Selected methods in N-body simulations

Aleksy Bałaziński

May 8, 2025

Contents

1 Introduction

The dominant force over large distances is the gravitational force. The force exerted on a body with mass m_2 at the point $\mathbf{x}_2$ by a body with mass m_1 located at $\mathbf{x}_1$ can be expressed by the relation

$$\mathbf{F} = -G\frac{m_1 m_2}{|\mathbf{x}_{21}|^3}\mathbf{x}_{21}$$

where G is the gravitational constant $6.674 \times 10^{-11}\mathrm{m}^3\mathrm{kg}^{-1}\mathrm{s}^{-2}$ and $\mathbf{x}_{21} = \mathbf{x}_2 - \mathbf{x}_1$. Therefore, the evolution of a system of N bodies is described by N equations

$$\ddot{\mathbf{x}}_i = -G\sum_{j\neq i}\frac{m_j}{|\mathbf{x}_{ij}|^3}\mathbf{x}_{ij}. \tag{1}$$

for each $i = 1, \ldots, N$. Direct application of Equation 1 is the basis of the so-called *particle-particle* method. The method is characterized by $O(N^2)$ time complexity (more precisely, it requires $(N-1)N/2$ operations if Newton's 3rd law is used in the computation). Assuming that 100ns are required to perform the floating-point operations under the summation symbol, $N = 30,000$, and 150 iterations, the simulation would take approximately 2 hours to complete. Therefore, it is evident that more efficient algorithms are needed to make simulations of this scale feasible.

The *particle-mesh* (PM) technique, introduced around 1985 by Hockney and Eastwood, was an early improvement over the PP method. In the PM approach, the space is divided into a rectangular grid (or mesh) of cells. Each cell is assigned a portion of the mass of nearby particles, creating a density distribution $\rho(\mathbf{x})$. The relation between the density and gravitational potential ϕ, in the form of Poisson's equation

$$\nabla^2 \phi = 4\pi G \rho, \tag{2}$$

is then used to obtain the potential at each cell center. The gravitational field $\mathbf{g}$ can then be calculated as $\mathbf{g} = -\nabla\phi$. Since $\mathbf{g}$ equals the acceleration due to gravity, we get $\ddot{\mathbf{x}}_i = \mathbf{g}(\mathbf{x}_i)$.

The drawback of the PM method is its poor modeling of forces over short distances. Eastwood and Hockney proposed a remedy for this problem: the *particle-particle-particle-mesh* method (or P^3M in short). In the P^3M method, the force on the i-th particle is split into two components: *short-range* and *long-range* force. The long-range force is calculated using the PM method, whereas the short-range force can be found by direct summation of the forces due to nearby particles.

The computational complexity of the PM and P^3M methods depends on the implementation of the potential solver used to calculate ϕ from Equation 2. For instance, if a fast Fourier transform is used, then the complexity of the PM algorithm is $O(N + N_g^3 \log N_g)$, where N_g is the number of cells in a single dimension of the grid (note it is linear in N). For the P^3M method, the worst-case scenario happens when all particles are clustered closely together, which causes the short-range $O(N^2)$ correction part to become dominant.

2 Particle-mesh method

The particle-mesh method can be described as the following sequence of four steps:

1. Assign masses to mesh points,
2. Solve the field equation (Equation 2) on the mesh,
3. Calculate the field strength at mesh-points,
4. Find forces applied to individual particles by interpolation.

In this section, each of these steps will be described in more detail.

2.1 Mass assignment

The specifics of assigning mass from particles to mesh points depend on the density profile (or *shape*) associated with the particles. In general, the particles need not be represented as idealized dimensionless points; indeed, it is possible to construct a hierarchy of shapes, where each successive member covers a larger number of mesh points and whose application leads to smaller numerical errors.

An infinite hierarchy of shapes with this property, as described by Hockney and Eastwood in [3], can be generated by successive convolutions with the "top-hat" function Π, defined as

$$\Pi(x) = \begin{cases} 1, & |x| < \frac{1}{2} \\ \frac{1}{2}, & |x| = 1 \\ 0, & \text{otherwise.} \end{cases}$$

The three most popular assignment schemes that hail from this family (and the ones implemented in our program) are the *nearest grid point* (NGP), *cloud in cell* (CIC), and *triangular shaped cloud* (TSC) schemes, with shapes S given by

$$S_{\mathrm{NGP}} = \delta(x), \quad S_{\mathrm{CIC}} = \delta(x) * \frac{1}{H}\Pi\left(\frac{x}{H}\right) = \frac{1}{H}\Pi\left(\frac{x}{H}\right), \quad S_{\mathrm{TSC}} = \frac{1}{H}\Pi\left(\frac{x}{H}\right) * \frac{1}{H}\Pi\left(\frac{x}{H}\right) = \frac{1}{H}\Lambda\left(\frac{x}{H}\right),$$

where Λ is the triangle function

$$\Lambda(x) = \begin{cases} 1 - |x|, & |x| < 1 \\ 0, & \text{otherwise.} \end{cases}$$

In the one-dimensional case, the fraction of mass W_p assigned to mesh-point p from particle at position x is given by

$$W(x - x_p) = W_p(x) = \int_{x_p - H/2}^{x_p + H/2} S(x' - x)dx'.$$

A simple rule for relating the assignment function W defined above with shape S can be found by noticing that

$$W(x) = \int_{-H/2}^{H/2} S(x' - x)dx' = \int_{-\infty}^{\infty} \Pi\left(\frac{x'}{H}\right) S(x' - x)dx' = \Pi\left(\frac{x}{H}\right) * S(x).$$

This implies that

$$W_{\text{NGP}}(x) = \Pi\left(\frac{x}{H}\right), \quad W_{\text{CIC}}(x) = \Lambda\left(\frac{x}{H}\right), \quad W_{\text{TSC}}(x) = \Pi\left(\frac{x}{H}\right) * \frac{1}{H}\Lambda\left(\frac{x}{H}\right) = (\Pi * \Lambda)\left(\frac{x}{H}\right). \quad (3)$$

Splitting the domain of integration in the expression for W_{TSC} into five disjoint intervals shows that

$$(\Pi * \Lambda)(x) = \begin{cases} \frac{1}{8}(3 - 2|x|)^2, & \frac{1}{2} \le |x| < \frac{3}{2} \\ \frac{3}{4} - x^2, & |x| < \frac{1}{2} \\ 0, & \text{otherwise.} \end{cases}$$

Two- and three-dimensional versions of the assignment functions in Equation 3 are products of the assignment functions in each dimension. For example, the three-dimensional assignment function W is

$$W(\mathbf{x}) = W(x)W(y)W(z).$$

Hence, the mass assigned at mesh-point at $\mathbf{x_p}$ is

$$m(\mathbf{x_p}) = \sum_i m_i W_{\mathbf{p}}(\mathbf{x}_i),$$

or, in terms of density ρ,

$$\rho(\mathbf{x_p}) = \frac{1}{V} \sum_i m_i W_{\mathbf{p}}(\mathbf{x}_i), \quad (4)$$

where $V = H^3$ is the volume of a cell and i indexes the particles.

Obviously, Equation 4 is not suitable for direct application in the actual algorithm. Instead, we iterate over all particles, identify the parent cell $\mathbf{p}$ of each particle (and its neighborhood) and update ρ. This process is illustrated in Algorithm 1. The set $\mathcal{C}_S(\mathbf{x}_i)$ of cells that have to be considered while

Algorithm 1 Density assignment algorithm

```
1: for each particle i do
2:     for each cell q in C_S(x_i) do
3:         ρ(x_q) ← ρ(x_q) + m_i W(x_i − x_q)/V
```

assigning density from the i-th particle, depends on the shape S of the particle. Specifically, we have $\mathcal{C}_{\text{NGP}}(\mathbf{x}) = \{[\mathbf{x}/H]\}$, $\mathcal{C}_{\text{CIC}}(\mathbf{x}) = \{\lfloor \mathbf{x}/H \rfloor + \mathbf{t} \mid t_i = 0, 1\}$, and $\mathcal{C}_{\text{TSC}}(\mathbf{x}) = \{[\mathbf{x}/H] + \mathbf{t} \mid t_i = -1, 0, 1\}$. It follows that $|\mathcal{C}_{\text{NGP}}(\mathbf{x})| = 1$, $|\mathcal{C}_{\text{CIC}}(\mathbf{x})| = 8$, and $|\mathcal{C}_{\text{TSC}}(\mathbf{x})| = 27$ which illustrates the increasing computational cost resulting from using higher-order assignment schemes. We note that Algorithm 1 can be parallelized if atomic increments are used in line 3.

