N-BODY SIMULATION TECHNIQUES:

IT'S FOR EVERYBODY...

Neal Katz

The N-body Method

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

$$\frac{df(\mathbf{x},\mathbf{v},t)}{dt} = \frac{\partial f}{\partial t} + \mathbf{v}\cdot\vec{\nabla} f - \vec{\nabla}\Phi \frac{\partial f}{\partial \mathbf{v}} = 0,$$

  in concert with the Poisson equation

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

- Direct, 6-dimensional integration of f is usually impractical.

- In the N-body approach, one follows the orbits of representative mass elements, a.k.a. particles.

- Schematically:

  - Start with initial positions and velocities.
  - Update positions
  - Compute gravitational accelerations from particle distribution.
  - Update velocities.
  - Repeat.

Direct Techniques

- Determine the gravitational acceleration by directly summing over all particle pairs. Scales as $O(n^2)$.

- Example using leap frog to advance one timestep, Δt:

$$\mathbf{x}_i(t + 0.5\Delta t) = \mathbf{x}_i(t) + 0.5\Delta t \mathbf{v}_i(t)$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \Delta t \mathbf{a}_i$$

$$\mathbf{a}_i = \sum_{j, j \neq i} \frac{G m_j \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^3}, \qquad \mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$$

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t + 0.5\Delta t) + 0.5\Delta t \mathbf{v}_i(t + \Delta t)$$

- Introduce **gravitational softening**:

$$\sum_{j, j \neq i} \frac{G m_j \mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^3} \to \sum_{j, j \neq i} G m_j \frac{\mathbf{r}_{ij}}{(|\mathbf{r}_{ij}|^2 + \epsilon^2)^{3/2}}$$

- Above form is called **Plummer Softening** and ϵ is the **gravitational softening length**.

Two main reasons to include gravitational softening:

- Reduce the effects of two-body relaxation and better approximate a collisionless system.

- Allow larger timesteps to be used to decrease run time.

- Other forms for the softening can be used, for example a spline softened form of the acceleration is

$$\mathbf{a}_i = \sum_{j,i\neq j} Gm_j \begin{cases} \frac{1}{\epsilon^3}\left[\frac{4}{3} - \frac{6}{5}u^2 + \frac{1}{2}u^3\right] & 0 \leq u \leq 1 \\ \frac{1}{|\mathbf{r}|^3}\left[-\frac{1}{15} + \frac{8}{3}u^3 - 3u^4 + \frac{6}{5}u^5 - \frac{1}{6}u^6\right] & 1 \leq u \leq 2 \\ \frac{1}{|\mathbf{r}|^3} & u \geq 2 \end{cases}$$

where $u = |\mathbf{r}|/\epsilon$.

- Can add variable timesteps to increase efficiency. Destroys simplecticness of leap frog integrator but possible to retain second order accuracy.

  - Use power of two subdivisions of largest allowed timestep. System resynchronized every largest timestep.

  - Allow particles to all have different timesteps without restrictions. Aarseth codes use high order integrators, scale as $O(n^{1.7})$.

Reference: Aarseth, S.J. 1985 *Multiple Time Scales*, ed. J.W. Brackbill, B.J. Cohen, p. 377, New York: Academic.

Tree Codes

- Tree codes group together interactions with distant particles using a tree data structure. They are $O(n \ln n)$.

- Schematically the force calculation proceeds as follows:

  - Put particles into the tree structure.

  - Calculate properties of the tree nodes, e.g. the mass and center of mass of each node.

  - Determine the interaction list for each particle (walking the tree).

  - Sum the contributions from particles on the interaction list to determine the gravitational acceleration of each particle.

References:

Barnes, J. & Hut, P. 1986, *Nature* **324**, 446.

Hernquist, L. 1987, *Ap. J. Suppl.* **64**, 715.

Hernquist, L. 1990, *J. Comput. Phys.* **87**, 137.

Building Trees

- Two main types of trees used:
  - **Oct-trees**–split space into successively smaller octants by volume. Each node in the tree has up to 8 children. Splits in all three dimensions at once.
  - **kd-trees**–divide one dimension at a time, i.e. each node has one or two children. Can divide by volume or by putting a equal number of particles in each node. This is called a **balanced kd-tree**.

Restrict further discussion to oct-trees.

- Each node of the tree has:
  - pointers to 8 children (could be 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 mass of the node.
  - The center of mass position of the node.
  - Higher order multipoles of the node.

- Build the tree from top down, one level at a time as follows ($O(n \ln n)$):

  - Start with all the particles with the root node as parent, with only the root node as an *active* node and with all the particles *active*. Assign the root node a size and position such that it contains all the particles.

  - Loop over the *active* particles and determine how many particles are in each octant of each active node.

  - If an octant has more than one particle, create a node and assign it a size and a position (the size is half of the size of its parent node, etc.) Set one of the parent node's children pointers to point to this node and set this nodes parent pointer to point to its parent. Add this node to the new *active* node list. Reset the parent pointer of the particles contained within this node to point to this node. 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.

- Repeat until there are no active nodes.

- Calculate the mass, center of mass position, and higher order moments of the mass distribution from the bottom of the tree up as follows ($O(n \ln n)$):

  - Start with smallest size nodes, the lowest level of the tree (which were the last nodes created) and add their contributions to the node masses and center of mass coordinates (and contributions to higher order terms if they exist), i.e. $\sum m$ and $\sum m\mathbf{x}$.

  - Continue up the tree one level at a time until the root node is reached.

  - Divide $\sum m\mathbf{x}$ by $\sum m$ for each node to determine the center of mass coordinates (similar operations for the higher moments if they exist).

Walking Trees

- Walk the tree from the top down to determine each particles interaction list ($O(n \ln n)$):
  - Start with no particles or nodes on the interaction list and only the root node on the *active* node list.
  - Loop through the *active* node list.
  - 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.
  - If the node does satisfy the opening criteria then add it to the interaction list.
  - Repeat until there are no *active* nodes.
- Once the interaction list is made then the acceleration on the particle can be summed (including any higher order multipoles contributed by nodes, if such terms exist).

Opening Criteria

- Taking the geometric form for θ, the **opening angle**,

$$\theta \equiv s/d \leq \theta_{max}$$

where

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

is a standard choice for the opening criteria.

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

- Can make code more efficient by having groups or particles at the leaves of the tree instead of single particles (**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. Replace d with $d - \sqrt{3}\text{size}_{bucket}$ in the opening criteria.

Higher Order Multipoles

- Can include higher order multipoles for the nodes to make things more accurate. This allows a larger choice for the opening angle. Usually it is more efficient as well, i.e. faster.

- Since the center of mass is used as the expansion center, the lowest order correction is of quadrupole order. $\mathbf{Q}$ is the traceless quadrupole tensor evaluated with respect to the center of mass of the node.

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

where the $\mathbf{x}$'s are relative to the center of mass and

$$\mathbf{a}_i = \sum_j -Gm_j\frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^3} + \frac{G}{|\mathbf{r}_{ij}|^5}\mathbf{Q}\bullet\mathbf{r} - \frac{5G}{2}\left(\mathbf{r}\bullet\mathbf{Q}\bullet\mathbf{r}\right)\frac{\mathbf{r}}{|\mathbf{r}_{ij}|^7}$$

- The quadrupole tensor of each node can be passed up the tree when the masses and center of masses are calculated using the relation

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

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

- If one constructs the interaction list carefully using multipoles of various orders for each item on the list it is possible to devise a scheme that is $O(n)$ called the **Fast Multipole Method**.