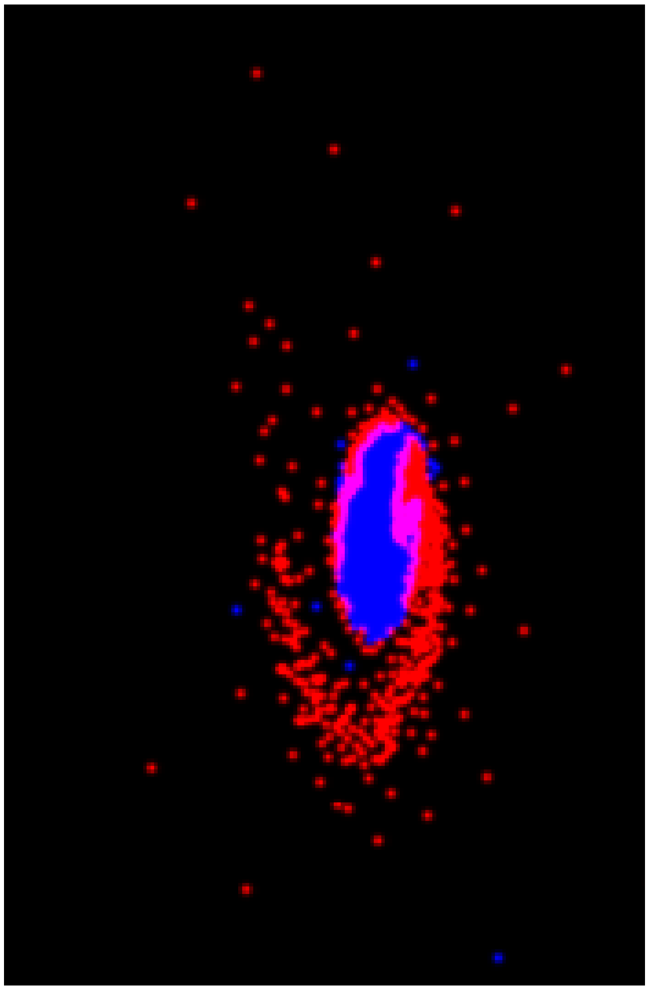
Figure 25: Many rotations later, the centrifugal force and gravity still battles over the failed moon. This forms a short spiral of material as it eventually falls back to proto Earth

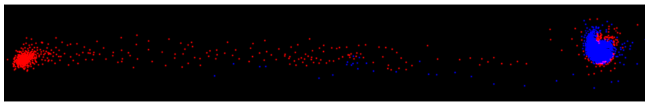
Figure 26: Too fast an impact blasts material far away

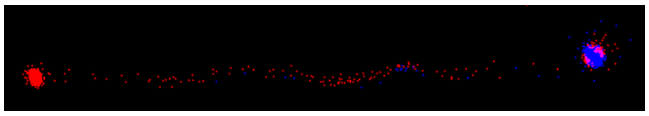
Figure 27: A while later the material from the previous figure is even farther away and moving above escape velocity

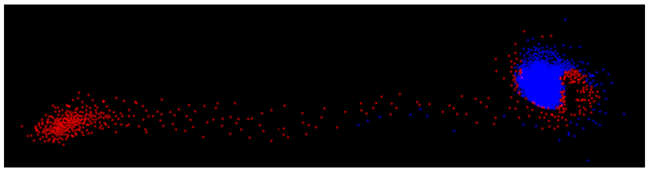
Figure 28: A moon starts to form. A matter bridge still connects it to proto earth, but it is seen that both objects have begun collecting the material

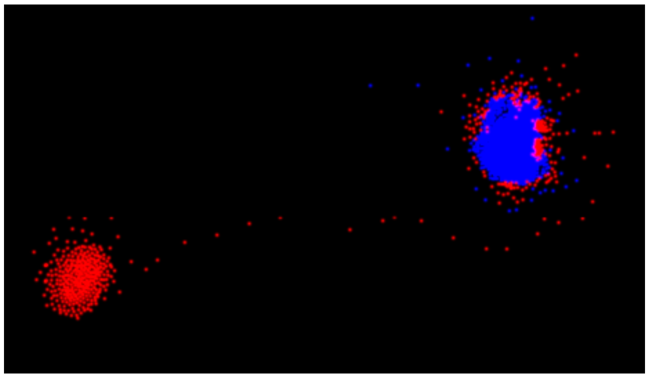
Figure 29: The moon is now more clearly defined and the matter bridge has almost been absorbed

