18.337 Parallel Scientific Computing	February 11, 1995
Lecture 5	
Topic: N-body Simulation	*1995 Scribe: Peter Godman*

1 Introduction

Imagine we want to model the basic mechanics of our solar system. We would probably start with the sun, somehow representing its mass, velocity, and position. We might then add each of the nine planets in turn, recording their own masses, velocities, and positions at a point in time. Let's say we add in a couple of hundred of the larger asteroids, and a few of our favorite comets. Now we set the system in motion. Perhaps we would like to know where Pluto will be in a hundred years, or whether a comet will hit us soon. To solve Newton's equations directly with more than even two bodies is intractably difficult. Instead we decide to model the system using discrete time intervals, and computing at each time interval the force that each body exerts on each other, and changing the velocities of the bodies accordingly. This is an example of an N-body problem. To solve the problem in a simple way requires $O(n^2)$ time. With some considerable effort, we can reduce this to $O(n)$ [1]. A relatively simple algorithm can compute movement in $O(n * log(n))$ time.[2]

2 Outline

- Formulation and applications
- "Easiest part": the Euler method to move bodies.
- Direct methods for force computation.
- Hierarchical methods *(Barnes-Hut, Appel)*

3 What is N-Body Simulation?

We take n bodies (or particles) with state describing the initial position $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \Re^k$ and initial velocities $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \Re^k$.

We want to simulate the evolution of such a system, i.e., to compute the trajectories of each body, under an interactive force: the force exerted on each body by the whole system at a given point. For different applications we will have different interaction forces, such as gravitational or Coulombic forces. We could even use these methods to model spring

[1] We accomplish this using the multipole algorithm described in the next handout

[2] The Barnes-Hut Algorithm, described later

systems, although the advanced methods, which assume forces decreasing with distance, do not work under these conditions.

4 Examples

- **Astrophysics**: The bodies are stars or galaxies, depending on the scale of the simulation. The interactive force is gravity.
- **Plasma Physics**: The basic particles are ions, electrons, etc; the force is Coulombic.
- **Molecular Dynamics**: Particles are atoms or clusters of atoms; the force is electrostatic.
- **Fluid Dynamics**: Vortex method where particle are fluid elements (fluid blobs).

Typically, we call this class of simulation methods, the **particle methods**. In such simulations, it is important that we choose both spatial and temporal scales carefully, in order to minimize running time and maximize accuracy. If we choose a time scale too large, we can lose accuracy in the simulation, and if we choose one too small, the simulations will take too long to run. A simulation of the planets of the solar system will need a much larger timescale than a model of charged ions. Similarly, spatial scale should be chosen to minimize running time and maximize accuracy. For example, in applications in fluid dynamics, molecular level simulations are simply too slow to get useful results in a reasonable period of time. Therefore, researchers use the vortex method where bodies represent large aggregates of smaller particles. Hackney and Eastwood's book **Computer Simulations Using Particles,**, McGraw Hill (1981), explores the applications of particle methods applications, although it is somewhat out of date.

5 The Basic Algorithm

Figure 1 illustrates the key steps in n-body simulation. The step of collecting statistical information is application dependent, and some of the information gathered at this step may be used during the next time interval.

We will use gravitational forces as an example to present N-body simulation algorithms. Assume there are n bodies with masses $m_1, m_2, \ldots, m_n$, respectively, initially located at $\vec{x_1}, ..., \vec{x_n} \in \Re^3$ with velocity $\vec{v_1}, ..., \vec{v_n}$. The gravitational force exert on the ith body by the jth body is given by

$$\vec{F_{ij}} = G\frac{m_i m_j}{r^2} = G\frac{m_i m_j}{|\vec{x_j} - \vec{x_i}|^3}(\vec{x_j} - \vec{x_i}),$$

where G is the gravitational constant. Thus the total force on the ith body is the vector sum of all these forces and is give by,

$$\vec{F_i} = \sum_{j \neq i} \vec{F_{ij}}.$$

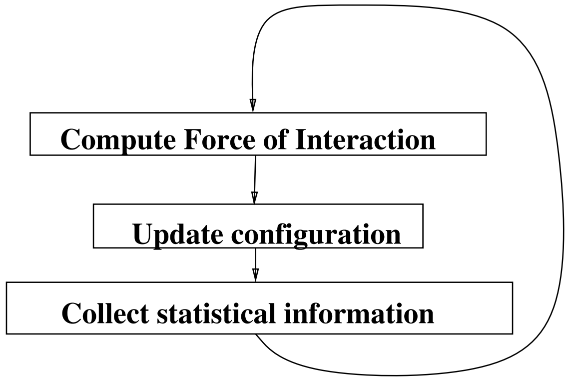

Figure 1: Basic Algorithm of N-body Simulation

Let $\vec{a_i} = d\vec{v_i}/dt$ be the acceleration of the body i, where where $\vec{v_i} = d\vec{x_i}/dt$. By Newton's second law of motion, we have $\vec{F_i} = m_i\vec{a_i} = m_i d\vec{v_i}/dt$.

