The N-body Method I

In principle, an N-body simulation is an attempt to solve the collisionless Boltzmann equation (CBE),

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla f - \nabla \Phi \cdot \frac{\partial f}{\partial \vec{v}} = 0$$

in concert with the Poisson equation

$$\nabla^2 \Phi(\vec{x}, t) = 4\pi G \int f(\vec{x}, \vec{v}, t) \, d^3\vec{v}$$

The direct, 6-dimensional integration of $f$ is usually impractical.

Since typically $N_{\text{simulation}} \neq N_{\text{real}}$ by a large factor, in the N-body approach one follows the orbits of representative mass elements, a.k.a. particles. Hence,

$$\ddot{\vec{x}}_i = \sum_{j \neq i}^N -\nabla \phi(\vec{x}_i, \vec{x}_j)$$

where $\phi(\vec{x}_i, \vec{x}_j)$ is the gravitational potential between two point masses $i$ and $j$ are NOT the equations of motion that we should be solving.

The CBE is a first order non-linear partial differential equation. Such equations can be solved by the method of characteristics. The characteristic of a partial differential equation is the path in the independent variables along which information propagates.
For the CBE, the characteristics are defined by

\[
\frac{d\vec{x}}{dt} = \vec{v} \\
\frac{d\vec{v}}{dt} = -\nabla \Phi
\]

These are the equations of motion of a particle in a potential $\Phi$. Furthermore, we know that $f$ is constant along the characteristics. Therefore, each particle can be considered as carrying a piece of $f$ as it follows its trajectory.

The only difficulty is in the evaluation of $\Phi(\vec{x}, t)$. In terms of the distribution function,

\[
\Phi(\vec{x}, t) = -GM \int d^6 \vec{w}' \frac{f(\vec{w}', t)}{|\vec{x} - \vec{x}'|}
\]

This integral can be done by Monte Carlo integration. This is based on the result that for any reasonable function $g(\vec{w})$,

\[
\int d^6 \vec{w} g(\vec{w}) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \frac{g(\vec{w}_i)}{f_s(\vec{w}_i, t)}
\]

where the points $\vec{w}_i$ are randomly chosen with a sampling probability density, $f_s(\vec{w})$. 
Applying this to the Poisson integral,

$$\Phi(\vec{x}, t) \simeq -\frac{GM}{N} \sum_{i=1}^{N} \frac{f(\vec{w}_i, t)/f_s(\vec{w}_i, t)}{|\vec{x} - \vec{x}_i|}$$

where now $f_s$ could be time dependent. The gravitational potential is generated by particles that have masses

$$m_i = \frac{M}{N} \frac{f(\vec{w}_i, t)}{f_s(\vec{w}_i, t)}$$

In a conventional N-body simulation, $f_s(\vec{w}, t) = f(\vec{w}, t)$, so the particle density represents the underlying phase space density. Then

$$\Phi(\vec{x}, t) \simeq -\frac{GM}{N} \sum_{i=1}^{N} \frac{m_i}{|\vec{x} - \vec{x}_i(t)|}$$

Since $f_s(\vec{w}, t) = f(\vec{w}, t)$ one is tempted to imagine that the sampling points are real stars (or dark matter particles), but in reality one must remember that we are just integrating a partial differential equation, the CBE:

- An N-body simulation is an algorithm for **Monte Carlo simulation** of the CBE.
- The particles are tracers of the density field that are used to simultaneously solve for the gravitational potential and sample the phase-space density.
Schematically a N-body code proceeds as follows:

(1) Start with some initial positions and velocities.

(2) Update the positions.

(3) Compute the gravitational accelerations from the particle distribution.

(4) Update the velocities.

(5) Repeat.
Direct Techniques

One determines the gravitational acceleration by directly summing over all particle pairs.

This method scales as $O(N^2)$.

Here is an example using leap frog to advance particle $i$ one timestep, $\Delta t$:

\[
\begin{align*}
\vec{x}_i(t + 0.5\Delta t) &= \vec{x}_i(t) + 0.5\Delta t\vec{v}_i(t) \\
\vec{a}_i(t + 0.5\Delta t) &= \sum_{j,j \neq i} Gm_j (\vec{x}_i - \vec{x}_j) / |\vec{x}_i - \vec{x}_j|^3 \\
\vec{v}_i(t + \Delta t) &= \vec{v}_i(t) + \Delta t\vec{a}_i(t + 0.5\Delta t) \\
\vec{x}_i(t + \Delta t) &= \vec{x}_i(t + 0.5\Delta t) + 0.5\Delta t\vec{v}_i(t + \Delta t)
\end{align*}
\]
Actually, the Poisson integral is not well suited to Monte Carlo integration because of the singularity at $\vec{x} = \vec{x}'$. This causes a very large scatter in the estimation of $\Phi$ when a sampling point falls close to the singularity. This scatter produces a fluctuation in the potential, $\delta \Phi$, that has two effects:

1. The change in a particle’s energy along its orbit is
   \[ \frac{dE}{dt} = \frac{\partial E}{\partial x_i} \dot{x}_i + \frac{\partial E}{\partial v_i} \dot{v}_i + \frac{\partial E}{\partial t} \]
   However, remember that Hamilton’s equations tell us $\dot{x}_i = \frac{\partial H}{\partial v_i}$ and $\dot{v}_i = -\frac{\partial H}{\partial x_i}$, so the only term that remains is $\frac{\partial E}{\partial t}$, and the explicit time dependence of $E$ is the $\delta \Phi$ from fluctuations in the potential. Hence a particle’s energy changes as
   \[ \frac{dE}{dt} = \frac{\partial \Phi}{\partial t} \]
   Fluctuations in $\Phi$ owing to discrete sampling break the energy conservation of a particle. This random walk in energy is the same as two-body relaxation.