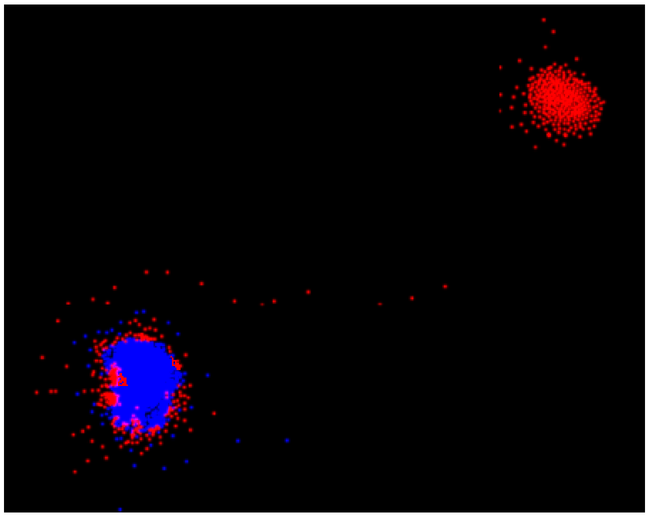
Figure 30: The moon has now completed several orbits around proto earth, but it is clearly too close and a short time after it is broken apart and absorbed

The conclusion is that, as the research has already showed, not all giant impacts will produce a big orbiting moon at a sufficient distance from Earth to not be broken apart by tidal forces. It is interesting that we did in fact come this close to creating a moon, even if it was too big, too close and too short lived. It should be noted though that given the mass of the individual particles of approximately 5E+21 kg, the Moon should consist of only 14 particles. Distinguishing real moons of 14 particles from random garbage due to noise is not feasible. Had we used more particles and run this simulation off line, then the resolution could have been good enough to accurately point out the real particle groups representing the Moon.

9.3 Roche limit

The Roche limit is the distance within where moons orbiting a planet cannot form by clumping together orbiting material and it is the distance within where an already formed moon will break up and be spread out to form an orbiting disk of material. It may equally well be planets orbiting a star or even stars orbiting a heavier star or black hole, but to avoid confusion we will call the orbiting object a moon and the central object a planet.

In 1848 the astronomer Edouard Roche originally calculated this limit. It is derived from the forces acting on a particle in the moon which are the gravitational pull from its neighboring moon particles F_m, the pull from the planet F_p and the centrifugal force F_c experienced due to the moons rotation around the planet. A moon with a fast self rotation will also experience centrifugal force due to its own spin which will counteract F_m Here we will ignore this factor.

Consider two points on the surface of the moon. P_{near} which is on the side of the moon towards the planet and P_{far} which is on the opposite of the moon. Both experience a gravitational pull towards the planet, towards the moon center and a centrifugal force away from the planet. If the forces experienced are not the same and if the moon is soft, it will be deformed from its spherical shape. The forces are written as accelerations below.

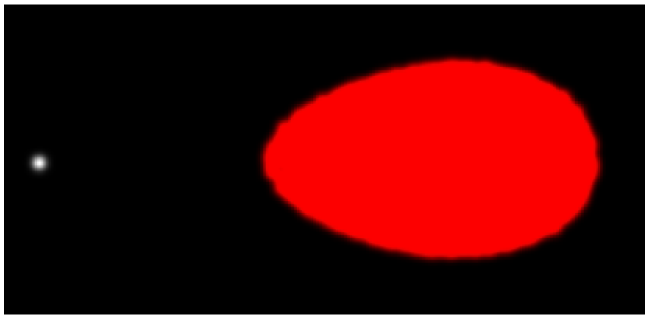

Figure 31: A fluid moon with zero initial motion has been accelerated towards a planet and due to the difference in gravity felt on the near and the far side of the moon, ir has been stretched into a tear shape.

$$a_p = \frac{GM_{planet}}{r_{orbit}^2} \tag{65}$$

$$a_m = \frac{GM_{moon}}{r_{moon}^2} \tag{66}$$

$$a_c = \frac{v_{orbit}^2}{r} \tag{67}$$

The total difference in acceleration for P_{near} and P_{far} will be the acceleration tearing the moon apart. As the moon begins to deform the elongated shape, as seen on Figure 31, it will cause the effect to accelerate till the point where the breakup is total.