5.1 Finite Difference and the Euler Method

In general, the force calculation is the most expensive step for N-body simulations. We will present several algorithms for this later on, but first assume we have already calculated the force $\vec{F_i}$ acting one each body. We can use a numerical method (such as the Euler method) to update the configuration.

To simulate the evolution of an N-body system, we decompose the time interval into discretized time steps: $t_0, t_1, t_2, t_3, ...$ For uniform discretizations, we choose a Δt and let $t_0 = 0$ and $t_k = k\Delta t$. The Euler method approximates the derivative by finite difference.

$$\begin{aligned} \vec{a_i}(t_k) &= \vec{F_i}/m_i = \frac{\vec{v_i}(t_k) - \vec{v_i}(t_k - \Delta t)}{\Delta t} \\ \vec{v_i}(t_k) &= \frac{\vec{x_i}(t_k + \Delta t) - \vec{x_i}(t_k)}{\Delta t}, \end{aligned}$$

where $1 \le i \le n$. Therefore,

$$\vec{v_i}(t_k) = \vec{v_i}(t_{k-1}) + \Delta t(\vec{F_i}/m_i) \tag{1}$$

$$\vec{x_i}(t_{k+1}) = \vec{x_i}(t_k) + \Delta t \vec{v_i}(t_k). \tag{2}$$

From the given initial configuration, we can derive the next time step configuration using the formulae by first finding the force, from which we can derive velocity, and then position, and then force at the next time step.

$$\vec{F_i} \to v_i(t_k) \to x_i(t_k + \Delta t) \to \vec{F_{i+1}}.$$

High order numerical methods can be used here to improve the simulation. In fact, the Euler method that uses uniform time step discretization performs poorly during the

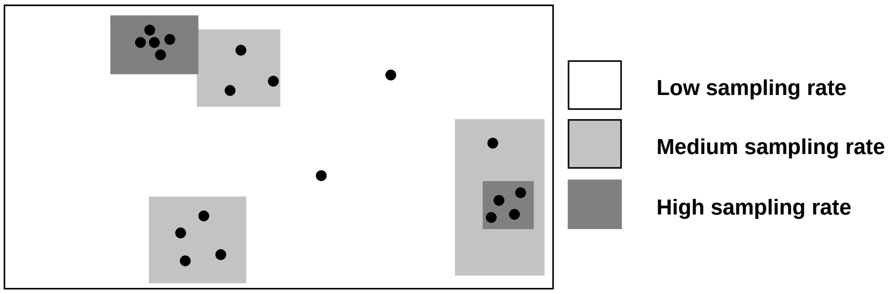

Figure 2: Adaptive Sampling Based on Proximity

simulation when two bodies are very close. We may need to use non-uniform discretization or a sophisticated time scale that may vary for different regions of the N-body system.

In one region of our simulation, for instance, there might be an area where there are few bodies, and each is moving slowly. The positions and velocities of these bodies, then, do not need to be sampled as frequently as in other, higher activity areas, and can be determined by extrapolation. See figure 2 for illustration.[3]

How many floating point operations (flops) does each step of the Euler method take? The velocity update (step 1) takes $2n$ floating point multiplications and one addition and the position updating (step 2) takes 1 multiplication and one addition. Thus, each Euler step takes $5n$ floating point operations. In Big-O notation, this is an $O(n)$ time calculation with a constant factor 5.

Notice also, each Euler step can be parallelized without communication overhead. In data parallel style, we can express steps (1) and (2), respectively, as

$$\begin{aligned} V &= V + \Delta t(F/M) \\ X &= X + \Delta t V, \end{aligned}$$

where V is the velocity array; X is the position array; F is the force array; and M is the mass array. V, X, F, M are $3 \times n$ arrays with each column corresponding to a particle. The operator / is the elementwise division.

6 Methods for Force Calculation

Computationally, the force calculation is the most time expensive step for N-body simulation. We now discuss some methods for computing forces.

[3] In figure 2 we see an example where we have some close clusters of bodies, and several relatively disconnected bodies. For the purposes of the simulation, we can ignore the movement of relatively isolated bodies for short periods of time and calculate more frames of the proximous bodies. This saves computation time and grants the simulation more accuracy where it is most needed. In many ways these sampling techniques are a temporal analogue of the later discussed Barnes and Hut and Multipole methods.