2. If different mass particles are present, the less massive particles will typically recoil from an encounter with more velocity than the massive particle causing mass segregation with the massive particles falling deeper into the potential well.
The problem of noise in estimating the potential is exacerbated because gravity causes any density fluctuations to grow.

The above effects can be lessened by introducing gravitational softening into the inter-particle potential. Note that all N-body techniques suffer from two body relaxation, although for some (particularly the FFT and multipole methods) the softening is introduced by the numerical technique itself, making the effects of two-body encounters less apparent.

One common form of gravitational softening is called Plummer Softening

\[
\sum_{j, j \neq i} Gm_j \frac{\vec{x}_i - \vec{x}_j}{|\vec{x}_i - \vec{x}_j|^3} \rightarrow \sum_{j, j \neq i} Gm_j \frac{\vec{x}_i - \vec{x}_j}{(|\vec{x}_i - \vec{x}_j|^2 + \epsilon^2)^{3/2}}
\]

where \(\epsilon\) is the gravitational softening length. Another form is spline softened gravity

\[
\vec{a}_i = \sum_{j, i \neq j} Gm_j \vec{r}_{ij} \left\{ \frac{1}{|\vec{r}_{ij}|^3} \left[ -\frac{1}{15} + \frac{8}{3} u^3 - 3u^4 + \frac{6}{5} u^5 - \frac{1}{6} u^6 \right] \right\}_{0 \leq u \leq 2}
\]

where \(u = |\vec{r}_{ij}|/\epsilon\) and \(\vec{r}_{ij} = \vec{x}_i - \vec{x}_j\).
There are actually two reasons for introducing gravitational softening:

(1) To reduce the effects of two-body relaxation and better approximate a collisionless system. The relaxation time

$$\tau_r = 0.34 \frac{\sigma_{1d}^3}{G^2 m_p \rho \ln \Lambda}$$

$$\approx 3.9 \text{ Gyr} \left( \frac{\sigma_{1d}}{100 \text{ km s}^{-1}} \right)^3 \frac{h^{-2}}{\ln \Lambda} \frac{10^{11} M_\odot}{m_p} \frac{(170 \rho_c)}{\rho}$$

where $\ln \Lambda$ is the Coulomb logarithm with, for Plummer Softening,

$$\Lambda \approx \frac{R_h}{4\epsilon}$$

and $R_h$ is the half mass radius.

(2) To allow larger timesteps to be used to decrease the run time.

Since $\epsilon$ only enters logarithmically the first reason is practically often the most important.
Variable Timesteps

One can add variable timesteps to increase computational efficiency. Remember, however, that this destroys the simplecticness of a leap frog integrator although it is still possible to retain second order accuracy.

There are two common choices for implementing variable timesteps:

(1) One can use power of two subdivisions of a largest allowed timestep. This allows the system to be resynchronized every largest timestep.

(2) One can allow particles to all have different timesteps without restrictions.

Aarseth codes do this and in addition use high order integrators. This combination improves the effective scaling of direct techniques from $O(N^2)$ to $O(N^{1.7})$ (Aarseth 1985).
Testing the Validity of Results

- Check energy, linear momentum and angular momentum conservation.
- Try running on problems with known solutions. A necessary but not sufficient condition.
- Try increasing the number of particles. Is the result the same? A necessary but not sufficient condition.
- Try decreasing the timestep. Is the result the same? A necessary but not sufficient condition.
- Try altering the gravitational softening.
- Try comparing with other codes.
- Follow the N-body constitution!
Preamble

We, the people of the N-body community, in order to perform more perfect N-body simulations, establish justice, insure domestic tranquility, provide for the common defense, promote the general welfare, and secure the blessings of scientific accuracy to ourselves and our posterity do ordain and establish this Constitution for the conduct of numerical N-body simulations.

Article I.

On the Gravitational Softening Length

Section 1.—The gravitational Softening length should be chosen large enough to minimize the effects of two body relaxation.

The softening length should be chosen such that there are at least 8 particles in each softening volume in objects of interest.
Section 1---Timesteps should be chosen to be small enough to eliminate the effects of two body scattering introduced through integration errors. The timestep must satisfy

\[ \Delta t < \frac{1}{20} \left( \frac{1}{G\rho} \right)^{1/2} . \]

For an isothermal sphere this is equivalent to

\[ \Delta t < .19 \frac{\varepsilon_{\text{grav}}}{v_{\text{max}}} \]

for a standard Plummer softening. For a typical cosmological simulation

\[ N_{\text{steps}} = \frac{T_{\text{HUBBLE}}}{\Delta t} \sim 9000 \frac{10\text{kpc}}{\varepsilon_{\text{grav}}} . \]

Spline kernel softening requires 33% more timesteps.

Usually sufficient to use the constraint

\[ \Delta t < .19 \sqrt{\frac{\varepsilon_{\text{grav}}}{a_{\text{max}}}} \]
Article III
On the Number of Particles

Section 1---The number of particles should be large enough that random potential fluctuations do not dominate the dynamics that one is trying to study. For example, to correctly model the internal dynamics of a galaxy halo requires at least ten million particles.