The above assumptions require the moon to be fluid with nothing keeping it together other than its own gravity. Rigid moons are held together by electro chemical bonds which are much stronger than gravity. This allows rock moons to exist inside the Roche limit even though they can not form there.

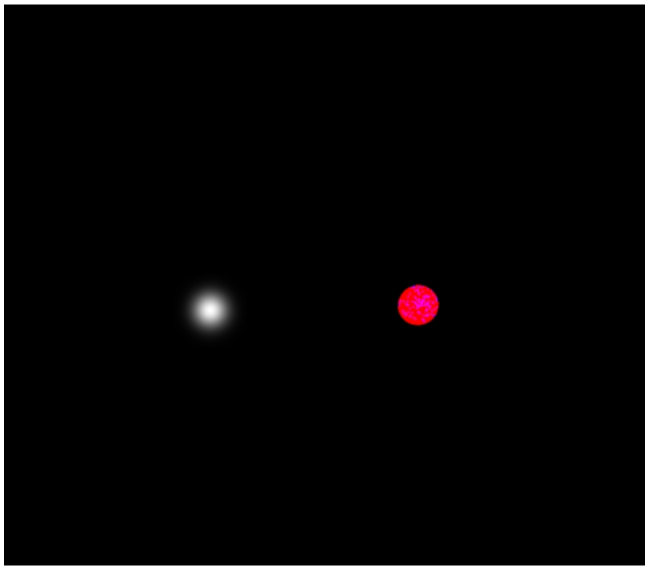
Figure 32: A spherical moon orbits a planet. The planet is not rendered to scale

9.4 Simulation

A video showing a zero momentum fluid moon accelerated towards a planet and stretched due to differential acceleration

`youtube.com/watch?v=hKlmz45YPis`

Another video shows the entire breakup of a moon

`youtube.com/watch?v=WQanorVXjhQ`.

A 100 particle spherical moon with the mass and radius of the Moon was created and positioned very close to a point sized planet having the mass of Earth. The moon was given an initial motion vector to put it into an orbit much too close to the planet.

Figure 32 shows the initial position and shape of the moon while Figure 33 shows the visibly deformed moon approximately 0.2 orbital period in the simulation. In Figure 34 the moon has broken almost entirely apart. The situation worsens in Figure 35 and after three orbital periods, the moon is forming a spiral pattern around the planet in Figure 36. Then finally in Figure 37

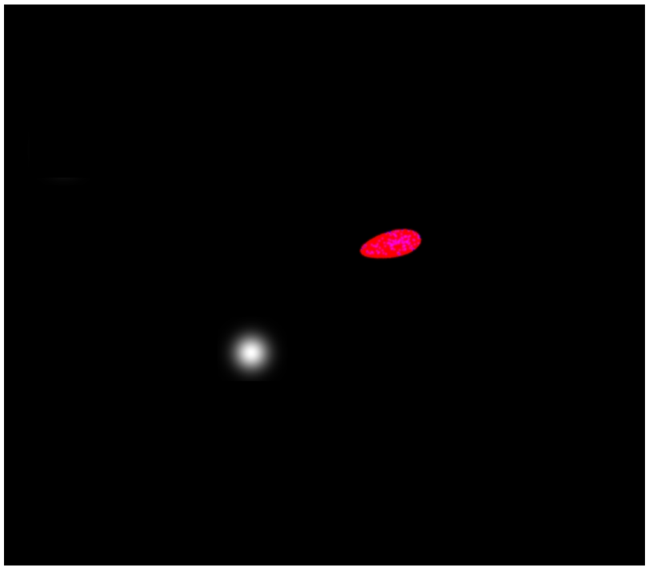
Figure 33: Already a fraction of an orbital period later, the moon has visibly deformed