2.2 Solving the field equation

The Poisson equation (Equation 2) can be restated in integral form

$$\phi(\mathbf{x}) = \int G(\mathbf{x} - \mathbf{x}')\rho(\mathbf{x}')dV',$$

which has the following discrete analogue

$$\phi(\mathbf{x_p}) = V \sum_{\mathbf{p'}} G(\mathbf{x_p} - \mathbf{x_{p'}})\rho(\mathbf{x_{p'}}), \tag{5}$$

where G is the Green's function (potential due to unit mass). The right-hand side of Equation 5 is a convolution sum that runs over a finite set of mesh points. If we assume periodic boundary conditions, we can apply the discrete Fourier transform to both sides and use the convolution theorem to conclude that[1]

$$\hat{\phi}(\mathbf{k}) = \hat{G}(\mathbf{k})\hat{\rho}(\mathbf{k}). \tag{7}$$

An approximation to $\hat{G}$ can be found using a discretized version of the Laplacian in Equation 5. Specifically, for a 7-point stencil,

$$\begin{aligned} 4\pi G\rho(\mathbf{x}_{ijk}) = &\frac{\phi(\mathbf{x}_{i-1,j,k}) - 2\phi(\mathbf{x}_{ijk}) + \phi(\mathbf{x}_{i+1,j,k})}{H^2} \\ &+ \frac{\phi(\mathbf{x}_{i,j-1,k}) - 2\phi(\mathbf{x}_{ijk}) + \phi(\mathbf{x}_{i,j+1,k})}{H^2} \\ &+ \frac{\phi(\mathbf{x}_{i,j,k-1}) - 2\phi(\mathbf{x}_{ijk}) + \phi(\mathbf{x}_{i,j,k+1})}{H^2}. \end{aligned}$$

Applying the discrete Fourier transform to both sides and using the shift theorem we get

$$\begin{aligned} 4\pi G\hat{\rho}(\mathbf{k}) &= \frac{1}{H^2}\sum_{i=1}^{3}\left(e^{-iHk_i} + e^{iHk_i} - 2\right)\hat{\phi}(\mathbf{k}) \\ &= \frac{1}{H^2}\sum_{i=1}^{3}\left(e^{iHk_i/2} - e^{-iHk_i/2}\right)^2\hat{\phi}(\mathbf{k}) \\ &= -\frac{4}{H^2}\sum_{i=1}^{3}\sin^2\left(\frac{Hk_i}{2}\right)\hat{\phi}(\mathbf{k}). \end{aligned}$$

and hence

$$\hat{\phi}(\mathbf{k}) = -4\pi G \underbrace{\frac{(H/2)^2}{\sin^2(Hk_1/2) + \sin^2(Hk_2/2) + \sin^2(Hk_3/2)}}_{\hat{G}(\mathbf{k})}\hat{\rho}(\mathbf{k}),$$

where $\hat{G}$ can be identified by comparison with Equation 7. It is worth noting that the constant multiplier $(-4\pi G)$ is often left out of $\hat{G}$ (this is the convention used in [3]). In the implementation, values of $\hat{G}$ should be computed only once and saved for future look-up.

2.3 Field strength calculation

The strength $\mathbf{g}$ of the gravitational field at mesh-point $\mathbf{x_p}$ can be approximated using a central difference. Our implementation currently supports two types of finite differences, described below.

The two-point finite difference operator $\mathbf{D}$, whose x component is given by

$$D_x(\phi)(\mathbf{x_p}) = \frac{\phi(\mathbf{x}_{i+1,j,k}) - \phi(\mathbf{x}_{i-1,j,k})}{2H}$$

[1] In this work, the Hockney & Eastwood definition of DFT is used, i.e.

$$D(x_p) = \frac{1}{L}\sum_{l=0}^{N-1}\hat{D}(k)e^{ikx_p}, \quad \hat{D}(k) = H\sum_{p=0}^{N-1}D(x_p)e^{-ikx_p},$$

where $x_p = pH$. The conversion between this form and another popular definition,

$$\widetilde{D_H}(k) = \sum_{p=0}^{N-1}D_H(p)e^{-i2\pi kp/N}, \tag{6}$$

is given by

$$\widetilde{D_H}(k) = \frac{1}{H}\hat{D}\left(\frac{2\pi}{NH}k\right),$$

where $D_H(p) = pH$.

(and analogously for the y and z components), is second order accurate.

The fourth-order accurate finite difference is given by

$$D_x(\phi)(\mathbf{x_p}) = \alpha\frac{\phi(\mathbf{x}_{i+1,j,k}) - \phi(\mathbf{x}_{i-1,j,k})}{2H} + (1-\alpha)\frac{\phi(\mathbf{x}_{i+2,j,k}) - \phi(\mathbf{x}_{i-2,j,k})}{4H},$$

where $\alpha = 4/3$.

We can alternatively define the finite difference operators in terms of the delta function to get rid of the dependence on the differenced function. (Technically, the resulting quantities are functions rather than operators.) Consider for example the two-point finite difference in Equation 8. The definition can be generalized beyond mesh points by letting

$$D_j(\phi)(\mathbf{x}) = -\frac{\phi(\mathbf{x}+H\mathbf{e}_j) - \phi(\mathbf{x}-H\mathbf{e}_j)}{2H} = -\int\left[\frac{\delta(\mathbf{x}+H\mathbf{e}_j-\mathbf{x}') - \delta(\mathbf{x}-H\mathbf{e}_j-\mathbf{x}')}{2H}\right]\phi(\mathbf{x}')d\mathbf{x}',$$

where $\mathbf{e}_j$ is the j-th standard basis vector. This motivates us to define

$$D_j(\mathbf{x}) = \frac{\delta(\mathbf{x}+H\mathbf{e}_j-\mathbf{x}') - \delta(\mathbf{x}-H\mathbf{e}_j-\mathbf{x}')}{2H} \tag{8}$$

and

$$D_j(\mathbf{x}) = \frac{\delta(\mathbf{x}+H\mathbf{e}_j-\mathbf{x}') - \delta(\mathbf{x}-H\mathbf{e}_j-\mathbf{x}')}{2H} + (1-\alpha)\frac{\delta(\mathbf{x}+2H\mathbf{e}_j-\mathbf{x}') - \delta(\mathbf{x}-2H\mathbf{e}_j-\mathbf{x}')}{4H} \tag{9}$$

as the two-point as four-point finite difference operators respectively.

If ϕ denotes the gravitational potential, then the field $\mathbf{g}$ is approximated at mesh point $\mathbf{x_p}$ as

$$\mathbf{g}(\mathbf{x_p}) = -\mathbf{D}(\phi)(\mathbf{x_p}).$$

2.4 Interpolation

The value of the field strength $\mathbf{g}(\mathbf{x})$ at the position particle's position $\mathbf{x}$ is calculated by interpolating the values of $\mathbf{g}$ from the neighboring mesh-points. Formally,

$$\mathbf{g}(\mathbf{x}) = \sum_{\mathbf{p}} W(\mathbf{x}-\mathbf{x_p})\mathbf{g}(\mathbf{x_p}).$$

In practice, there is no need to sum over all mesh points. Instead, we use an algorithm analogous to Algorithm 1 to only include the cells with non-zero contribution to the sum. The method is illustrated in Algorithm 2. It is important to note that in order to retain correct physical behavior, the interpolation

Algorithm 2 Field strength interpolation

```
for each particle i do
    for each cell q in C_S(x_i) do
        g(x_i) ← Σ_q W(x_i − x_q) g(x_q)
```

and mass assignment schemes must use the same shape to represent the particles. The procedure in Algorithm 2 is trivially parallelized by converting the sequential loop into a parallel one.

The procedures of density assignment and interpolation presented in Algorithm 1 and Algorithm 2 are high level description. More concrete formulations suitable for direct use in an implementation are given in [3] and [4].

3 Particle-particle particle-mesh method

The P^{3}M algorithm is a hybrid method: Forces between distant particles are calculated using the PM method, whereas, for particles lying closely together, the PP method is used. The total force applied to particle i is

$$\mathbf{F}_i^{\mathrm{SR}} + \mathbf{F}_i = \sum_{j\neq i}(\mathbf{f}_{ij}^{\mathrm{tot}} - \mathbf{R}_{ij}) + \mathbf{F}_i, \tag{10}$$

where $\mathbf{F}_i \approx \sum_{j \neq i} \mathbf{R}_{ij}$ is the force computed using the PM method and $\mathbf{R}_{ij} = \mathbf{R}(\mathbf{x}_i - \mathbf{x}_j)$ is a prescribed *reference force*. The reference force is defined as the force between two particle-clouds, i.e. each particle is represented by a sphere with diameter a and a given density profile. The two examples of reference forces described in [3] are

$$R(r) = Gm_1m_2 \times \begin{cases} \frac{1}{35a^2}(224\xi - 224\xi^3 + 70\xi^4 + 48\xi^5 - 21\xi^6), & 0 \leq \xi \leq 1 \\ \frac{1}{35a^2}(12/\xi^2 - 224 + 896\xi - 840\xi^2 + 224\xi^3 + 70\xi^4 - 48\xi^5 + 7\xi^6), & 1 < \xi \leq 2 \\ \frac{1}{r^2}, & \xi > 2 \end{cases}$$

where $\xi = 2r/a$ for a sphere with uniformly decreasing density (shape S_2) and

$$R(r) = Gm_1m_2 \times \begin{cases} \frac{1}{a^2}(8r/a - 9r^2/a^2 + 2r^4/a^4), & r < a \\ \frac{1}{r^2}, \end{cases} \tag{11}$$

for a solid sphere (shape S_1). The reference force vector lies along the line joining the two bodies.