Section 2---The number of particles should be large enough that there is sufficient particle coverage in phase space near the important resonances. For example, to correctly model the internal dynamics of a disk within a galaxy halo requires at least one hundred million halo particles.
Section 1---Forces should be calculated with a maximum absolute and relative error.

The maximum relative error, $\frac{\Delta a}{a_i}$, and the maximum absolute error, $\frac{\Delta a}{a_{rms}}$, in the calculated acceleration should always be less than 0.5% and be less than the errors introduced by discreteness effects.
Article V.

On the Accuracy of the Integrator

Section 1---The integrator must retain at least second order accuracy at all times, be time symmetric, and avoid any correlated higher order error terms.

Retaining second order accuracy is particularly important when the length of the timestep is changed during the calculation.

However, one must be careful not to introduce correlated third order errors when correcting to second order accuracy.

If the integrator is not time symmetric then energy need not be conserved.
The N-Body Constitution

Article VI.
On the Size of the Simulation Volume

Section 1---The simulation volume must be large enough to correctly model all important non-linear effects.

There are mode couplings between large and small scales that increase power on small scales, which manifest themselves as filaments and greatly affect the gravitational evolution on small scales.

The diameter of the simulation for a CDM spectrum with $\sigma_8 = 0.7$ must be at least $40h^{-1}$ Mpc.

Section 2---No one object should dominate the evolution of the simulated volume in a simulation with periodic boundary conditions.

To prevent objects from tidally influencing themselves and to prevent objects from artificially lowering the mean background density, the simulation volume must be large enough to ensure that no virialized object contains more than $1/27$ the total simulated mass.
Section 3---The fundamental mode should remain well within the linear regime.

The simulated volume should be chosen to be large enough to ensure that the amplitude in the fundamental mode $\delta_1 < 0.01$ when the simulation is concluded.

Article VII.

On the Starting Redshift of the Simulation

Section 1---The simulation must start at a redshift high enough to ensure that all represented mass scales are still in the linear regime.

In particular, if the initial conditions are generated using the Zel’dovich approximation, the starting redshift must be sufficiently high to ensure that the absolute maximum $|\delta| = 1$.

For example, a CDM simulation with $\sigma_8 = 0.7$ in a cubic volume $40h^{-1}$ Mpc on a side simulated with $512^3$ particles would require a starting redshift of at least $z = 200$. 
Article VIII.
What Ratification Shall Establish Constitution

Section 1—This constitution must be ratified by the representatives of at least two different research groups.

The effects of violating the above articles are not known. Some violations will introduce unphysical dissipative effects, others will introduce errors that act like an artificial heat source, while the effects of other violations are not clear but are equally unphysical.

To ensure scientific accuracy all N–body simulations should at least meet the above guidelines.
Approximate Methods

Since the direct N-body method scales as \( O(N^2) \) when \( N \) becomes large enough the simulation becomes computationally impracticable.

To perform larger simulations people have developed the following methods to approximate the potential, and hence the forces, which scale better than \( O(N^2) \):

- **Multipole Expansion Codes**: the potential is expanded in Spherical Harmonics. Sometimes a radial basis is also included.

- **Particle-Mesh (PM) and Particle-Particle Particle-Mesh (P\(^3\)M) codes**: the particles are smoothed onto a regular grid and the potential is determined using a Fast Fourier Transform (FFT). The Forces are then calculated using finite differencing.

- **Tree Codes**: the particles are placed in a tree data structure that allows one to group together distant particles efficiently, which are then treated as one large particle. Typically one also includes Multipole moments of the grouped particles to increase accuracy and efficiency.
Since the Poisson equation is a linear equation, if we can find a complete set of orthogonal functions $\psi_n$ such that

$$\nabla^2 \psi_n = \lambda_n \psi_n$$

where orthogonal means

$$\int \psi_n^* \psi_m w(\vec{x}) d^3 \vec{x} = \delta_{nm}$$

where $w(\vec{x})$ is some weight function, then we have an eigenvalue problem and the potential for any arbitrary mass distribution is easily found. The density is decomposed into a sum over the eigenfunctions:

$$\rho(\vec{x}) = \sum_n C_n \psi_n(\vec{x})$$

where

$$C_n = \int \psi_n^* \rho(\vec{x}) d^3 \vec{x}$$

The potential is now just

$$\Phi(\vec{x}) = 4\pi G \sum_n \frac{C_n}{\lambda_n} \psi_n(\vec{x})$$
This procedure is usually only practical if the density distribution in question can be reasonably approximated by a small number of eigenfunctions. For the case of a nearly spherical mass distribution, we certainly have this hope if we look at Poisson’s equation in spherical coordinates:

\[
\frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial \psi}{\partial r} \right) + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial \psi}{\partial \theta} \right) + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 \psi}{\partial \phi^2} = \lambda \psi
\]

Using the method of separation of variables we can split this into three equations, one in each of the independent variables. To accomplish this we assume \( \psi \) will be a product of eigenfunctions of the individual equations.

\[
\psi(r, \theta, \phi) = R(r) P(\theta) Q(\phi)
\]