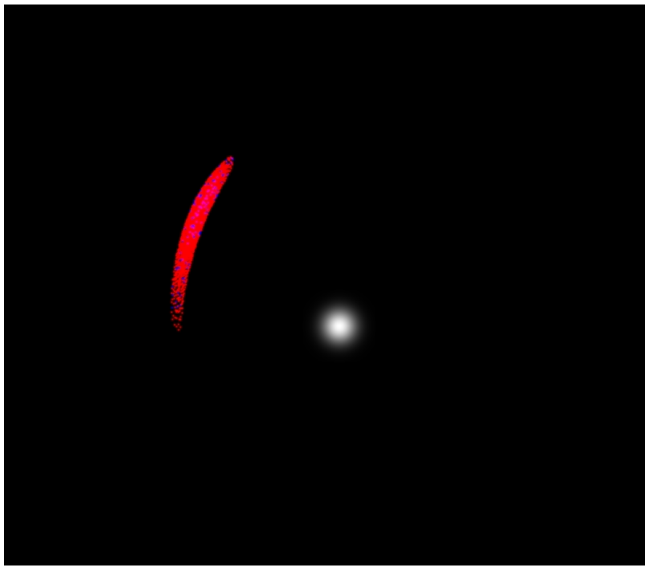
Figure 34: Here the moon has broken almost entirely apart