3.1 Optimal Green's function

As it is apparent from Equation 10, the method's validity depends on how well the reference force is approximated by the mesh force. The average deviation between the two forces can be minimized by a suitable choice of Green's function given (in k space) by

$$\hat{G}(\mathbf{k}) = \frac{\hat{\mathbf{D}}(\mathbf{k}) \cdot \sum_{\mathbf{n}} \hat{U}^2(\mathbf{k_n}) \hat{\mathbf{R}}(\mathbf{k_n})}{|\hat{\mathbf{D}}(\mathbf{k})|^2 \left[\sum_{\mathbf{n}} \hat{U}^2(\mathbf{k_n})\right]^2}. \tag{12}$$

The derivation is mathematically involved and is detailed in [3]. We now proceed to examine the terms appearing in Equation 12.

3.1.1 Finite difference operator

The Fourier transform $\hat{\mathbf{D}}$ of the two-point finite difference operator defined in Equation 8 has the components

$$\begin{aligned} \hat{D}_j(\mathbf{k}) &= \int D_j(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}} d\mathbf{x} \\ &= \frac{1}{2H}\left(\int \delta(\mathbf{x} + H\mathbf{e}_j) e^{-i\mathbf{k}\cdot\mathbf{x}} d\mathbf{x} - \int \delta(\mathbf{x} - H\mathbf{e}_j) e^{-i\mathbf{k}\cdot\mathbf{x}} d\mathbf{x}\right) \\ &= \frac{1}{2H}(e^{ik_jH} - e^{-ik_jH}) = \frac{i\sin(k_jH)}{H}. \end{aligned}$$

Similarly, for the four-point finite difference (Equation 9) we have

$$\hat{D}_j = \alpha \frac{i \sin k_j H}{H} + (1-\alpha)\frac{i \sin 2k_j H}{2H},$$

where $j = 1, 2, 3$.

3.1.2 Assignent function

The quantity $\hat{U}$ is defined as $\hat{W}/V$. For the mass assignment scheme hierarchy described in subsection 2.1 we have

$$\hat{U}(\mathbf{k}) = \left(\prod_{i=1}^{3} \frac{\sin(k_iH/2)}{k_iH/2}\right)^p,$$

where $p = 1, 2, 3, \ldots$ with $p = 1$ corresponding to NGP assignment, etc. In particular, for the TSC assignment scheme, the *alias sum*[2]

$$\sum_{\mathbf{n}} \hat{U}^2(\mathbf{k_n}) \equiv \sum_{\mathbf{n}} \hat{U}^2\left(\mathbf{k} + \mathbf{n}\frac{2\pi}{H}\right)$$

[2]To get the alias sums compatible with the DFT definition given in Equation 6, one has to compute

$$\sum_{\mathbf{n}} \tilde{U}^2(\mathbf{k_n}) \equiv \sum_{\mathbf{n}} \tilde{U}^2(\mathbf{k} + \mathbf{n}N) = \frac{1}{H}\sum_{\mathbf{n}} \hat{U}^2\left(\mathbf{k}\frac{2\pi}{NH} + \mathbf{n}\frac{2\pi}{H}\right)$$

instead.

can be rewritten as

$$\begin{aligned}\sum_{\mathbf{n}} \hat{U}^2\left(\mathbf{k}+\mathbf{n}\frac{2\pi}{H}\right) &= \sum_{\mathbf{n}}\prod_{i=1}^{3}\left[\frac{\sin(k_iH/2+n_i\pi)}{k_iH/2+n_i\pi}\right]^6 \\ &= \prod_{i=1}^{3}\sum_{n_i}\left[\frac{\sin(k_iH/2+n_i\pi)}{k_iH/2+n_i\pi}\right]^6 \\ &= \prod_{i=1}^{3}\left[\sin^6\left(\frac{k_iH}{2}\right)\sum_{n_i}\frac{1}{(k_iH/2+n_i\pi)^6}\right]\end{aligned}$$

Using the the partial fractions expansion of the cotangent function [1],

$$\frac{(-1)^s}{s!}\frac{d^s}{dx^s}\cot x = \sum_{n=-\infty}^{\infty}\frac{1}{(x-n\pi)^{s+1}},$$

we can simplify the sum over n_i to

$$\frac{-1}{5!}\frac{d^5}{dx^5}\cot\left(\frac{k_iH}{2}\right) = 1-\sin^2\frac{k_iH}{2}+\frac{2}{15}\sin^4\frac{k_iH}{2}.$$

Hence,

$$\sum_{\mathbf{n}}\hat{U}^2_{\mathrm{TSC}}(\mathbf{k_n}) = \prod_{i=1}^{3}\left(1-\sin^2\frac{k_iH}{2}+\frac{2}{15}\sin^4\frac{k_iH}{2}\right).$$

Using the same approach, we can obtain similar results for the CIC and NGP schemes, namely

$$\sum_{\mathbf{n}}\hat{U}^2_{\mathrm{CIC}} = \frac{1}{3}\prod_{i=1}^{3}\left(1+2\cos^2\frac{k_iH}{2}\right) \quad \text{and} \quad \sum_{\mathbf{n}}\hat{U}^2_{\mathrm{NGP}} = 1.$$

3.1.3 Reference force

The quantity $\hat{\mathbf{R}}$, the transformed reference force, is related to the shape S of the particle-cloud by

$$\hat{\mathbf{R}}(\mathbf{k}) = -\frac{i\mathbf{k}\hat{S}^2(k)}{k^2}, \tag{13}$$

where $k=|\mathbf{k}|$. This can be shown by recalling that $\mathbf{R}(\mathbf{x}_i-\mathbf{x}_j)$ is the force applied to cloud i due to cloud j. Hence $\mathbf{R}(\mathbf{x})$ is the force on cloud centered at $\mathbf{x}$ due to a cloud at the origin. The force acting on mass element $d\mathbf{x}'S(\mathbf{x}'-\mathbf{x})$ of the cloud centered at $\mathbf{x}$ is therefore

$$Gd\mathbf{x}'S(\mathbf{x}'-\mathbf{x})\int d\mathbf{x}''S(\mathbf{x}'')\frac{\mathbf{x}'-\mathbf{x}''}{|\mathbf{x}'-\mathbf{x}''|^3}$$

and the total force is

$$\mathbf{R}(\mathbf{x}) = G\int d\mathbf{x}'S(\mathbf{x}'-\mathbf{x})\int d\mathbf{x}''S(\mathbf{x}'')\frac{\mathbf{x}'-\mathbf{x}''}{|\mathbf{x}'-\mathbf{x}''|^3}. \tag{14}$$

The expression on the right-hand side of Equation 14 is a double convolution, $\mathbf{R}=S*(S*\mathbf{g})$, where $\mathbf{g}(\mathbf{x})=\mathbf{x}/|\mathbf{x}|^3$. The x-coordinate of $\mathbf{g}$ is $x/|\mathbf{x}|^3$ which coincides with $\partial h/\partial x$ for $h(\mathbf{x})=-1/|\mathbf{x}|$. The Fourier transform of h is given in [2] (p. 363):

$$\hat{h}(\mathbf{k}) = \frac{4\pi}{|\mathbf{k}|^2}.$$

The formula for the transform of a derivative yields

$$\hat{g}_x(\mathbf{k}) = \frac{4\pi ik_x}{|\mathbf{k}|^2}.$$

Applying the convolution theorem twice and factoring the constant $(-4\pi G)$ out of the Green's function in Equation 12 leaves us with the desired Equation 13.

Since the shapes S are spherically symmetric, the calculation of $\hat{S}$ (appearing in Equation 13) can be simplified. The Fourier transform of S is

$$\hat{S}(\mathbf{k}) = \int S(\mathbf{x}) e^{-i\mathbf{k}\cdot\mathbf{x}} d\mathbf{x}.$$

Observe that $\mathbf{k}\cdot\mathbf{x} = kr\cos\theta$, where θ is the angle between $\mathbf{k}$ and $\mathbf{x}$ and $r = |\mathbf{x}|$. Since S is invariant under rotations, θ can be chosen to be the angle between $\mathbf{x}$ and the z-axis. Thus, the integral, rewritten in spherical coordinates becomes

$$\hat{S}(k) = \int_0^{2\pi} d\phi \int_0^{\pi} d\theta \int_0^{\infty} dr S(r) e^{-ikr\cos\theta} r^2 \sin\theta = 2\pi \int_0^{\infty} r^2 dr \int_0^{\pi} d\theta e^{-ikr\cos\theta} \sin\theta.$$

The θ-integral upon the substitution $-kr\cos\theta \to u$ becomes

$$\frac{1}{kr}\int_{-kr}^{kr} e^{iu} du = \frac{1}{kr}\int_{-kr}^{kr} (\cos u + i\sin u) du = \frac{2\sin kr}{kr}$$

and hence

$$\hat{S}(k) = 4\pi \int_0^{\infty} r^2 S(r) \frac{\sin kr}{kr} dr.$$

This integral, evaluated for the S_1 and S_2 shapes respectively, gives

$$\hat{S}_1(k) = \frac{3}{(ka/2)^3}\left(\sin\frac{ka}{2} - \frac{ka}{2}\cos\frac{ka}{2}\right)$$

and

$$\hat{S}_2(k) = \frac{12}{(ka/2)^4}\left(2 - 2\cos\frac{ka}{2} - \frac{ka}{2}\sin\frac{ka}{2}\right).$$

Finally, we note that the infinite sum in the numerator of Equation 12 does not have a closed form but this does not pose a problem since the summand decays rapidly with $\mathbf{n}$ moving further away from zero.