Substituting, we have

\[
\frac{\sin^2 \theta}{R} \frac{d}{dr} \left( r^2 \frac{dR}{dr} \right) - \lambda r^2 \sin^2 \theta + \frac{\sin \theta}{P} \frac{d}{d\theta} \left( \sin \theta \frac{dP}{d\theta} \right) = -\frac{1}{Q} \frac{d^2Q}{d\phi^2}
\]

The left-side of the equation does not depend on \( \phi \) and the right-side does not depend on \( r \) or \( \theta \), so both sides must be equal to a constant which we will call \( m^2 \).
Therefore, we have

\[ \frac{d^2 Q}{d\phi^2} = -m^2 Q \]

whose eigenfunctions are easily seen as

\[ Q(\phi) = \sum_m e^{im\phi} \]

where \( m \) must be an integer for single valued functions. Rewriting the other side of the equation, we have

\[ \frac{1}{R} \frac{d}{dr} \left( r^2 \frac{dR}{dr} \right) - \lambda r^2 = \frac{m^2}{\sin^2 \theta} - \frac{1}{P \sin \theta} \frac{d}{d\theta} \left( \sin \theta \frac{dP}{d\theta} \right) \]

Again, we can set both sides equal to a constant, \( l(l + 1) \). In terms of \( x \equiv \cos \theta \), we have

\[ \frac{d}{dx} \left[ (1 - x^2) \frac{dP}{dx} \right] - \frac{m^2}{1-x^2} P = -l(l + 1) P \]

the eigenfunctions of which are associated Legendre functions, \( P_l^m \), with eigenvalues, \( l(l + 1) \).
The combination of Legendre functions and circular functions are referred to as spherical harmonics, $Y_{l}^{m}(\theta, \phi)$, where

$$\int_{0}^{\pi} \sin \theta \, d\theta \int_{0}^{2\pi} \, d\phi Y_{l}^{m*}(\theta, \phi) Y_{l'}^{m'}(\theta, \phi) = \delta_{ll'} \delta_{mm'}$$

The eigenfunctions of the radial equation are spherical Bessel functions, $j_{l}(kr)$ with eigenvalues $-k^2$.

In practice, the radial eigenfunctions are rarely used since they do a very poor job of representing the radial density distribution of a galaxy, but since many galaxies are nearly spherical, the spherical harmonics can be of considerable practical use.

However, we can expand the radial part in any set of functions for which we can solve the radial part of Poisson’s equation.

A more practical way to proceed is to consider the potential of a spherical shell. In this case, everywhere except on the shell, we have Laplace’s equation: $\nabla^2 \Phi = 0$. The solution to this equation is very similar to Poisson’s equation except that the radial equation is now:

$$\frac{d}{dr} \left( r^2 \frac{dR}{dr} \right) - l(l + 1)R = 0.$$
The simplest solutions of the radial Laplace equation are:

\[ R(r) = Ar^l \quad \text{and} \quad Br^{-(l+1)} \]

Note that spherical Bessel functions reduce to these in the limit of large \( k \).

We expect the potential for a thin shell to be finite in its interior so it should be comprised of terms that look like:

\[ \Phi_{\text{int}}(r, \theta, \phi) = Ar^l Y_{lm}(\theta, \phi) \]

Conversely, externally the field should fall to zero hence

\[ \Phi_{\text{ext}}(r, \theta, \phi) = Dr^{-(l+1)} Y_{lm}(\theta, \phi) \]

The interior and the exterior solutions can be matched at the shell by requiring that the potential be continuous and by matching the gradient of the potential using Gauss’s theorem on a small volume around the shell.
Therefore, we can find the potential at any point by summing the exterior potentials of all shells interior to that point, and summing the interior potentials of all shells exterior to that point:

$$\Phi(r, \theta, \phi) = -4\pi G \sum_l m \frac{Y_l^m(\theta, \phi)}{2l+1} \left[ \frac{1}{r^{l+1}} \int_0^r m_l m(r') r'^{(l+2)} dr' + r^l \int_r^\infty m_l m(r') \frac{dr'}{r'^{(l-1)}} \right]$$

Note that for the exterior potential: the monopole ($l = 0$) declines as $1/r$, the dipole ($l = 1$) as $1/r^2$ and the quadrupole ($l = 2$) as $1/r^3$.

The correction to the monopole potential gets very small as $r$ becomes large.

If we expand about the center of mass, the dipole will be zero.

If the quadrupole is very large, one can have super-Keplerian falloff.
Two main approaches:

• Use spherical harmonics for angular portion of the potential and radial shells for the radial portion.
• Use spherical harmonics plus a radial basis. The radial basis can be either analytic or numerical.

Pros:

• Can use many particles as it scales as $O(N)$.
• Can reduce the amount of small scale noise by limiting the number of terms.

Cons:

• One has to assume a geometry, e.g. it is better suited to modeling one object at a time.
• It can have centering issues.
Spherical Shells

(1) Create a radial grid of spherical shells with particles assigned to each spherical grid cell. The grid can be spaced linearly or logarithmically in radius. Center the spherical shells on the system center of mass to remove dipole terms.

(2) Calculate spherical harmonics for each spherical shell with central radius \( r_\alpha \) by summing over particles in that shell.

\[
m_{lm}(r_\alpha) \sim \sum_{j \in \alpha} m_j Y_l^m(\theta_j, \phi_j)
\]

Once all the \( m_{lm}(r_\alpha) \) are calculated for each spherical shell \( \alpha \), use interpolation to determine them at any radius, \( m_{lm}(r) \).