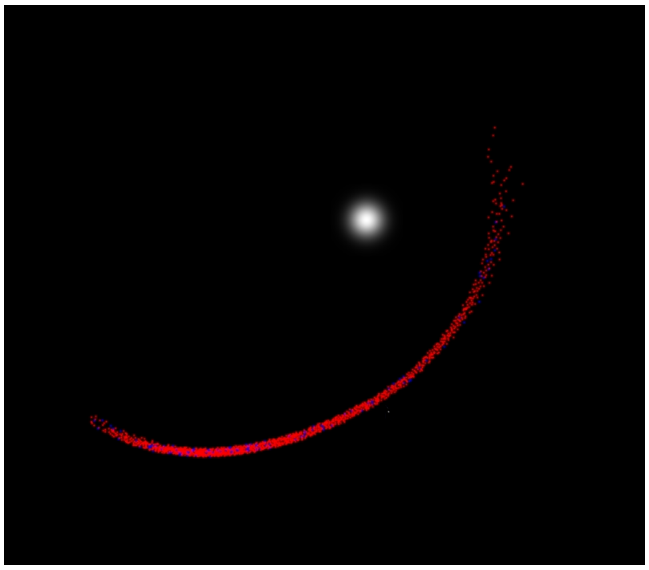

Figure 35: An even more broken up moon

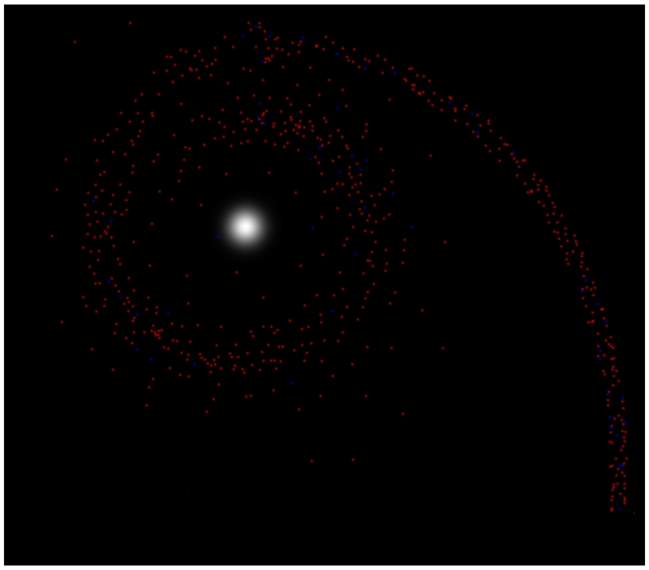

Figure 36: The remains of the moon is spiraling around the planet

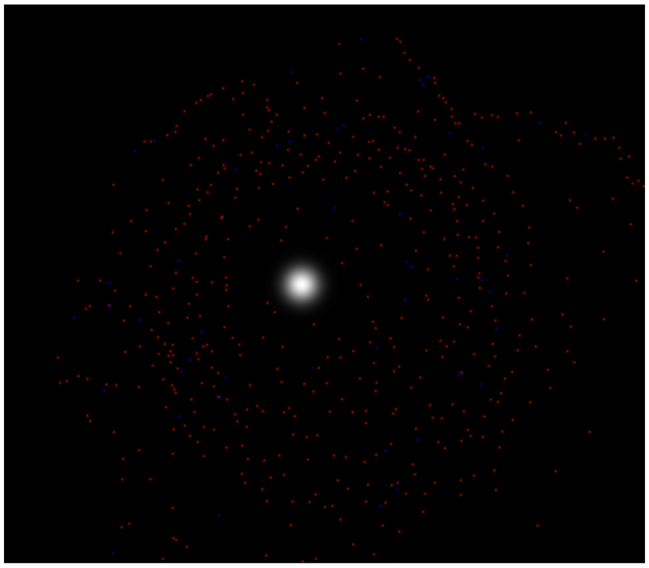

Figure 37: The moon is completely gone and instead the planet has a disk of fragmented orbiting material

9.4.1 Transference of momentum

A very interesting thing about this moon breaking up is that the fragmented material had a very high viscosity. The viscosity would not be high in the real world, but it was artificially enhanced to give a larger shear viscosity among particles with different orbital velocities. This means that even over the very short time simulated here, we are able to see another physical effect from differentially rotating disks of orbiting material. That is the transference of momentum from the fast rotating inner parts to the slower rotating outer parts. This means that rather than having the different parts of the material disk rotate with their own optimal velocity, the inner parts will be slowed down and will have a decaying orbit eventually making them impact the planet. At the same time the outer parts will be accelerated into a higher orbit.

The images showing the breakup of the moon are all kept to same scale to ensure that comparisons can be made. It is clear when comparing Figure 32, Figure 35 and Figure 37 that some parts of the material are in much lower orbits while other parts are in much higher orbits. It can also be seen in the video mentioned previously of the event, that the inner most particles are absorbed by the planet while outer particles keep gaining distance from the planet.

10 The simulator

A simulator was implemented using using the language C# and the graphics API XNA.

The application is highly parallelized and can run any user defined number of threads which lets it easily scale to different systems with different numbers of processing cores. The test system was an Intel Quad Core2 Q9550 running at 2.83GHz and having 3 GB usable memory. The operating system was windows 7 professional 32 bit.

10.1 What is missing

The evolution of internal energy over time was not implemented. Neither was individual time-steps, though adaptive time stepping is included in the simulator. All other elements described in this text were implemented.

10.2 Parallel efficiency

The application was tested with from 1 to 100 threads on 1000 particles and the performance scaled in an almost linear way with the number of threads up to the point where there were more threads than cores. From this point, the calculation speed remained constant. This indicates that the parallelization was quite efficient though not all the code was parallelized and it shows that there are very few waits among the threads. It also indicates that the application can be built to use a large number of threads and it will not negatively affect performance to have this number of threads even though there are fewer cores. This means that a given build can adapt well to future systems with more cores than the test system. The only requirement is that the number of particles must be a multiple of the number of threads.

Number of threads	physics frame-rate
1	7
2	12
3	16
4	20
10	20
100	20

10.3 Spatial partitioning

The number of calculated gravity sources is plotted over 80 time-steps of a simulation of a collapsing particle cloud consisting of 1000 particles. It is evident that the smaller the value for θ the more particle-particle gravity is calculated while a larger number tends to approximate far-away particle groups as a single source.

A value for θ of 0.0 ensures that no approximation will be used. Comparing the number of gravity sources with a value of 0.5 shows Figure 38 that the workload is approximately ten times smaller. Having used a larger cloud would have shown an even greater saving using the approximations.