The result of applying the optimal Green's function in Equation 7 is shown in Figure 1. As can be

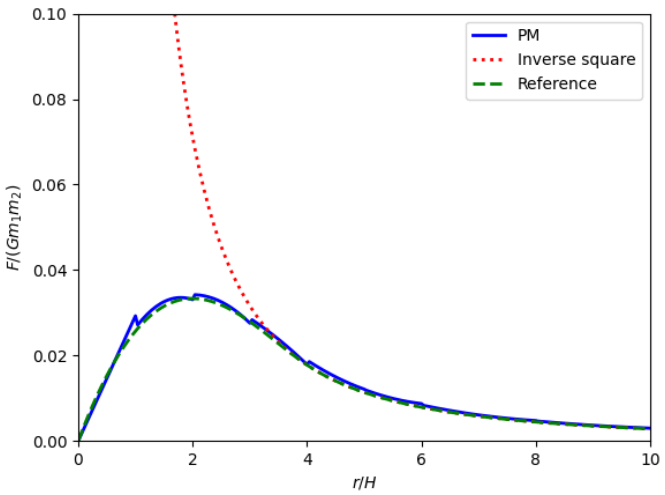


Figure 1: Magnitude of the force between two masses. The mesh approximation to the reference force was calculated using the PM method with TSC assignment scheme, two-point finite difference, and Green's function optimal for the S_1 shape with diameter $a = 4H$. The force resultant from the universal law of gravitation is also shown.

seen in the figure, the PM force closely follows the reference force. Moreover, for $r > a$, the reference force is identical to the inverse-square force. It is also worth noting that the reference force (and its mesh approximation) approximates the inverse-square force accurately for r slightly smaller than a. For this reason r_e, the *cutoff radius* designating the boundary of the region handled by the direct summation, can be chosen to be smaller than a (e.g. $r_e = 0.7a$). This can have a noticeable positive impact on performance.

3.2 Identifying close pairs of particles

In the P^{3}M method, in addition to the mesh used in the PM algorithm (the "potential mesh"), a second mesh (the *chaining mesh*) is used. The chaining mesh is sparser than the potential mesh. Its sole purpose is to partition the space into cells so that particles "close" to the ones in a given cell can be found efficiently. In this context, two particles are considered to lie close to one another if their separation is less than the cutoff radius.

The number of chaining mesh cells in a single dimension is given by $M_i = \lfloor L_i/r_e \rfloor$, where L_i is the side length of the computational box ($i = 1, 2, 3$). This implies that the side length of a chaining mesh cell is $HC_i = L_i/M_i \geq r_e$. Thus, for every particle i in a given cell $\mathbf{p}$, it is sufficient to search through the immediate neighborhood of $\mathbf{p}$ to find all the particles within the cutoff radius from i.

The chaining mesh can be implemented as a *head-of-chain* (HOC) array, depicted in Figure 2. The

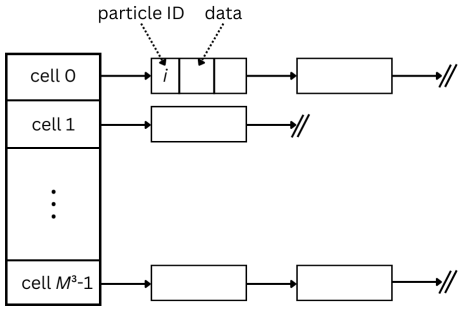


Figure 2: Head-of-chain data structure used for mapping particles to their parent cells in the chaining mesh. Here $M_1 = M_2 = M_3 = M$.

basic version of the HOC array is very cheap to build since the whole process is linear in the number of particles. Hence, the array can be constructed anew at each time step. Additional computational savings can be made by preallocating a memory pool large enough to store N nodes of the linked lists and reusing it for the HOC array initialization. Another noteworthy possible optimization is to sort the individual linked lists by the value of a particle coordinate of choice, say the y coordinate. This allows for an early return from the direct summation loop on the condition that $|y_i - y_j| > r_e$ while particle i is sweeping through a cell containing particles j.

3.3 Short-range correction

The short-range correction, which takes place immediately after the mesh forces are found using the PM method, is at the heart of the P^{3}M algorithm. Since it scales with a square of the number of particles in each neighborhood, further optimizations are highly desirable.

By Newton's 3rd law, $\mathbf{f}_{ji}^{\mathrm{SR}} = -\mathbf{f}_{ij}^{\mathrm{SR}}$, which allows us to do the calculation of the short-range inter-particle force for any pair (i, j) of particles only once, leading to the reduction of the total running time by half. In practice, the particle i will update its total short-range force $\mathbf{F}_i^{\mathrm{SR}}$ as well as the total short-range force $\mathbf{F}_j^{\mathrm{SR}}$ of its neighbor j. To avoid double-counting, the particle i residing in cell $\mathbf{q}$ has to look for its neighbors in a subset $\mathcal{N}$ of the immediate neighborhood of $\mathbf{q}$. More specifically, define

$$\mathcal{N}(\mathbf{q} = (q_1, q_2, q_3)) = \{(q_1 + t, q_2 - 1, q_3 + s), (q_1 + s, q_2, q_3 - 1), (q_1 - 1, q_2, q_3) \mid s, t = -1, 0, 1\}.$$

Thus $|\mathcal{N}| = 13$, which is half of the size of the immediate neighborhood.

The short-range correction part of the P^{3}M method is shown in Algorithm 3. The UpdateShort-Range procedure is defined in Algorithm 4. As suggested in [3], the computational burden of operations in lines 5-8 in Algorithm 4 can be greatly reduced by storing the values of $f^{\mathrm{SR}}(r)/r = (f^{\mathrm{tot}}(r) - R(r))/r$ in a lookup table T at uniform intervals Δ^2 of $[0, r_e^2]$ and interpolating. The schematic depiction of the interpolation is shown in Figure 3. If we define $\xi = r^2/\Delta^2$ and $t = \lfloor \xi \rfloor$, then

$$\frac{f^{\mathrm{SR}}(r)}{r} \approx T[t]\,(1 - (\xi - t)) + T[t+1](\xi - t) = T[t] + (\xi - t)(T[t+1] - T[t]).$$

The value $\mathbf{f}_{ij}^{\mathrm{SR}}$ can then be obtained by multiplying the interpolated quantity $f^{\mathrm{SR}}(r_{ij})/r_{ij}$ by $Gm_im_j\mathbf{r}_{ij}$, eliminating the use of the square root operations and reducing the total number of floating-point operations to just four.

Algorithm 3 Short-range correction

for each chaining cell $\mathbf{q}$ **do**
  for each $\mathbf{q}_n \in \mathcal{N}(\mathbf{q}) \cup \{\mathbf{q}\}$ **do**
    for each $i \in \mathrm{HOC}(\mathbf{q})$ **do**
      for each $j \in \mathrm{HOC}(\mathbf{q}_n)$ **do**
        if $|y_i - y_j| > r_e$ **then**
          break
        UpdateShortRange(i, j, $\mathbf{q}$, $\mathbf{q}_n$)

Algorithm 4 Updating short-range forces

1: **procedure** UpdateShortRange(i, j, $\mathbf{q}$, $\mathbf{q}_n$)
2: **if** $i = j$ **then return**
3: $\mathbf{r}_{ij} \leftarrow \mathbf{r}_i - \mathbf{r}_j$
4: **if** $|\mathbf{r}_{ij}|^2 > r_e^2$ **then return**
5: $r_{ij} \leftarrow |\mathbf{r}_{ij}|$
6: $\hat{\mathbf{r}}_{ij} \leftarrow \mathbf{r}_{ij}/r_{ij}$
7: $\mathbf{R}_{ij} \leftarrow -m_i m_j R(r_{ij})\hat{\mathbf{r}}_{ij}$
8: $\mathbf{f}^{\mathrm{tot}} \leftarrow -Gm_i m_j/r_{ij}^2 \hat{\mathbf{r}}_{ij}$
9: $\mathbf{f}_{ij}^{\mathrm{SR}} \leftarrow \mathbf{f}^{\mathrm{tot}} - \mathbf{R}_{ij}$
10: $\mathbf{f}_{ji}^{\mathrm{SR}} \leftarrow -\mathbf{f}_{ij}^{\mathrm{SR}}$
11: $\mathbf{F}_i^{\mathrm{SR}} \leftarrow \mathbf{F}_i^{\mathrm{SR}} + \mathbf{f}_{ij}^{\mathrm{SR}}$
12: **if** $\mathbf{q}_n \neq \mathbf{q}$ **then** ▷ *Avoid double-counting in the parent cell*
13: $\mathbf{F}_j^{\mathrm{SR}} \leftarrow \mathbf{F}_j^{\mathrm{SR}} + \mathbf{f}_{ji}^{\mathrm{SR}}$

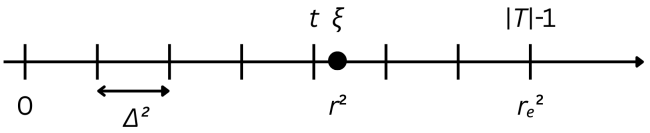


Figure 3: Interpolation of short-range force values.