Then to determine the potential at the position of particle \( i \)

\[
\Phi(\vec{r}_i) = -4\pi G \sum_{l=0}^{l_{\text{max}}} \sum_{m=-l}^{m=l} \frac{Y_i^m(\theta_i, \phi_i)}{2l+1} \left[ \frac{1}{r(l+1)} \int_0^r m_{lm}(r')r'^{(l+2)} dr' + r^l \int_r^\infty m_{lm}(r') \frac{dr'}{r'(l-1)} \right]
\]

(3) Determine the forces from \( \vec{a} = -\vec{\nabla} \Phi \).
Designer Basis Functions

We are not forced to use orthogonal functions. If we choose any linearly independent \( \psi_n \), then we can decompose the potential as

\[
\Phi(\vec{x}) = \sum_n C_n \psi_n(\vec{x})
\]

where the \( C_n \) satisfy

\[
\sum_n M_{mn} C_n = 4\pi \int \psi^*_m(\vec{x}) \rho(\vec{x}) d^3\vec{x}
\]

and

\[
M_{mn} = \int \psi^*_m \nabla^2 \psi_n d^3\vec{x}
\]

The \( M_{mn} \) need only be calculated once for any set of \( \psi_n \).

A reasonable scheme is to choose

\[
\psi_{klm} = F_{kl}(r) Y_{lm}(\theta, \phi)
\]

and

\[
F_{kl} = W_l(r) U_p(r)
\]

where \( W_l(r) \) is a reasonable approximation to the radial dependence and \( U_p(r) = (u(r))^p \). One example is :

\[
W_l = \frac{r^l}{(1+r)^{2l+1}}, \quad u = \frac{r^{-1}}{r+1}
\]

see Saha 1993
Given an arbitrary radial density-potential pair one can construct:

\[ \Phi(r, \theta, \phi) = \Phi_0(r)u(r)Y_{lm}(\theta, \phi) \]

and

\[ \rho(r, \theta, \phi) = \rho_0(r)u(r)Y_{lm}(\theta, \phi) \]

where \( \Phi_0 \) and \( \rho_0 \) are a known potential-density pair.

Hence, \( \rho_0 \) can be chosen to match the unperturbed density profile.

When you put these into the radial part of Poisson’s equation, you now have an eigenvalue problem for \( u(r) \), the Sturm-Liouville equation. One can then use numerical Sturm-Liouville equation solvers to get the eigenfunctions of this equation, the first of which is just a constant.

One can then truncate this series of eigenfunctions.

see Weinberg 1999
Particle Mesh Codes (PM)

Particle-Mesh (PM) codes use fast Fourier transforms (FFT’s) to calculate the gravitational accelerations. In the simplest case this is performed on a regular rectangular $M^3$ grid. The method is faster than direct techniques and scales as $O(M \ln M)$. Gravitational forces are naturally softened on about the grid scale.

Schematically the algorithm proceeds as follows:

1. Assign particle masses to the grid to determine the density at the grid points through interpolation.
2. Compute the gravitational potential at the grid points by solving the Poisson equation on the grid using FFT’s.
3. Compute gravitational accelerations at the grid points by finite-differencing the potential.
4. Calculate the acceleration at each particle position using the same interpolation scheme used in the mass assignment to ensure momentum conservation.

see Efstathiou, Davis, Frenk, & White 1985
Mass Assignment

Assign mass points to the grid using

$$\rho(\vec{n}/M) = \frac{M^3}{N} \sum_{i=1}^{N} W(\vec{x}_i - \vec{n}/M)$$

where $N$ is total number of particles, $M$ is the number of grid cells along one dimension, and $\vec{n}$ is an integer triple that defines the center of each grid cell.

$W$ defines the interpolation scheme.

Throughout our discussion of PM codes we will assume that the length of the entire grid is one and that the total mass is one.

In three dimensions $W$ can be expressed as the product of three one dimensional functions of the one dimensional displacements of a particle from the cell center $\delta \vec{x} = \vec{x} - \vec{n}/M$.

$$W_{ijk} = w_i w_j w_k$$

There is a hierarchy of $W$’s of increasing order $p$ where a mass point is assigned to $p^3$ cells.
Mass Assignment Schemes

(1) \( p = 1 \) Nearest Grid Point (NGP): the density is discontinuous across cell boundaries.

\[ w_i = 1, \quad M |\delta x_i| \leq \frac{1}{2} \]

(2) \( p = 2 \) Cloud In Cell (CIC): the density is continuous across cell boundaries but the first derivative is discontinuous.

\[ w_i = 1 - M |\delta x_i|, \quad M |\delta x_i| < 1 \]

(3) \( p = 3 \) Triangular Shaped Cloud (TSC): both the density and its first derivative is continuous across cell boundaries but higher order derivatives are discontinuous.

\[ w_i = \begin{cases} 
\frac{3}{4} - M^2 \delta x_i^2, & M |\delta x_i| \leq \frac{1}{2} \\
\frac{1}{2} \left( \frac{3}{2} - M |\delta x_i| \right)^2, & \frac{1}{2} \leq M |\delta x_i| \leq \frac{3}{2} \\
0, & M |\delta x_i| > \frac{3}{2}
\end{cases} \]
Determining the Potential

The potential at every grid point $\Phi$ is determined from the density field by solving Poisson’s equation:

$$\Phi(\vec{n}/M) = \frac{1}{M^3} \sum_{n'} G[(\vec{n}' - \vec{n})/M] \rho(\vec{n}'/M)$$

where $G$ is an approximation to the Green’s function of the Poisson equation. The convolution can be solved rapidly in Fourier space using FFT’s

$$\Phi(l, m, n) = \sum_{p,q,r} \hat{G}_{p,q,r} \hat{\rho}_{p,q,r} e^{2\pi i (pl + qm + rn)/M}$$

where circumflexes denote Fourier transforms and the subscripts are integer triples denoting a grid point.