10.4 Conservation of momentum

Regretfully, the total linear momentum of the simulation was not conserved as seen in Figure 39. The green circles show momentum of a system using a time-step of 100 seconds and the solid red line show the same using a time-step of 200 seconds. Both grow in a monotonous way throughout the sampling. Both grow with almost the same amount indicating that this is not a time-step dependent error.

A system with 1000 particles was run for 200 time-steps and momentum was logged. The system was using a Barnes Hutt gravity approximation with $\theta = 0.5$. Initially there was zero linear and angular momentum.

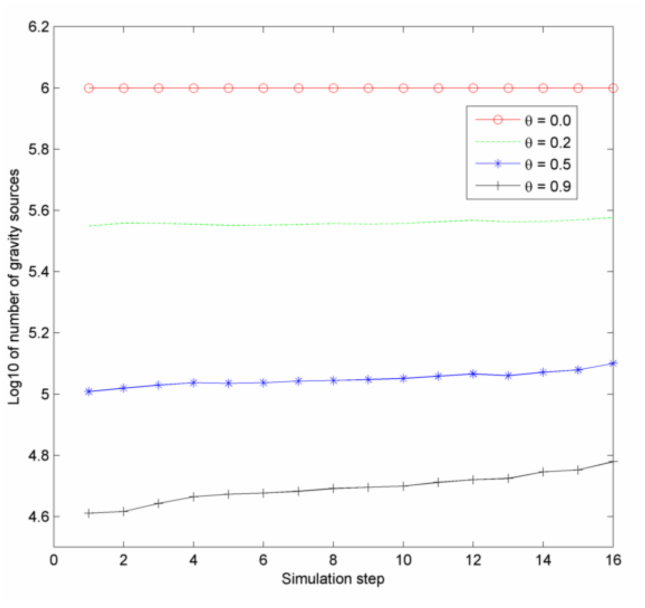

Figure 38: Plot of calculated gravity sources for different values of θ

Note that momentum, mass times velocity, is here scaled down to momentum per kg mass in the simulation. The particles used here each weighed 2E+28 kg so obviously even the slightest error in velocity would change the momentum enormously and comparisons with lighter systems would be impossible.

The bright side to this momentum problem is however the black stared line at the bottom. It is the result of running the same simulation with the large time-step of 200 seconds, but this time without the gravity approximation. Now momentum is almost constant. A closeup is seen in Figure 40 where it is evident that the error is not not growing and it is also seen that per kg simulation mass it is very tiny.

The conclusion is that the approximation not only results in a slightly wrong gravity force, as first assumed and as described in the literature. It also results in a violation of the law of conservation of momentum. The explanation can be described using an example with three particles A,B and C. B and C may be located close to each other some distance from A. When A queries the tree for gravity the B-C group is far enough away for it to be ap-

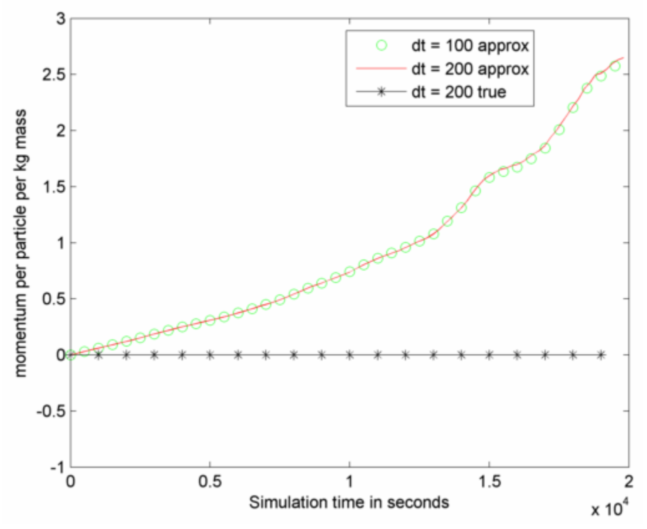

Figure 39: Total linear momentum per kg mass in the simulation with and without a Barnes Hutt approximation