The procedure outlined inAlgorithm 3 can be parallelized by splitting the work done in the outmost loop between some number of threads. In doing so, extra care has to be taken to avoid data races. A thread t that is currently processing cell $\mathbf{p}$ and its neighbors (we say that t is *assigned* to $\mathbf{p}$) may "clash" with a different thread assigned to a nearby cell $\mathbf{q}$ (because possibly $\mathbf{p} \in \mathcal{N}(\mathbf{q})$). However, by the construction of the set $\mathcal{N}$, it is possible to split the short-range force into 14 parts, each of which is accessed by only one thread. For example, consider a particle i in cell $\mathbf{p} = (p_1, p_2, p_3)$ (in other words, $\mathbf{p}$ is the parent cell of i). If thread t is currently assigned to this cell, t will update the part of $\mathbf{F}_i^{\mathrm{SR}}$ corresponding to updates of i coming from within the same cell as the parent cell of i. Possibly at the same time, thread t' assigned to cell $\mathbf{q} = (p_1 + 1, p_2, p_3)$ will update a different part of $\mathbf{F}_i^{\mathrm{SR}}$, i.e. the part corresponding to updates of i coming from the cell "to the right" of the parent cell of i. Since only one thread is responsible for updates to particle i coming "from the right," (or any other "direction") no data races can occur. This approach has two drawbacks. First, memory requirements increase significantly as memory for $13N$ additional three-dimensional vectors has to be used. Second, there is no way to guarantee uniform work distribution among the threads.

4 Barnes-Hut algorithm

The idea behind Barnes-Hut algorithm differs substantially from previously described mesh-based methods. The algorithm deals with gravitational forces directly instead of deriving them from the mesh-defined potential, as was the case with the PM and $\mathrm{P^3M}$ methods. Significant reduction of time complexity (from quadratic to $O(N \log N)$) is achieved by approximating "far enough" groups of particles by their center of mass (COM) [6]. The grouping of particles is hierarchical in nature and is thus best understood as a tree. The entire set of particles comprises the top-level group, represented by the root of the tree; the eight children of the root node are representative of groups of particles residing in each of the octants of the computational domain, etc. The process of subdividing the space into eight smaller volumes at each node continues recursively until there is only one or zero particles left in a given volume. Nodes which satisfy this condition are the leafs of tree and are sometimes called the *external nodes*. The remaining nodes, each of which has eight children, are called *internal nodes*.

4.1 Building the tree

The data structure that fits the above description is called an *octree*. An internal node of the octree stores the COM vector and the total mass of the group it represents, whereas an external node stores a reference to the actual particles (or is empty if no particle was found in its associated volume). The recursive procedure of building the tree is shown in Algorithm 5.

Algorithm 5 Insert a particle into the Barnes-Hut tree

```
function INSERT(n, p)
    if n is an internal node then
        Update n.COM and total mass n.M of n with p
        INSERT(child of n that should contain p, p)
    else if n is empty then
        Assign p to n
    else                                              ▷ Occupied external node
        Subdivide n into child nodes
        Move existing particle p' in n into child that should contain p'
        Update center of mass and total mass of n with p and p'
        INSERT(child of n that should contain p, p)
```

4.2 Force calculation

The last method presented in this work is the Barnes-Hut algorithm. In the Barnes-Hut algorithm, the net gravitational force on a particle p is calculated by summing the contributions from single particles or groups of particles while traversing the tree. The decision whether the force can be approximated using the information stored in an internal node n depends on the relative distance from p to n.COM (the center of mass of group represented by n). The distance is relative to the *width* H of the node, i.e. the

side length of the cubical volume encompassed by the node. More concretely, the approximation takes place if $n.H/|n.\text{COM} - p.\mathbf{x}| < \theta$, where θ is the so-called *opening angle*. In the extreme case when θ is set to zero, no approximations take place, and the algorithm reduces to the PP method. The procedure described above is illustrated in Algorithm 6. In the implementation, the GRAVITY function calculates

Algorithm 6 Compute gravitational force on a particle using Barnes-Hut approximation

```
function FINDFORCE(n, p, θ)
    if n is an external node then
        if n contains a particle q ≠ p then
            p.F ← p.F + GRAVITY(q.x, q.m, p.x)
        return
    if n.H/|n.COM − p.x| < θ then
        p.F ← p.F + GRAVITY(n.COM, n.M, p.x)
        return
    for each child n_c of n do
        FINDFORCE(n_c, p, θ)
```

the gravitational force softened by ϵ, i.e. it returns the value

$$\mathbf{F}_{ij}^{\text{soft}} = -G\frac{m_i m_j}{(r_{ij}^2 + \epsilon^2)^{3/2}}\mathbf{r}_{ij}.$$

The pairwise potential energy associated with this force is given by

$$\Phi_{ij}^{\text{soft}} = -\frac{Gm_i m_j}{\sqrt{r_{ij}^2 + \epsilon^2}}. \tag{15}$$

We note that direct calculation of total potential energy is infeasible as $O(N^2)$ operations would be required. Instead, we use an approximation based on the values stored in the tree. The approximate value of the potential energy is accumulated for each particle using a procedure analogous to force calculation. Indeed, the only difference between the two is the replacement of gravitational force calculation in Algorithm 6 with potential energy calculation according to Equation 15.

It is also noteworthy that the procedure outlined in Algorithm 6 is embarrassingly parallel. In our CPU implementation, the workload is split between an arbitrary number of threads on particle-by-particle basis. On the other hand, the parallelization of the tree building procedure in Algorithm 5 is far from trivial (see for example [7]). In this work, we will not be exploring this idea further.

5 Time integration

In the previous sections we described various methods of calculating forces applied to particles in the simulation. Once these forces are found, the evolution of the system in time can be tracked by integrating Newton's 2nd law of motion,

$$\ddot{\mathbf{x}}_i = \frac{\mathbf{F}_i}{m_i}. \tag{16}$$

5.1 Euler's method

Possibly, the simplest numerical method that could be used is Euler's method described by the update rules

$$\begin{aligned} \mathbf{v}_i^{(k+1)} &= \mathbf{v}_i^{(k)} + \text{DT}\frac{\mathbf{F}_i^{(k+1)}}{m_i}, \\ \mathbf{x}_i^{(k+1)} &= \mathbf{x}_i^{(k)} + \text{DT}\mathbf{v}_i^{(k)}. \end{aligned} \tag{17}$$

The method defined in Equation 17 is not suitable for physical simulations, however. Its shortcomings are best illustrated by an example of an undamped pendulum of length l in gravitational field of magnitude g. Although it is a simple system, it illustrates the numerical challenges faced in gravitational simulations over long timescales, particularly the issue of energy preservation.

The motion of the pendulum is governed by the differential equation

$$\ddot{\theta} = -\frac{g}{l}\sin\theta,$$

and its kinetic and potential energy are given by $\mathrm{KE} = (1/2)ml^2\dot{\theta}$ and $\mathrm{PE} = -mgl\cos\theta$ respectively. As

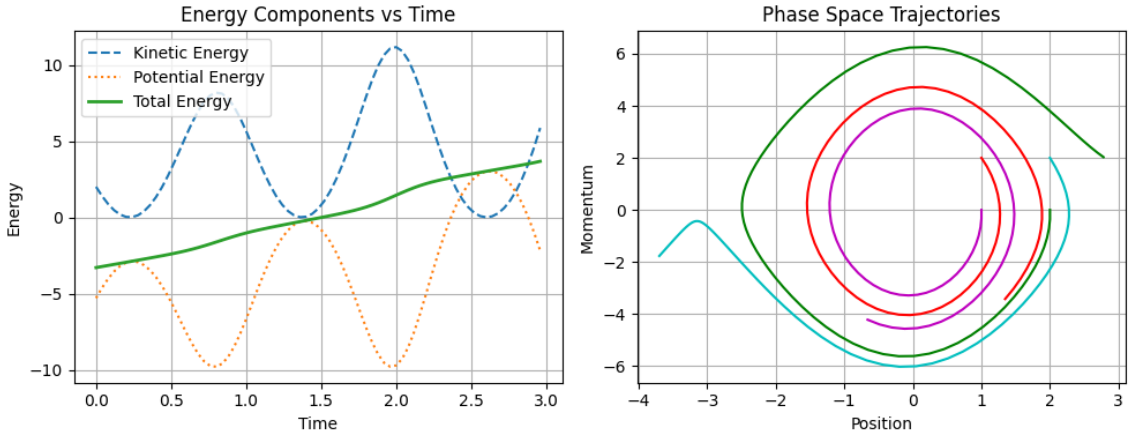


Figure 4: Behavior of Euler's method: lack of conservation of energy and phase space trajectories spiraling out.

shown in Figure 4, Euler's method fails to conserve total energy (PE + KE) and produces trajectories in phase space that are not closed, contrary to expectations for periodic systems. Additionally, the evolution of an area element in phase space violates Liouville's theorem, as described in [5], making the method unsuitable for long-term physical simulations (see Figure 5).