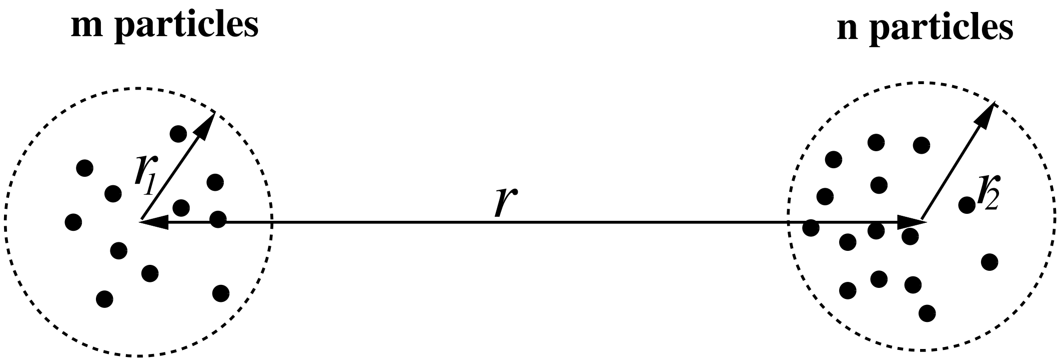

Figure 3: Well-separated Clusters

6.1 Direct force calculation

The simplest way is to calculate the force directly from the definition.

$$\vec{F_{ij}} = G\frac{m_i m_j}{r^2} = G\frac{m_i m_j}{|\vec{x_j} - \vec{x_i}|^3}(\vec{x_j} - \vec{x_i}),$$

Note that the step for computing $\vec{F_{ij}}$ takes 9 flops. It takes n flops to add $\vec{F_{ij}}$ $(1 \leq j \leq n)$. Since $\vec{F_{ij}} = -\vec{F_{ji}}$, the total number of flops needed is roughly $5n^2$. In Big-O notation, this is an $O(n^2)$ time computation. For large scale simulation (e.g., $n = 100$ million), the direct method is impractical with today's level of computing power.

It is clear, then, that we need more efficient algorithms. The one fact that we have to take advantage of is that in a large system, the effects of individual distant particles on each other may be insignificant and we may be able to disregard them without significant loss of accuracy. Instead we will cluster these particles, and deal with them as though they were one mass. Thus, in order to gain efficiency, we will approximate in space as we did in time by discretizing.

6.2 Hierarchical methods

We now discuss several methods which use a hierarchical structure to decompose bodies into clusters. Then the force field is approximated by computing the interaction between bodies and clusters and/or between clusters and clusters. We will refer this class of methods *hierarchical methods or tree-code methods.*

Suppose we have two clusters A and B, one of m particles and one of n particles, the centers of which are separated by a distance r. See Figure 3.

Suppose we want to find the force acting on all bodies in A by those in B and vice versa. A direct force calculation requires $O(mn)$ operations, because for each body in A we need to compute the force induced by every body in B.

Notice that if r is much larger than both r_1 and r_2, then we can simplify the calculation tremendously by replacing B by a larger body at its center of mass and replacing A by a larger body at its center of mass. Let M_A and M_B be the total mass of A and B, respectively. The center of mass c_A and c_B is given by

$$\begin{aligned} c_A &= \frac{\sum_{i \in A} m_i x_i}{M_A} \\ c_B &= \frac{\sum_{j \in B} m_j x_j}{M_B}. \end{aligned}$$

We can approximate the force induced by bodies in B on a body of mass m_x located at position s by viewing B as a single mass M_B at location c_B. That is,

$$F(x) \approx \frac{G m_x M_B (x - c_B)}{||x - c_B||^3}.$$

Such approximation is second order: The relative error introduced by using center of mass is bounded by $(\max(r_1, r_2)/r)^2$. In other words, if $f(x)$ be the true force vector acting on a body at x, then

$$F(x) = f(x) \left(1 + O * \left(\left(\frac{max(r_1, r_2)}{r}\right)^2\right)\right).$$

This way, we can find all the interaction forces between A and B in $O(n + m)$ time. The force calculations between one m particle will computed separately using a recursive construction. This observation gives birth the idea of hierarchical methods.

7 Quadtree (2D) and Octtree (3D) : Data Structures for Clustering

Quadtrees and octtrees provide a way of hierarchically decomposing two dimensional and three dimensional space. Consider first the one dimensional example of a straight line segment. One way to introduce clusters is to recursively divide the line as shown in Figure 4.