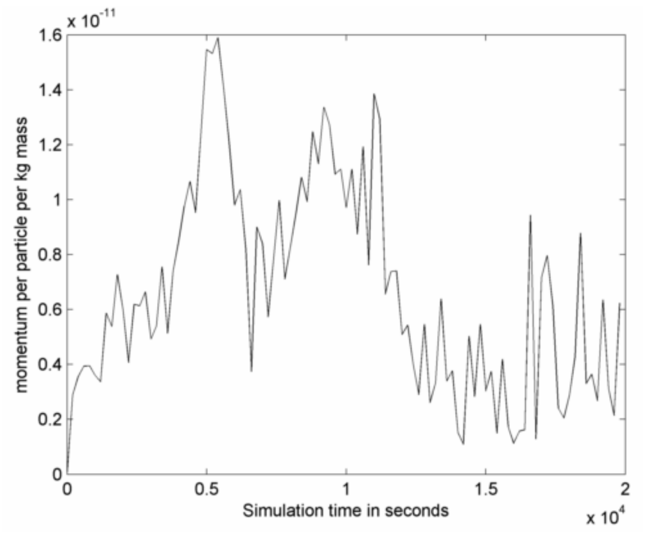

Figure 40: Total linear momentum per kg mass in the simulation using true gravity

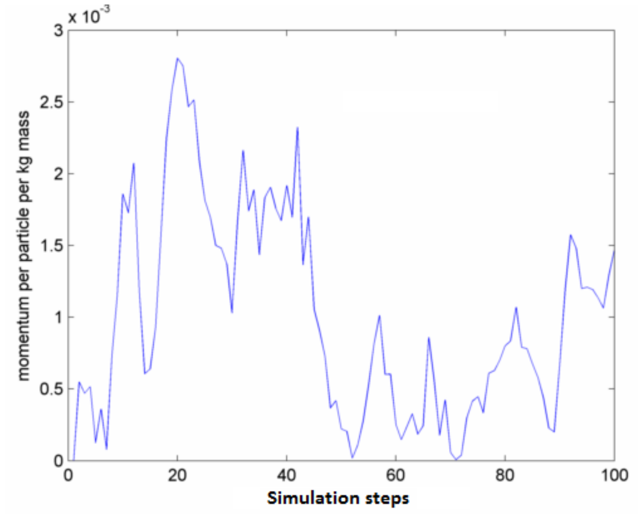

Figure 41: Total angular momentum per kg mass in the simulation using true gravity

proximated by their center of mass at a distance equal to their average distance from A. This results in one force F_A^{BC} felt by A from BC. When on the other hand B queries the tree for gravity, C is close and should not be approximated and A is far but is isolated and is not approximated (it is not grouped close to any other particles). Therefore B received the forces F_B^A and F_B^C. The same happens for C which received F_C^A and F_C^B. The problem is now that the force felt by A from BC is not the same as the force they feel from A! In other words $F_A^{BC} \neq F_B^A + F_C^A$ since the Barnes Hutt approximation means that $F_A^{BC} \approx F_A^B + F_A^C$.

This means that there is a lack of symmetry in the forces between particles and this in turn means that momentum is not conserved.

Angular momentum is affected in exactly the same way by the approximation and as soon as true gravity is used, the angular momentum is also conserved. Not quite as perfectly as linear momentum, but quite well none the less. In Figure 41 is shown a scaled plot of angular momentum which is small and not growing.

10.4.1 A possible solution

Only after discovering that the Barnes Hutt method did not conserve momentum did we look for alternative methods. One which we found is described in[Dehnen 2000]. The general idea is to not only have particle-cell and particle-particle interactions as in Barnes Hutt, but also have cell-cell interactions for sufficiently compact and distant cells. Ironic enough this idea was considered early on in the project as an optimization to regular Barnes Hutt, even though we had not seen it described anywhere. It was considered only for its possible better speed and not for conservation of momentum though, since we were at that time not aware that naturally regular Barnes Hutt was not conservative.

