Advanced N-Body Methods

**Cutoff Distances**
- PM and PPPM
- Barnes-Hut and K-D tree methods
- Recent results

---

**PP**

- The basic method we are using for Project 2 is a "particle-particle" (PP) method (aka "direct method")
  - based on pairwise interactions between bodies
  - "particle" = star, planet, billiard ball, atom, ...
- Key to accuracy: size of $\Delta t$
  - too large: "ragged simulation"
  - solar system with $\Delta t = 15552000 \approx 180$ days)?
  - too small: limits duration of simulated time
- Molecular dynamics: some phenomena require 50 fsec time step
  - protein folds require $> 1$ sec
  - $\approx 10^{12}$ time steps

---

**Cutoff Distance**

- One way to apply PP to larger problems: cutoff distances
  - define new parameter $\alpha$
  - don’t compute interactions between bodies if $||r_i - r_j|| > \alpha$
- Problem: knowing which bodies are far apart
  - want to avoid calculating distances
  - bodies move into/out of range
  - need to keep "neighbor lists" up to date

---

**Readings**

- Methods outlined in Wilkinson & Allen
- Section 12.4 of *Parallel Computing Works!* by Fox, et al
  - collection of tech reports from Caltech
  - available on-line: [http://www.npac.syr.edu/copywrite/pcw](http://www.npac.syr.edu/copywrite/pcw)
- PDFs on class web page: Dubinski et al, Wiegert et al
  - good candidates for term paper...
Cutoff Distance (cont’d)

- Cutoffs are often used in molecular dynamics
  - coulombic (electrostatic) forces are small wrt other forces
  - these forces shielded by intervening atoms
  - molecular structure helps organize bodies, locate “neighbors”
- But:
  - DNA is “highly charged” -- electrostatic forces are important
  - Paper on SIMD algorithm found about 15% error when cutoffs used
- Cutoffs are less common in astrophysics
  - Very weak forces from distant bodies have little effect
  - But number of bodies increases with distance, and total effect might be significant

PM

- An alternative to PP is Particle-Mesh (PM)
- Basically a continuous model
  - view space as a continuous force field
  - motion of a body is determined by local value of force field
- Divide space into regular mesh
- Place bodies in cells
- Compute force field at each grid point
  - Move all bodies in a cell by applying that cell’s force

PM (cont’d)

- Pro:
  - efficient algorithms for computing force field
  - FFT: $O(n \log n)$ for $n$ cells ($n$ bodies if distributed evenly)
  - compared to $O(n^2)$ for PP
- Con:
  - all bodies in a cell move the same way
  - ignores strong local interactions
  - extra work if mass not distributed evenly (empty cells)
- Has been used successfully for plasma, fluid models

PPPM

- Particle-particle/particle-mesh is a hybrid that combines best of PP and PM
- Use PP for “near force” calculations within a cell
- Compute sum of masses for each cell
- Use PM computations of “far forces” between cells
Center of Mass

- The force on body $A$ by a set of bodies $B_1, B_2, \ldots, B_n$ is the same as the force exerted by a single body $B$
  - mass of $B =$ sum of masses of $B_i$
  - position of $B =$ center of mass of $B_i$

$$M = \sum m_i \quad \vec{r}_{cm} = \frac{1}{M} \sum m_i \vec{r}_i$$

- We use this fact in the solar system simulation
  - position of a planet is the position of its center
  - ignore rotation, composition, ...
  - consider only translational movement of whole body caused by external forces

Center of Mass

- Algorithms that use this approximation have a parameter that behaves like a cutoff value
- When considering whether to use the center of mass of a group, compute $D/r$, where
  - $D =$ size of the “box” that encloses the group of particles
  - $r =$ distance to the center of mass of the group
- Use the center of mass approximation when $D/r < \theta$ where $\theta$ is a predefined value
  - usually close to (but less than) 1.0

A Case Study

- I was curious about the effects of distance and angle on the accuracy of the force calculations
- I set up a simple experiment using 10 bodies in a box
  - the masses of the bodies and their distances were based on the solar system data
- What happens to the force calculations as the starting point of body $b$ is closer to the box?
Case Study

- I did the force calculation 11 times, each time moving $b$ closer to the box
  - the columns labeled $\Delta$ are the difference in position calculated by the all-pairs method and the center-of-mass method
  - the $\Delta$s for high $r$ values are probably round-off errors (these are big numbers)
  - note what happens near $D/r = 1$...