This results in a binary tree[4].

In two dimensions, a *box* decomposition is used to partition the space (Figure 5). Note that a box may be regarded as a "product" of two intervals. Each partition has at most one particle in it.

The root of the tree is the original box which defines the space containing the particles.

In three dimensions, a *cube* decomposition is used to partition the space (Figure 6).

[4]A tree is a graph with a single *root* node and a number of subordinate nodes called *leaves* or *children*. In a binary tree, every node has at most two children.

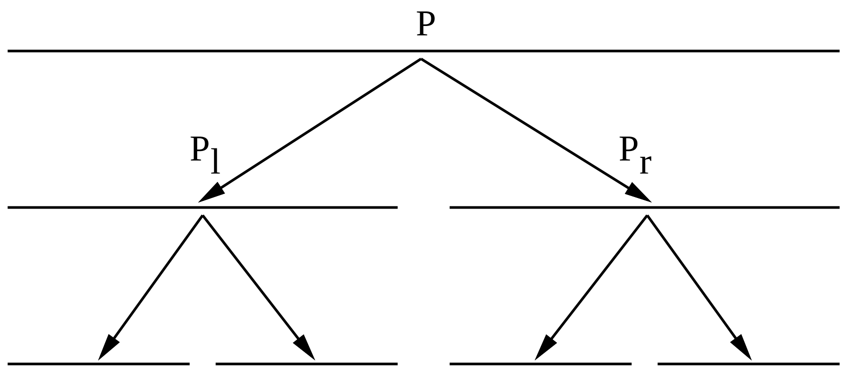

Figure 4: Binary Tree (subdivision of a straight line segment)

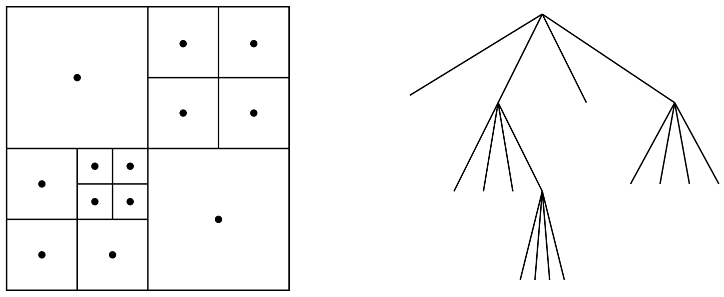

Figure 5: Quadtree

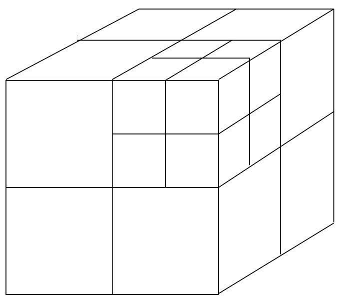

Figure 6: Octtree

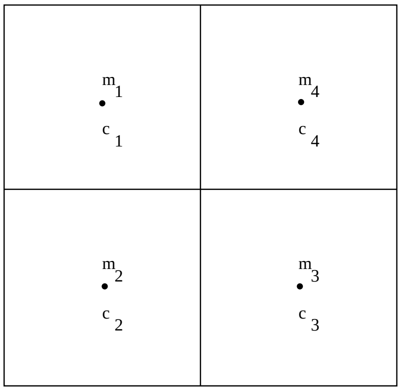

Figure 7: Computing the new Center of Mass

8 Barnes-Hut Method (1986)

The Barnes-Hut method uses these clustered data structures to represent the bodies in the simulation, and takes advantage of the distant-body simplification mentioned earlier to reduce computational complexity to $O(n * log(n))$.

The method of Barnes and Hut has two steps.

1. **Upward evaluation of center of mass**

   Refer to Figure 5 for the two dimensional case. Treating each box as a uniform cluster, the center of mass may be hierarchically computed. For example, consider the four boxes shown in Figure 7.

   The total mass of the system is

   $$m = m_1 + m_2 + m_3 + m_4 \tag{3}$$

   and the center of mass is given by

   $$\vec{c} = \frac{m_1\vec{c_1} + m_2\vec{c_2} + m_3\vec{c_3} + m_4\vec{c_4}}{m} \tag{4}$$

   The total time required to compute the centers of mass at all layers of the quadtree is proportional to the number of nodes, or the number of bodies, whichever is greater, or in Big-O notation, $O(n + v)$, where v is for *vertex*

   This result is readily extendible to the three dimensional case.