10.5 Simulator evaluation

The simulator handles systems with 2000 particles at generally 20 physics steps per second. The rendering is constant at 60 FPS. For most purposes, the 20 physics steps per second is reasonable, but for some very high velocity collisions the adaptive time-step will obviously be very small which makes the simulation seem slow. Since rendering is always fast and since the camera can be freely moved in 3D the simulations are still interesting even for very small time-steps since it can be observed from various views. Simulations such as the moon forming impact ran at a speed where it took around 3 minutes for the moon to form and do a few revolutions around proto Earth. That is quite reasonable for demonstration purposes.

11 Future work

The next obvious step would be to parallelize the simulator even further. The speedup from one core to four cores was convincing so the upscaling to many more threads running on either a cluster system or on a Graphics Processing Unit would most likely let the simulator run at interactive rates with a much higher particle count. There already exist particle system simulations running on graphics processors and they are very impressive, but most of those simulators are pure N-body simulators or ordinary particle systems simulating earthly fluids - not the combination of the two.

It would also be interesting to implement the alternative tree method and validate that it is in fact faster and even more importantly that it does conserve momentum.

References

BATE, M. R., BONNELL, I. A., AND PRICE, N. M. 1995. Modelling accretion in protobinary systems.

BATE, M. 2008. Stellar, brown dwarf, and multiple star properties from hydrodynamical simulations of star cluster formation. Tech. Rep. arXiv:0811.0163, Nov. Comments: Accepted by MNRAS, 28 pages, 26 figures. Animations available at http://www.astro.ex.ac.uk/people/mbate/.

BEEMAN, D. 1976. Some multistep methods for use in molecular dynamics calculations. *Journal of Computational Physics 20*, 2, 130 – 139.

BODENHEIMER, P., LAUGHLIN, G., ROZYCZKA, M., AND YORKE, H. 2007. *Numerical methods in astrophysics - an introduction*. Taylor and Francis.

CANUP, R. M. 2004. Simulations of a late lunar-forming impact. *Icarus 168*, 2, 433 – 456.

DEHNEN, W. 2000. A Very Fast and Momentum-Conserving Tree Code.

DURDA, D. D. 2003. Comparing results of sph/n-body impact simulations using both solid and rubble-pile target asteroids. *American Astronomical Society, DPS meeting Bulletin of the American Astronomical Society, Vol. 38, p.582*.

FLEBBE, O., MÜNZEL, S., RIFFERT, H., AND HEROLD, H. 1994. Physical Viscosity with SPH. *Memorie della Societa Astronomica Italiana 65*, 1049–+.

GINGOLD, R. A. M., AND J., J. 1977. Smoothed particle hydrodynamic: theory and application to non-spherical stars.

HARTMANN, W. K., AND DAVIS, D. R. 1975. Satellite-sized planetesimals and lunar origin. *Icarus 24* (Apr.), 504–514.

HOOVER, W. G. 2003. Impact fractures in solids. *American Astronomical Society, DPS meeting Bulletin of the American Astronomical Society, Vol. 38, p.582*.

LIU, AND LIU. 2003. *Smoothed Particle hydrodynamics - a meshfree particle method*. World Scientific.

MCKINNEY, J. C. 2006. General relativistic magnetohydrodynamic simulations of jet formation and large-scale propagation from black hole accretion systems. Tech. Rep. astro-ph/0603045, Mar.

MONAGHAN, J. 1989.

1986. *Origin of the Moon*. Lunar and Planetary Institute.

MÜLLER, M., CHARYPAR, D., AND GROSS, M. 2003. Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 154–159.

NATURE. Viscosity and eos. `http://www.nature.com/nature/journal/v392/n6678/abs/392805a0.html`.

SAITOH, T., AND MAKINO, J. A necessary condition for individual time-steps in sph simulations.

TOMASSINI, B. M. 2002. Equation of state data for iron at pressures beyond 10 mbar.

WADA, K., AND KOKUBO, E. High-resolution simulations of a moon-forming impact and post-impact evolution.

WILLIAMS, P. R., CHURCHES, D. K., AND NELSON, A. H. 2002. The Dangers of Constraining the Kernel Radius in SPH Galaxy Simulations. In *Disks of Galaxies: Kinematics, Dynamics and Peturbations*, E. Athanassoula, A. Bosma, & R. Mujica, Ed., vol. 275 of *Astronomical Society of the Pacific Conference Series*, 452–+.