<table>
<thead>
<tr>
<th>$r$</th>
<th>$\Delta x$</th>
<th>$\Delta y$</th>
<th>$\Delta z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1E+11</td>
<td>456.343</td>
<td>816.526</td>
<td>0</td>
</tr>
<tr>
<td>1E+10</td>
<td>456.343</td>
<td>816.526</td>
<td>0</td>
</tr>
<tr>
<td>1E+09</td>
<td>456.343</td>
<td>816.526</td>
<td>0</td>
</tr>
<tr>
<td>1E+08</td>
<td>456.343</td>
<td>816.527</td>
<td>0</td>
</tr>
<tr>
<td>1E+07</td>
<td>456.344</td>
<td>816.529</td>
<td>0</td>
</tr>
<tr>
<td>1E+06</td>
<td>456.353</td>
<td>816.557</td>
<td>0</td>
</tr>
<tr>
<td>1E+05</td>
<td>456.437</td>
<td>816.834</td>
<td>0</td>
</tr>
<tr>
<td>1E+04</td>
<td>457.283</td>
<td>819.607</td>
<td>0</td>
</tr>
<tr>
<td>1E+03</td>
<td>465.761</td>
<td>847.995</td>
<td>0</td>
</tr>
<tr>
<td>1E+02</td>
<td>547.381</td>
<td>1209.9</td>
<td>0</td>
</tr>
<tr>
<td>1E+01</td>
<td>2696.62</td>
<td>-9253.87</td>
<td>0</td>
</tr>
</tbody>
</table>

Barnes-Hut

- Center of mass gives us a way of organizing force calculations
- Arrange the bodies with a tree data structure
  - bodies are leaves
  - internal nodes: center of mass of bodies below that node
- For each body $b$:
  - traverse the tree, starting from the root
  - at a node $n$, compute $D/r$ using the distance from $b$ to the center of mass of $n$
  - if $D/r < \theta$ use the center of mass to update $b$, otherwise descend recursively through node $n$
- Examples on the next few slides...

Building the Tree

- Start by defining a square box* that encloses all the bodies

Building the Tree (cont’d)

- Recursive step: if box has more than one body, divide into four equal sub-boxes and examine each sub-box

* these examples are for 2D; easy to see how method generalizes to 3D, e.g. start with a cube
Building the Tree (cont’d)

- The outer box defines the root of the tree
- Each sub-box corresponds to an internal node

![Diagram of a tree structure]

Tree Structure

- Similar trees are used in computer graphics
  - 2D: quadtree
  - 3D: octree
- An interior node contains
  - spatial coordinates of box defined by that node
  - sum of masses of all bodies at leaves below the node
  - center of mass of the bodies
- Interior node defines a “virtual body”

Computing Forces

- After the tree is built, use it to compute forces
- A recursive algorithm computes force(b,T):
  - if T contains 0 bodies return 0
  - if T contains 1 body b’ return pairwise interaction of b, b’
  - if D/r < \theta return pairwise interaction of b with the center of mass of T
  - return the sum of force(b,Ti) where Ti are the descendants of T
- At each time step:
  - for each body b
  - compute force(b,root)

Comments on Barnes-Hut

- The tree should be rebuilt after each time step
  - moving bodies may cross cell boundaries
  - heuristic: use for N time steps before rebuilding
- Tree may not be balanced
  - in practice: very effective for simulations with clusters of bodies, especially if clusters are far apart
Time Complexity of Barnes-Hut

- The asymptotic running time at each time step depends on:
  - time to build the tree
  - time to traverse tree and compute interactions

- Building the tree: $O(n)$
  - time will be proportional to the number of nodes
  - worst case: 2 descendants per node (binary tree)
  - for $n$ bodies, $n/2$ interior nodes

\[ n + n/2 = O(n) \]

Time Complexity (cont’d)

- Time to traverse will be dominated by the number of force calculations

For large $\theta$
- worst case: no long-range forces
- each body computes $n$ pairwise interactions: $O(n^2)$

For small $\theta$
- traversal will be a path to current body
- at each interior node: 3 force calculations and one recursive call
- height of tree is $\log n$: $O(n \log n)$

K-D Tree

- Another tree building algorithm builds a more balanced tree

- Create an initial bounding box, as in Barnes-Hut

- Pick a dimension, e.g. $x$

- To divide box B along dimension $d$:
  - stop if the current box has 0 or 1 body
  - split B so half the bodies ($d$-coordinate) are in each box
  - divide each sub-box along the next dimension

K-D Tree (cont’d)

- After two steps
- Final partition
Examples

The method used by Dubinski and colleagues for the Milky Way vs Andromeda simulation:

- Application: Partree
  - parallelized tree method, written in C with MPI
- 300 million “particles”
  - 125 million stars in each galaxy
  - also 25 million particles for “dark matter halos” in each galaxy
- 5300 time steps to simulate 2.3 billion years (~.4MY/step)
- 180 seconds / time step: 11 days per simulation
- 256-node Beowulf cluster
  - dual 2.4GHz Xeon processors (512 CPUs), 1GB RAM per node
- Dubinski, et al. High Performance Commodity Networking... (PDF on class web site)

Examples (cont’d)

A newer method from Dubinski et al: GOTPM

- Grid-of-Oct-Trees / Particle Mesh
- divides space into 3D mesh, as in PM methods
- use PM technique for long-range interactions
- build an octree within each mesh cell for closer range interactions
- paper describes tests on 1G particles

Survey

- Describes requirements for solar-system scale simulations
  - asteroid and Kuiper belts, planet formation, ...
- Compares several sequential methods
  - direct (O(n^2)) methods
  - tree methods (including Partree)
  - fast multipole method (FALCON: force algorithm with complexity O(n))
    - potentially O(n), more like O(n log n) in practice