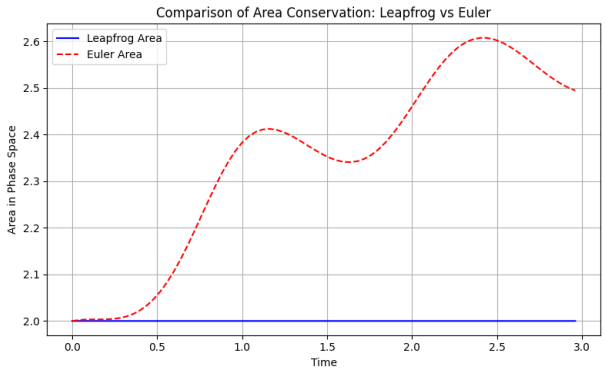


Figure 5: Area in phase space over time. Violation of Liouville's theorem by Euler's method.

5.2 Leapfrog algorithm

The leapfrog algorithm, given by the update rule [9]

$$\begin{aligned}
\mathbf{v}_i^{(1/2)} &= \mathbf{v}_i^{(0)} + \frac{1}{2}\mathrm{DT}\frac{\mathbf{F}_i^{(0)}}{m_i^{(0)}}, \\
\mathbf{x}_i^{(k+1)} &= \mathbf{x}_i^{(k)} + \mathrm{DT}\mathbf{v}_i^{(k+1/2)}, \\
\mathbf{v}_i^{(k+3/2)} &= \mathbf{v}_i^{(k+1/2)} + \mathrm{DT}\frac{\mathbf{F}_i^{(k+1)}}{m_i}.
\end{aligned} \tag{18}$$

is the preferred way of integrating Equation 16. When applied to the same pendulum system, it conserves energy much more faithfully and preserves the area in phase space, consistent with Liouville's theorem (see Figure 6 and Figure 5). Given its simplicity and excellent long-term energy behavior, we adopt the leapfrog algorithm to integrate Newton's equations in our program.

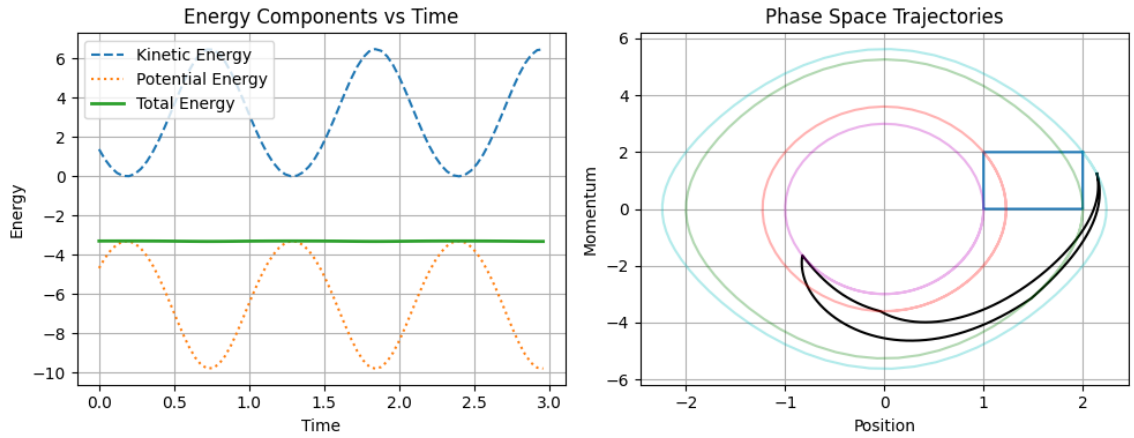


Figure 6: Behavior of the leapfrog algorithm: conservation of energy and phase space trajectories forming closed loops. Evolution of an area element in phase space is shown on the right-hand side: blue rectangle – initial conditions for many copies of the system; black distorted quadrilateral – their state by the end of the simulation.

6 Galaxy model

The model of a galaxy used as a test bed for the implementation is a simple one. The galaxy is assumed to comprise only two parts: a thin disk and a spherically symmetric halo. The disk comprises a large number of particles, each representing some number of stars. The halo is simulated as a fixed external gravitational field.

6.1 Disk

The disk particles are sampled from a radial distribution

$$p(r) = \frac{3}{\pi R_D^2}\left(1 - \frac{r}{R_D}\right), \; z = 0,$$

where R_D is the radius of the disk and $r \leq R_D$. The cumulative distribution function is therefore

$$F(r, \phi) = \int_0^{\phi}\int_0^{r} p(r')r'dr'd\phi' = \frac{\phi}{2\pi R_D^3}(3R_D r^2 - 2r^3)$$

and the marginal CDFs are

$$F_R(r) = F(r, 2\pi) = \frac{1}{R_D^3}(3R_D r^2 - 2r^3) \quad \text{and} \quad F_\Phi(\phi) = F(R_D, \phi) = \frac{\phi}{2\pi}.$$

Now we use inverse transform sampling to generate initial positions (r, ϕ) for the particles, i.e. $\phi = 2\pi u$ and r is given implicitly by $h(r) \equiv 2r^3 - 3R_D r^2 + uR_D^3 = 0$ with $u \sim U(0, 1)$. A straightforward calculation shows that $dh/dr < 0$ for $0 < r < R_D$ and $h(0)h(R_D) < 0$ implying that h has exactly one zero between 0 and R_D (which can be found for example using Newton's method).

Strength of the gravitational field $\mathbf{g}_D$ due to the disk at point $\mathbf{x}_0$ lying in the disk is

$$\mathbf{g}_D = G\int_0^{2\pi}\int_0^{R_D} \sigma(r)\frac{\mathbf{x} - \mathbf{x}_0}{|\mathbf{x} - \mathbf{x}_0|^3} r dr d\phi,$$

where $\sigma(r) = \sigma_0(1 - r/R_D)$ describes the density profile of the disk for $r \leq R_D$. If M_D is the total mass of the disk, then $\sigma_0 = 3M_D/(\pi R_D^2)$. By symmetry, the point $\mathbf{x}_0$ may be chosen to lie on the x-axis, i.e. $\mathbf{x}_0 = (-x_0, 0)$, so that $\mathbf{x} - \mathbf{x}_0 = (x_0 + r\cos\phi, r\sin\phi)$. Letting $\bar{r} = r/R_D$ and $\bar{x}_0 = x_0/R_D$, the integral becomes

$$\mathbf{g}_D = G\sigma_0\int_0^{2\pi}\int_0^{1}(1 - \bar{r})\frac{(\bar{x}_0 + \bar{r}\cos\phi, \bar{r}\sin\phi)}{|(\bar{x}_0 + \bar{r}\cos\phi, \bar{r}\sin\phi)|^3}\bar{r}d\bar{r}d\phi.$$

By symmetry $g_{D,y} = 0$ and thus the radial component of the field $\mathbf{g}_D$ at distance $R\bar{x}_0$ from the center is

$$g_{D,r} = -|\mathbf{g}_D| = -G\sigma_0 \int_0^{2\pi} \int_0^1 (1-\bar{r}) \frac{\bar{x}_0 + \bar{r}\cos\phi}{(\bar{x}_0^2 + \bar{r}^2 + 2\bar{x}_0\bar{r}\cos\phi)^{3/2}} \bar{r} d\bar{r} d\phi. \tag{19}$$

If the disk had constant density, $\mathbf{g}_D$ could be expressed in terms of elliptic integrals [8]. However, to the best of the author's knowledge, the integral in Equation 19 cannot be further simplified. For this reason, a crude approximation with a quadratic function is used: $g_{D,r} \approx a(r-h)^2 + k$, where the values $k = 2.5$ and $h = 0.66$ (the maximum of $g_{D,r}$ and the argument thereof) were estimated based on the graph of $g_{D,r}$ (see Figure 7). The value of $a = -k/h^2$ can be found by setting $g_{D,r}(0) = 0$ in the approximate formula.