2. **Pushing the particle down the tree**
   Consider the case of the octtree i.e. the three dimensional case. In order to evaluate the potential at a point $\vec{x}_i$, start at the top of the tree and move downwards. At each

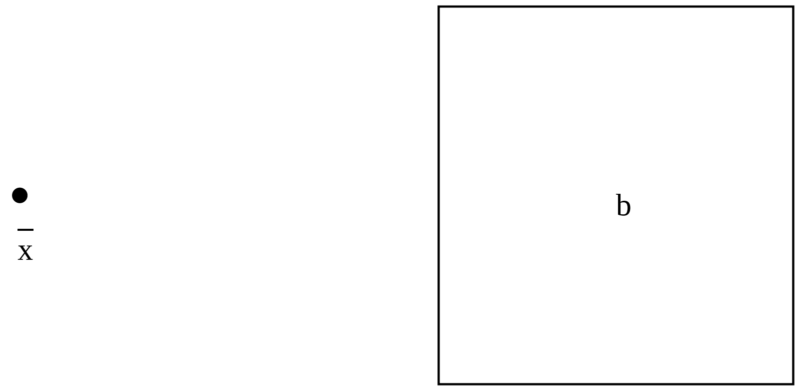

Figure 8: Pushing the particle down the tree

node, check whether the corresponding box, b, is well separated with respect to $\vec{x}_i$ (Figure 8).

Let the force at point $\vec{x}_i$ due to the cluster b be denoted by $\vec{F}(i, b)$. This force may be calculated using the following algorithm:

- *if b is "far" i.e. well separated from $\vec{x}_i$*, then

$$\vec{F}(\vec{x}_i) := \vec{F}(\vec{x}_i) + \frac{G m_x M_b (\vec{x} - \vec{c_b})}{||\vec{x}_i - \vec{c_b}||^3} \qquad \text{in } \Re^3 \tag{5}$$

- *else if b is "close" to $\vec{x}_i$*

$$\begin{aligned} &for\ k = 1\ to\ 8 \\ &\qquad \vec{F}(\vec{x}_i) = \vec{F}(\vec{x}_i) + \vec{F}(i, child(b, k)) \end{aligned} \tag{6}$$

$$\tag{7}$$

The computational complexity of pushing the particle down the tree has the upper bound $9hn$, where h is the height of the tree and n is the number of particles. (Typically, for more or less uniformly distributed particles, $h = \log_4 n$.)

9 Potential based calculation

For N-body simulations, sometimes it is easier to work with the (gravitational) potential rather than with the force directly. The force can then be calculated as the gradient of the potential.

In three dimensions, the gravitational potential at position $\vec{x}$ defined by n bodies with masses $m_1, ..., m_n$ at position $\vec{x_1}, ..., \vec{x_n}$, respectively is equal to

$$\Phi(\vec{x}) = \sum_{i=1}^{n} G \frac{m_i}{||\vec{x} - \vec{x_i}||}.$$

The force acting on a body with unit mass at position $\vec{x}$ is given by the *gradient* of Φ, i.e.,

$$F = -\nabla\Phi(x).$$

The potential function is a sum of local potential functions

$$\phi(\vec{x}) = \sum_{i=1}^{n} \phi_{\vec{x}_i}(\vec{x}) \tag{8}$$

where the local potential functions are given by

$$\phi_{\vec{x}_i}(\vec{x}) = \frac{G * m_i}{||\vec{x} - \vec{x}_i||} \qquad \text{in } \Re^3 \tag{9}$$

9.1 Approximating potentials

We can now rephase Barnes and Hut scheme in term of potentials. Let

$$\begin{aligned} m_A &= \text{total mass of particles in } \mathbf{A} \\ m_B &= \text{total mass of particles in } \mathbf{B} \\ \vec{c_A} &= \text{center of mass of particles in } \mathbf{A} \\ \vec{c_B} &= \text{center of mass of particles in } \mathbf{B} \end{aligned}$$

The potential at a point $\vec{x}$ due to the cluster **B**, for example, is given by the following second order approximation:

$$\phi(\vec{x}) \approx \frac{m_B}{||\vec{x} - \vec{c_B}||}(1 + \frac{1}{\delta^2}) \qquad \text{in } \Re^3 \tag{10}$$

In other words, each cluster may be regarded as an individual particle when the cluster is sufficiently far away from the evaluation point $\vec{x}$.

A more advanced idea is to keep track of a higher order (Taylor expansion) approximation of the potential function induced by a cluster. Such an idea provides better tradeoff between time required and numberical precision. If interested, see???. The following sections provide the two dimensional version of the fast multipole method developed by Greengard and Rokhlin.