The usual Green’s function for the Laplacian is

$$\hat{G}_{p,q,r} = \begin{cases} 
0 & p = q = r = 0 \\
-\frac{GM^2}{\pi(p^2+q^2+r^2)} & \text{otherwise}
\end{cases}$$

Another choice is the seven-point finite difference approximation to the Laplacian

$$\hat{G}_{p,q,r} = \begin{cases} 
0 & p = q = r = 0 \\
-\frac{G\pi}{M^2} \left[ \sin^2\left(\frac{\pi p}{M}\right) + \sin^2\left(\frac{\pi q}{M}\right) + \sin^2\left(\frac{\pi r}{M}\right) \right] & \text{otherwise}
\end{cases}$$
Computing the Accelerations

The accelerations are calculated by finite differencing on the grid of gravitational potential energies.

\[
\ddot{a}(\vec{n}/M) = D_n \Phi(\vec{n}/M)
\]

where the operator \(D_n\) represents the finite differencing scheme.

One choice for the finite differencing scheme is the two-point finite difference approximation

\[
a_x(\vec{n}/M) = \frac{M}{2} (\Phi_{l-1,m,n} - \Phi_{l+1,m,n})
\]

with similar terms for the other two directions.

Another choice is the four-point finite difference approximation

\[
a_x(\vec{n}/M) = \frac{2M}{3} (\Phi_{l-1,m,n} - \Phi_{l+1,m,n}) + \frac{7M}{12} (\Phi_{l-2,m,n} - \Phi_{l+2,m,n})
\]

The accelerations are interpolated to each particle location using

\[
\ddot{a}(\vec{x}_i) = \sum_n W(\vec{x}_i - \vec{n}/M) \ddot{a}(\vec{n}/M)
\]

The \(W\) used in the force interpolation must be the same as that in the mass assignment to conserve momentum.

To reduce small scale errors that can lead to poor energy conservation the Green’s function can be shaped.
To gain better force resolution at small scales one can add the direct summation (PP) of particles within a grid cell to the PM forces.

One has to choose the PM Green’s function and the PP gravitational softening to make the forces as continuous as possible.

To group together particles within a grid cell one typically uses a linked list approach.

However, the method becomes inefficient if there are too many particles within a grid cell because then it starts to scale like the direct technique as $O(N^2)$.

There are two ways that people have tried to fix this issue:

- By using a hierarchical series of grids, i.e. put a subgrid in the cells that contain many particles (Couchman 1991) so that the scaling returns to $O(M \ln M)$. These are often referred to as adaptive mesh techniques.

- By using a tree code technique (see below) within each grid cell, which makes the scaling within the cells $O(N \ln N)$ (Bode, Ostriker, & Xu 2000 ). These are often called PM-tree codes.
Comoving Coordinates I

For cosmological simulations it is expedient to work with coordinates that are comoving with the expansion of the Universe, $\vec{x}$, which are related to the proper coordinates, $\vec{r}$, and the expansion factor $a$ through

$$\vec{x} = \frac{\vec{r}}{a}$$

where $a = 1/(1 + z)$ and $z$ is the redshift.

In these comoving coordinates the momentum of a particle with mass $m$ is

$$\vec{p} = ma^2 \dot{\vec{x}}$$

From the Friedmann equations $a$ evolves according to

$$\left(\frac{\dot{a}}{a}\right)^2 - \frac{8\pi}{3} G \rho = \frac{\Lambda}{3} - \frac{k c^2}{a}$$

where $\rho$ is the mean mass density of the Universe, $\Lambda$ is the cosmological constant, and $k = 1, 0, \text{ or } -1$, which indicates a closed, flat, and open universe, respectively.
It is customary to express the mean mass density of the Universe in terms of the critical density $\rho_c$, which is the density that would result in a flat ($k = 0$) Universe if $\Lambda = 0$ by

$$\Omega \equiv \frac{\rho}{\rho_c}$$

where

$$\rho_c = \frac{3H^2}{8\pi G}$$

and $H(t) \equiv \dot{a}/a$ is the Hubble parameter.