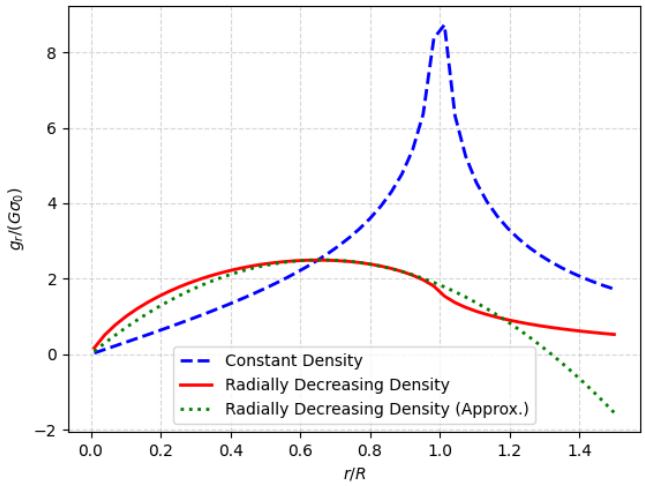


Figure 7: Magnitude of the radial component of the field strength due to a disk. The peak at $r = R$ for the constant density disk is in fact infinite.

6.2 Halo

The density profile of the halo is analogous to the one used for the disk, save for the fact it is 3-dimensional, i.e.

$$\rho(r) = \begin{cases} \rho_0 \left(1 - \frac{r}{R_H}\right), & r \le R_H \\ 0, & \text{otherwise,} \end{cases}$$

where R_H is the radius of the halo. If we let M_H be the mass of the halo, then $\rho_0 = 3M_H/(\pi R_H^3)$. Application of Gauss's law shows that we have

$$g_{H,r} = -GM_H \times \begin{cases} \frac{r}{R_H^3}\left(4 - \frac{3r}{R_H}\right), & r \le R_H \\ \frac{1}{r^2}, & \text{otherwise.} \end{cases}$$

6.3 Initial conditions

The total field $\mathbf{g} = \mathbf{g}_D + \mathbf{g}_H$ is used to find initial velocities for the particles with initial positions $(x, y, 0)$. The formula for the centripetal force yields

$$\frac{v^2}{r} = -g_r$$

and thus

$$\mathbf{v} = \left(-v\frac{y}{r}, v\frac{x}{r}, 0\right)$$

with $v = \sqrt{-rg_r}$ for counter-clockwise rotation.

7 Results

The parameters used in the simulation of a spiral galaxy are shown in Table 1. The galaxy is simulated

Parameter	Value
Halo radius	3 kpc
Halo mass	$60 \times 10^9 M_\odot$
Disk radius	15 kpc
Disk mass	$15 \times 10^9 M_\odot$
Disk thickness	0.3 kpc
Disk density profile	Uniformly decreasing
Mass assignment scheme	TSC
Finite difference	Two-point
Time integration method	Leapfrog

Table 1: Galaxy model parameters used in the simulation.

as an isolated system, however, in deriving Equation 7, periodic boundary conditions were assumed. The simplest way (and the one used) to obtain a free-space solution from the PM method is to extend the computational domain twice in every dimension and fill the space unused in mass distribution with zeros. The total size of the potential mesh used was $128 \times 128 \times 64$ with the region of interest occupying a box of size $60\,\text{kpc} \times 60\,\text{kpc} \times 30\,\text{kpc}$ located in a $64 \times 64 \times 32$ octant of the mesh.

7.1 Particle-mesh method

In the PM method, $N = 50,000$ particles were used. Cell size H and time-step length were set to $60/64 = 0.9375$ kpc, and 1 Myr respectively. The system's evolution over 200 Myr is shown in Figure 8.

During the simulation, total energy $E = \text{KE} + \text{PE}$, angular momentum $\mathbf{l}$, and the z-component of the momentum vector $\mathbf{p}$ should stay constant. The x- and y-components of momentum change due to the presence of an external gravitational field (representing the halo). We can verify if this variation satisfies the expected relation

$$\dot{\mathbf{p}} = \mathbf{F}^{\text{ext}} \tag{20}$$

by finding the initial total momentum $\mathbf{p}(t = 0)$ and incrementing the value of $\mathbf{p}$ in each time-step by $\mathbf{F}^{\text{ext}}\text{DT}$.

The exact calculation of the potential energy [5] using the formula

$$\text{PE} = -\sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{G m_i m_j}{r_{ij}}$$

is computationally infeasible considering the $O(N^2)$ cost. An approximation based on the potential values at mesh points,

$$\text{PE} \approx \frac{V}{2} \sum_{\mathbf{p}} \rho(\mathbf{x_p}) \phi(\mathbf{x_p}),$$

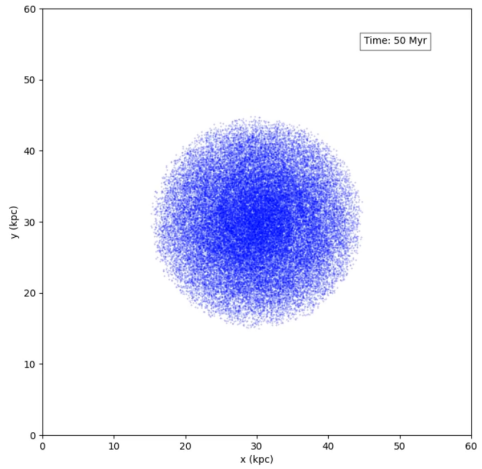
is used instead (for derivation refer to [3]).

7.2 Particle-particle particle-mesh method

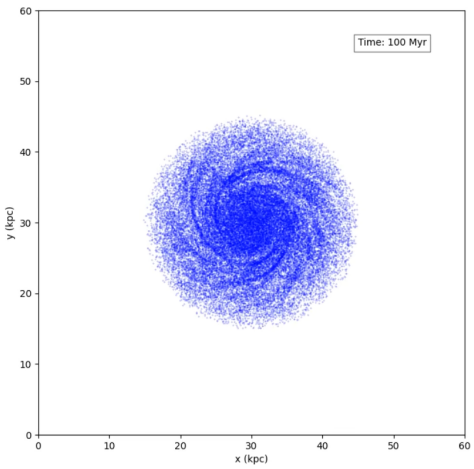
The P^3M based simulation uses the same parameters as the PM method. The reference force was calculated using the S_1 shape formula (Equation 11) with particle diameter $a = 3H$. The cutoff radius was set to $r_e = 0.7a$. One extra free parameter is the *softening length* ϵ which modifies the universal law of gravitation so that division by zero can be avoided, i.e. the modified law is

$$F_{ij}^{\text{soft}}(r) = \frac{G m_i m_j}{r_{ij}^2 + \epsilon^2}.$$

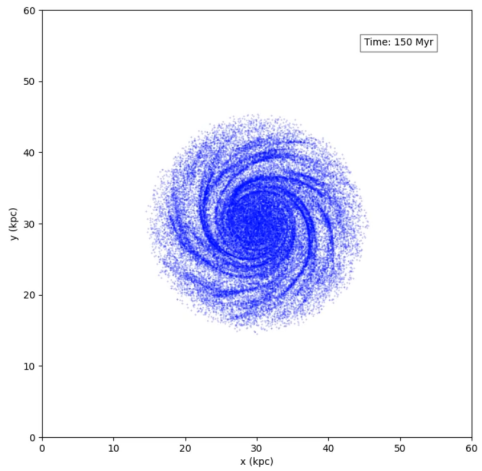
In the simulation, ϵ was set arbitrarily to 1.5 kpc. The system's evolution is presented in Figure 10. Graphs of energy, angular momentum, and momentum components vs. time are shown in Figure 11.



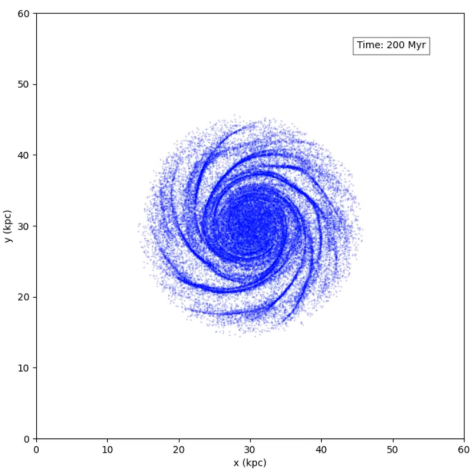
(a) $t = 50\,\mathrm{Myr}$



(b) $t = 100\,\mathrm{Myr}$

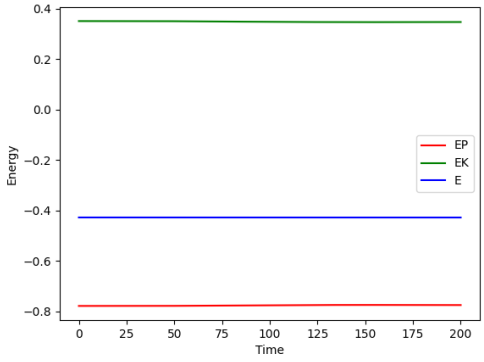


(c) $t = 150\,\mathrm{Myr}$

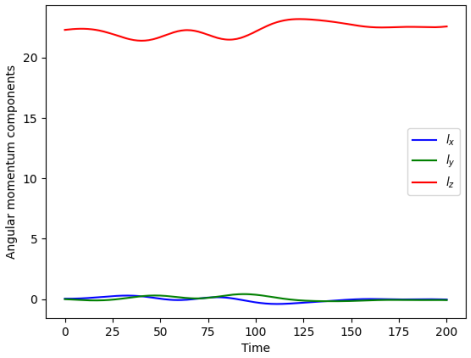


(d) $t = 200\,\mathrm{Myr}$

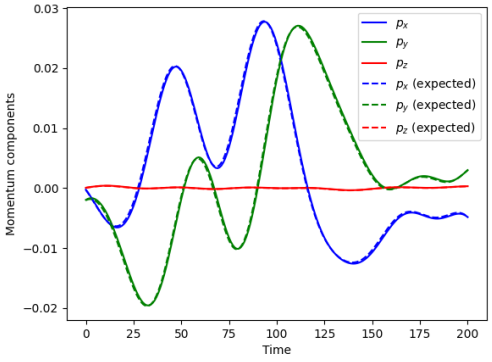
Figure 8: Evolution of a spiral galaxy as predicted by the PM method.



