

## Gravitational $N$ -Body Problem

Finding positions and movements of bodies in space subject to gravitational forces from other bodies, using Newtonian laws of physics.

# Gravitational $N$ -Body Problem Equations

Gravitational force between two bodies of masses  $m_a$  and  $m_b$  is:

$$F = \frac{Gm_a m_b}{r^2}$$

$G$  is the gravitational constant and  $r$  the distance between the bodies. Subject to forces, body accelerates according to Newton's 2nd law:

$$F = ma$$

$m$  is mass of the body,  $F$  is force it experiences, and  $a$  the resultant acceleration.

## Details

Let the time interval be  $\Delta t$ . For a body of mass  $m$ , the force is:

$$F = \frac{m(v^{t+\Delta t} - v^t)}{\Delta t}$$

New velocity is:

$$v^{t+\Delta t} = v^t + \frac{F \Delta t}{m}$$

where  $v^{t+\Delta t}$  is the velocity at time  $t + \Delta t$  and  $v^t$  is the velocity at time  $t$ .

Over time interval  $\Delta t$ , position changes by

$$x^{t+\Delta t} - x^t = v^t \Delta t$$

where  $x^t$  is its position at time  $t$ .

Once bodies move to new positions, forces change. Computation has to be repeated.

## Sequential Code

Overall gravitational  $N$ -body computation can be described by:

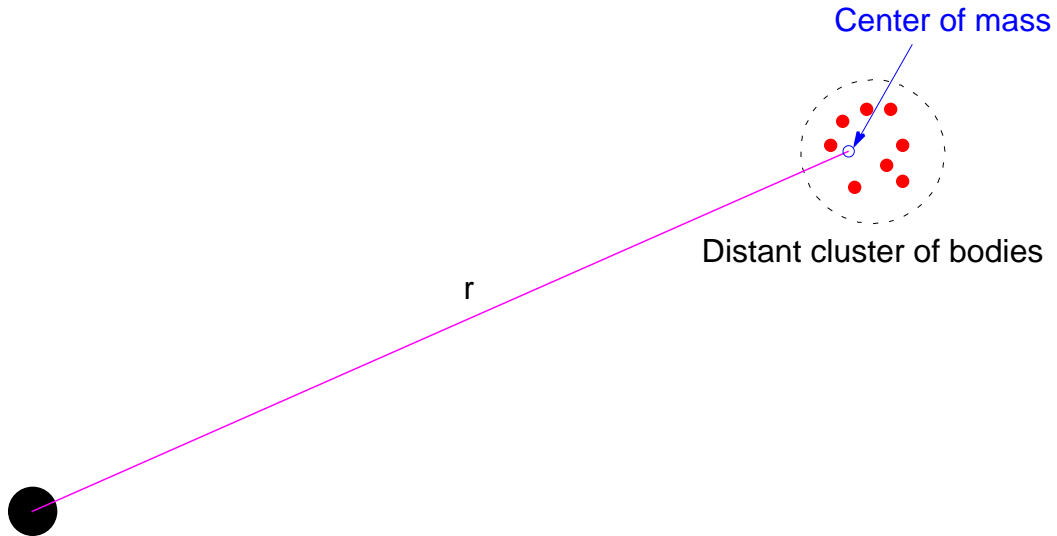
```
for (t = 0; t < tmax; t++)      /* for each time period */
  for (i = 0; i < N; i++) {    /* for each body */
    F = Force_routine(i);      /* compute force on ith body */
    v[i]_new = v[i] + F * dt / m; /* compute new velocity */
    x[i]_new = x[i] + v[i]_new * dt; /* and new position */
  }
for (i = 0; i < nmax; i++) {  /* for each body */
  x[i] = x[i]_new;           /* update velocity & position*/
  v[i] = v[i]_new;
}
```

## Parallel Code

The sequential algorithm is an  $O(N^2)$  algorithm (for one iteration) as each of the  $N$  bodies is influenced by each of the other  $N - 1$  bodies.

Not feasible to use this direct algorithm for most interesting  $N$ -body problems where  $N$  is very large.

Time complexity can be reduced using observation that a cluster of distant bodies can be approximated as a single distant body of the total mass of the cluster sited at the center of mass of the cluster:



# Barnes-Hut Algorithm

Start with whole space in which one cube contains the bodies (or particles).

- First, this cube is divided into eight subcubes.
- If a subcube contains no particles, the subcube is deleted from further consideration.
- If a subcube contains one body, this subcube retained
- If a subcube contains more than one body, it is recursively divided until every subcube contains one body.

Creates an *octtree* - a tree with up to eight edges from each node.

The leaves represent cells each containing one body.

After the tree has been constructed, the total mass and center of mass of the subcube is stored at each node.

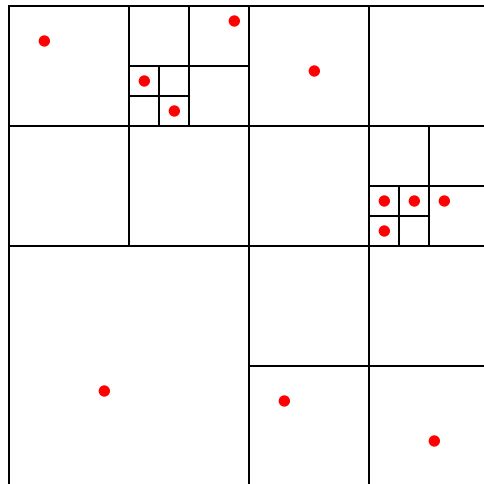
Force on each body obtained by traversing tree starting at root, stopping at a node when the clustering approximation can be used, e.g. when:

$$r \leq \frac{d}{\alpha}$$

where  $\alpha$  is a constant typically 1.0 or less.

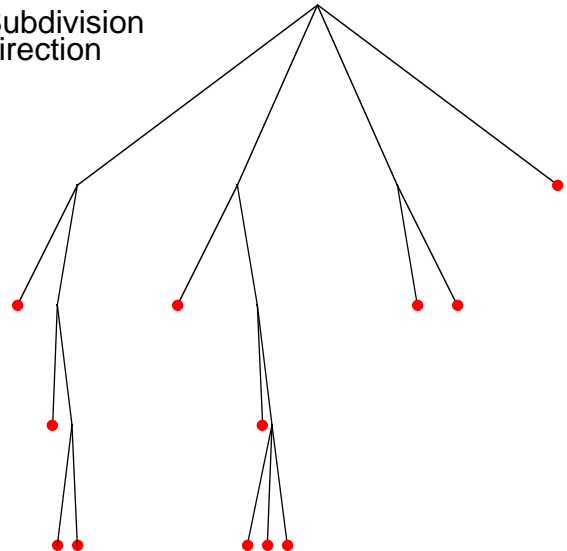
Constructing tree requires a time of  $O(n \log n)$ , and so does computing all the forces, so that the overall time complexity of the method is  $O(n \log n)$ .

# Recursive division of two-dimensional space



Particles

Subdivision  
direction



Partial quadtree

# Orthogonal Recursive Bisection

(For 2-dimensional area) First, a vertical line found that divides area into two areas each with equal number of bodies. For each area, a horizontal line found that divides it into two areas each with equal number of bodies. Repeated as required.