If we employ the convention that at the present time, $t_0$, the expansion factor is $a(t_0) = 1$ then the solution for a flat universe with no cosmological constant is

$$a(t) = \left(\frac{3H_0 t}{2}\right)^{2/3}$$

In these comoving coordinates the equations of motion for the $i$-th particle are

$$\ddot{x}_i = \frac{d\dot{x}_i}{dt}$$

$$\dddot{x}_i + 2H\dot{x}_i - \frac{1}{a} \nabla \Phi_i = 0$$
One can derive Drift and Kick operators for a timestep $\tau$ to make a symplectic leap frog integrator in comoving coordinates:

$$D(\tau) = \vec{r}_i(t) + \vec{p}_i(t + .5\tau) \int_t^{t+\tau} \frac{dt}{a^2}$$

$$K(\tau) = \vec{p}_i(t) - \vec{\nabla}\Phi(t + .5\tau) \int_t^{t+\tau} \frac{dt}{a}$$

Note that it is assumed here that there is no explicit time dependence in $\Phi$, e.g. any gravitational softening must be constant in comoving coordinates. For standard cosmologies the integrals in the above operators can be easily evaluated. For example, in a critical density universe

$$\int_t^{t+\tau} \frac{dt}{a^2} = \frac{2}{H_0} \left[ a^{-1/2}(t) - a^{-1/2}(t + \tau) \right]$$

and

$$\int_t^{t+\tau} \frac{dt}{a} = \frac{2}{H_0} \left[ a^{1/2}(t + \tau) - a^{1/2}(t) \right]$$

In comoving coordinates one then makes a complete integrator by taking $K(0.5\tau)D(\tau)K(0.5\tau)$. 
Tree Codes

Let's go back to our original definition of $\Phi$:

$$\Phi(\vec{x}) = -G \int \frac{\rho(\vec{x}')}{|\vec{x}' - \vec{x}|} d^3 \vec{x}'$$

One can Taylor expand this about $\vec{x}' = 0$ and get:

$$\Phi(\vec{x}) = -G \int d^3 \vec{x}' \left[ \frac{\rho(\vec{x}')}{|\vec{x}|} + \frac{\vec{x}' \rho(\vec{x}') \cdot \vec{x}}{|\vec{x}|^3} + \frac{(3 \vec{x}'_i x_j' - |x'|^2 \delta_{ij}) \rho(\vec{x}') x_i x_j}{2|\vec{x}|^5} + \cdots \right]$$

Note that

$$\int \rho(\vec{x}') d^3 \vec{x}' = M$$

etc.

This expansion converges quickly for $|\vec{x}| >> |\vec{x}'|$. Also note that these are the Cartesian forms of the $Y_{lm}$.

This leads to the tree code principle: if a collection of particles are far away compute their potential using multipoles of the collection; if they are close by compute their potential directly.
Tree Codes In Practice

Tree codes group together interactions with distant particles using a tree data structure.

They are more efficient than direct techniques and scale as $O(N \ln N)$.

Schematically the force calculation proceeds as follows:

(1) Put the particles into a tree data structure.

(2) Calculate the properties of the tree nodes, i.e. the mass, center, size, center of mass, and multipoles (if used) of each tree node. Steps (1) and (2) together are called building the tree.

(3) Use the tree data structure to determine the interaction list for each particle. The red interaction list is composed of both particles and tree nodes. This is called walking the tree.

(4) Sum the contributions from elements of the interaction list to determine the gravitational acceleration of each particle. In addition to the direct force, tree nodes can also interact through their multipoles.

see Barnes & Hut 1986, Hernquist 1987, Hernquist 1990
There are two main types of tree geometries used:

(1) **Oct-trees**—divide space into successively smaller octants by volume, dividing space in all three dimensions at once. In three-dimensions all the **tree nodes** are cubes.

Each node in the tree has up to 8 children.
Types of Trees II

(2) **kd-trees**—divide space one dimension at a time. Each node has one or two children.

A *kd-tree* can divide space by volume. In three-dimensions tree nodes are never more elongated than 3:1.

![Diagram of kd-tree](image)
A *kd-tree* can divide space by putting an equal number of particles in each node, which is called a *balanced kd-tree*. This has the advantage that there are no empty *tree nodes*. However, this leads to *tree nodes* that are highly elongated.

Spherical mass distributions require the smallest number of multipoles to get accurate forces at a distance, e.g. perfect spheres only require the monopole term. *Oct-tree* nodes are closest to spheres and hence have the smallest errors when calculating accelerations. Hence we will restrict our further discussion to *oct-trees*. 
Oct-Tree Nodes

Each node of the oct-tree has:

- **Pointers** to 8 children. A pointer could point to another node, a particle, or null if that node contains no particles.
- A pointer to its parent node (except for the root node).
- The position of the center of the node.
- The size of the node.
- The mass of the node.
- The center of mass position of the node.
- Higher order multipoles of the node (if used).
Build the tree from the top down, one level at a time as follows ($O(N \ln N)$):

(1) Assign the root node a size and central position such that it contains all the particles. Add the root node to the active node list and add all the particles to the active particle list.

(2) Loop over the active particles and determine how many particles are in each octant of each active node.

(3) If an octant has more than one particle, create a new node and assign it a size and a position. The size is half of the size of its parent node, etc.

   (i) Set one of the parent node’s children pointers to point to this node.

   (ii) Set this node’s parent pointer to point to its parent.

   (iii) Add this node to the new active node list.

   (iv) Reset the parent pointer of the particles contained within this node to point to this node.

   (v) Add the particles to the new active particle list.
If an octant has one particle, set one of the parent node’s children pointers to point to this particle. Do not add this particle to the new active particle list.

If an octant contains no particles, set one of the parent node’s children pointers to point to NULL. Do not add this node to the new active node list.