(a) Energy

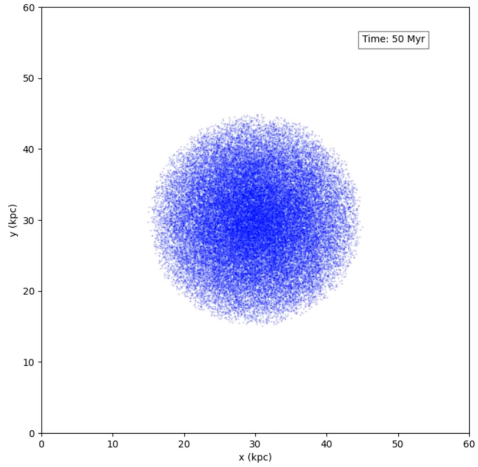


(b) Angular momentum

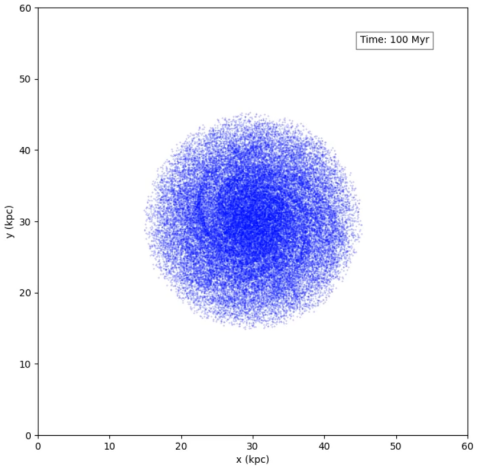


(c) Momentum; broken lines represent the expected momentum following Equation 20

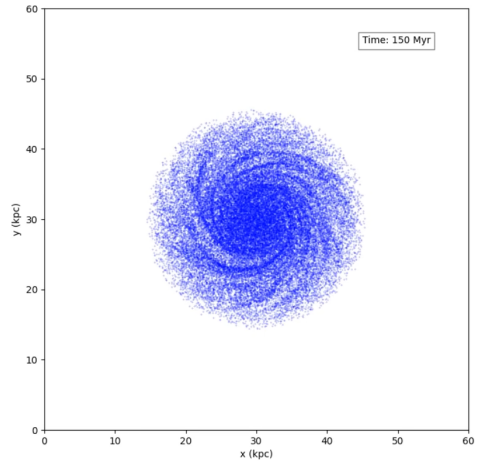
Figure 9: Fundamental physical quantities describing the system over time in the PM simulation. Time is in Myr and the quantities are expressed in units consistent with Table 1



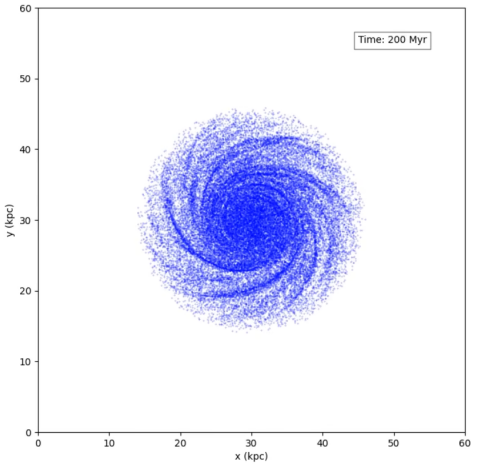
(a) $t = 50\,\mathrm{Myr}$



(b) $t = 100\,\mathrm{Myr}$

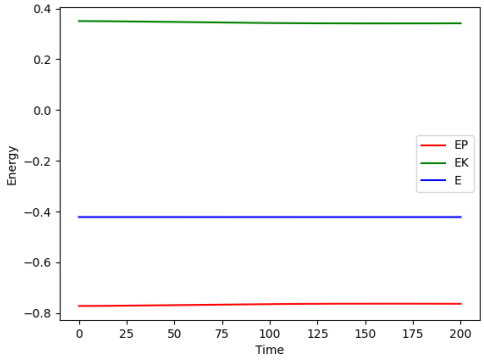


(c) $t = 150\,\mathrm{Myr}$

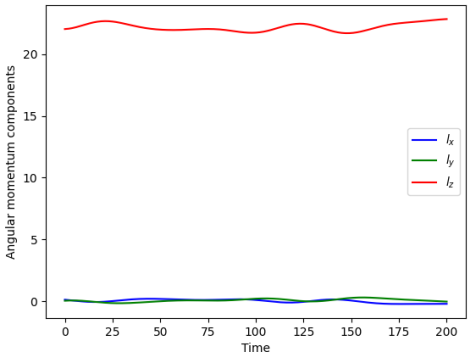


(d) $t = 200\,\mathrm{Myr}$

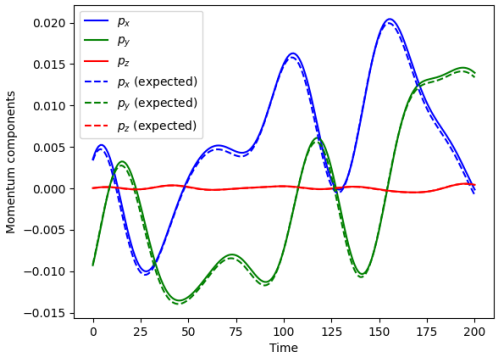
Figure 10: Evolution of a spiral galaxy as predicted by the $\mathrm{P^3M}$ method.



(a) Energy



(b) Angular momentum



(c) Momentum; broken lines represent the expected momentum following Equation 20

Figure 11: Fundamental physical quantities describing the system over time in the P^3M simulation. Time is in Myr and the quantities are expressed in units consistent with Table 1

7.3 Performance analysis

The PM and P^3M methods were implemented exactly as described in the previous sections. The PM method was developed for both CPU and GPU architectures, using C++ and CUDA C++, respectively, while the P^3M method is currently available only in the CPU variant. The implementation relies on external libraries for fast Fourier transform computations: FFTW for the CPU version and cuFFT for the GPU version. A performance comparison of the PM method was conducted using $N = 2^{16} = 65,536$ particles on a $64 \times 64 \times 64$ mesh with the CIC assignment scheme. The tests were run on a system equipped with an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz and an NVIDIA GeForce GTX 1650 GPU. Over 200 iterations, the CPU implementation consistently took around 2.3 seconds, while the GPU implementation reduced this to approximately 1.0 second (excluding data transfer time between host and device). Notably, the most time-consuming part of the program was unrelated to computation—writing data to disk in text format took nearly 20 seconds.

For the P^3M method, performance was measured using $N = 50,000$ particles on a $128 \times 128 \times 64$ mesh with the TSC assignment scheme. The total runtime was approximately 1 minute and 30 seconds, with the time distribution among key algorithm components as follows:

- HOC table initialization: 12%
- Short-range force calculations: 80%
- PM step: 7.5%

The code is available at `https://github.com/AleksyBalazinski/ParticleSimulation` under the MIT license.

References

[1] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, Berlin, Germany, 6 edition, 2018.

[2] I. M. Gelfand and G. E. Shilov. *Generalized Functions, Vol. 1: Properties and Operations*. Academic Press, 1964. Originally published in Russian; this volume introduces the theory of generalized functions and includes topics such as Fourier transforms, homogeneous distributions, and distributions on submanifolds.

[3] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. CRC Press, 1st edition, 1988.

[4] Andrey Kravtsov. Writing a pm code, March 2002. Accessed: 2025-04-03.

[5] John R. Taylor. *Classical Mechanics*. University Science Books, 2005.

[6] M. Trenti and P. Hut. Gravitational n-body simulations, 2008.

[7] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree n-body algorithm. In *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, Supercomputing '93, page 12–21, New York, NY, USA, 1993. Association for Computing Machinery.

[8] J. Weiss. Certain aspects of the gravitational field of a disk. *Applied Mathematics*, 9:1360–1377, 2018.

[9] Peter Young. Leapfrog method and other "symplectic" algorithms for integrating newton's laws of motion. `https://courses.physics.ucsd.edu/2019/Winter/physics141/Assignments/leapfrog.pdf`, 2019. Physics 115/242 Course Notes, UC San Diego.