Repeat, i.e. go back to (2), until there are no active nodes.
Calculate the mass, center of mass position, and higher order multipole moments (if used) of the nodes starting from the bottom of the tree and working your way up the tree as follows ($O(N \ln N)$):

(1) Start with the bottom of the tree, which will all be particles. Add their contributions to the smallest size nodes, i.e. the next lowest level of the tree (which were the last nodes created) to the node masses and node center of mass coordinates (and contributions to the node higher order multipole terms if used), e.g. $\sum m$ and $\sum m\vec{x}$.

(2) Continue up the tree, one level at a time, adding the up to 8 children’s contributions to the node’s mass, center of mass, and multipoles (if used).

(3) Continue until the root node is reached.

(4) Divide $\sum m\vec{x}$ by $\sum m$ for each node to determine the center of mass coordinates. Similar operations are required for the higher multipole moments (if they exist).
Walk the tree from the top down for each particle to determine its interaction list \(O(N \ln N)\):

1. Start with no particles or nodes on the interaction list and only the root node on the active node list.

2. Loop over the active node list.
   - (i) If the node does not satisfy the opening criteria then open the node, i.e. add its children that are nodes to the new active node list. Add any children that are particles to the interaction list.
   - (ii) If the node does satisfy the opening criteria then add it to the interaction list.

3. Repeat, i.e. go to step (2), until there are no active nodes.

Once the interaction list is complete then calculate the acceleration on the particle by summing over the interaction list. Both particles and nodes interact through their direct Newtonian point mass force. In addition, for nodes also add any higher order multipoles (if used).
The opening criteria determines whether or not a node is opened while creating the interaction list during the tree walking procedure.

A standard choice for the opening criteria is to take the geometric form for $\theta$, the opening angle,

$$\theta \equiv \frac{s}{d} \leq \theta_{\text{max}}$$

where

$$s = \text{size}_{\text{node}}, \quad \text{and} \quad d = |\vec{x}_i - \vec{x}_{\text{node}}|$$

This can lead to problems if the center of mass of the node is far from the center of the node.

To solve this problem one can instead:

1. Replace $s$ by $\sqrt{3}/2 \max \{|\vec{x}_j - \vec{x}_{cm,\text{node}}|\}$ where $j$ runs over all the particles in the node and replace $d$ by $|\vec{x}_i - \vec{x}_{cm,\text{node}}|$.

2. Replace $d$ by $d - |\vec{x}_{\text{node}} - \vec{x}_{cm,\text{node}}|$.
One can make the code more efficient by having groups of particles as the leaves of the tree instead of single particles. These are sometimes called buckets.

- All the particles in the same bucket share the same interaction list except that interactions with itself are excluded.
- The interaction list is constructed by walking the whole bucket at once, replacing \( d \) with \( d - \sqrt{3} \text{size}_{\text{bucket}} \) in the opening criteria.
One can include higher order multipoles for the nodes to use during gravitational interactions to make things more accurate. This also allows a larger choice for the opening angle making the code more efficient, i.e. run faster for the same accuracy.

Since the center of mass is used as the expansion center, the lowest order correction is of quadrupole order. \( Q_{ij} \) is the traceless quadrupole tensor evaluated with respect to the center of mass of the node.

\[
Q_{ij} \equiv \sum_k \left( 3x_{k,i}x_{k,j} - |\vec{x}_k|^2 \delta_{ij} \right)
\]

where the \( \vec{x} \)'s are relative to the center of mass and

\[
\vec{a}_i = \sum_j -Gm_j \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|^3} + \frac{G}{|\vec{r}_{ij}|^5} \mathbf{Q} \cdot \vec{r}_{ij} - \frac{5G}{2} (\vec{r}_{ij} \cdot \mathbf{Q} \cdot \vec{r}_{ij}) \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|^7}
\]
The quadrupole tensor of each node can be passed up the tree when the masses and center of masses are calculated using the relation

$$ Q = \sum_{l=1}^{8} Q_l + \sum_{l=1}^{8} m_l \left( 3\bar{x}_l^2 - |\bar{x}_l|^2 1 \right) $$

where the index $l$ labels the child nodes, $m_l$ is the mass of the child node, $\bar{x}_l = \bar{x}_{cm} - \bar{x}_{cm}$, $1$ is the unit matrix and $Q_l$ is the quadrupole moment tensor of the child node.

Similar formula can be derived for higher order multipoles.

If one constructs the interaction list carefully using multipoles of various orders for each item on the interactions list it is possible to devise a scheme that is $O(N)$ called the Fast Multipole Method. Although theoretically possible, this has not been accomplished efficiently in three-dimensions.
There are two main choices for the boundary conditions of N-body simulations:

(1) **Vacuum**—good for simulating isolated objects like galaxies.

(2) **Periodic**—good for simulating pieces of larger structures like in cosmological simulations.

Each N-body method has a natural boundary condition:

- Direct summation–vacuum.
- Expansion–vacuum.
- Tree–vacuum.
- PM–periodic.
- \(P^3M\)–periodic.
One can change the natural boundary conditions. For example, there are two ways one can make a tree code fully periodic:

(1) By using Ewald summation (see Hernquist, Bouchet, & Suto 1991)
(2) Embed the tree code within a PM code (see Bode, Ostriker, & Xu 2000, GADGET3).

There are also two ways to make a PM or P$^3$M code have a vacuum boundary:

(1) By padding, i.e. adding zeros around the volume in effect making the calculation 8 times larger.
(2) Introducing a series of negative mass charges on the edges of the box; used in codes that use mesh refinement (see Couchman 1991).