N-body methods

(*) for

e form:

$$-\sum_{\substack{j=1 \\ i \neq j}}^{N} \tilde{m}_j \left(\tilde{r}_j - \tilde{r}_i \right) \Big/ \left(\tilde{r}_{ij}^2 + \varepsilon^2 \right)^{3/2}$$

where $\tilde{r}_{ij}^2 = (x_i - x_j$

N-body methods

1) Dimensionless units
2) Methods: ideas
3) Simple scheme
4) Variable time-step

Equations for N-body problem: N objects with masses m_i

$$(*) \qquad \frac{d^2\vec{r}_i}{dt^2} = -G\sum_{\substack{j=1 \\ i\neq j}}^{N} m_j \frac{(\vec{r}_i - \vec{r}_j)}{|\vec{r}_i - \vec{r}_j|^3}$$

Potential energy $W = -\frac{1}{2}\sum_{\substack{i,j \\ i\neq j}}^{N} \frac{G m_i m_j}{|\vec{r}_i - \vec{r}_j|}$

Kinetic energy $K = \sum \frac{m_i}{2}\left(\frac{d\vec{r}_i}{dt}\right)^2$

Initial conditions: $\vec{r}_i$, $\frac{d\vec{r}_i}{dt}$

It is a good idea to use "natural" units for variables. The N-body problem does not provide a unique choice of scales. As a matter of fact, it does not have a scale.

We can use the mass of the whole system M and a radius of the system R as those scales

M = total mass; R = Radius of the system

The time-scale can be introduces as

$$t_0 = \frac{1}{\sqrt{\frac{GM}{R3}}}$$

Using M, R, and t_0 we can introduce scale-free coordinates, velocities, and masses for our particles:

$$r_i = \tilde{r}_i R, \quad m_i = \tilde{m}_i M, \quad t = \tilde{t}\, t_0, \quad v_i = \tilde{v}_i \frac{R}{t_0}$$

The equation (*) for the N-body problem can be written in the form:

$$\begin{cases} \dfrac{d^2\tilde{\vec{r}}_i}{d\tilde{t}^2} = -\sum_{\substack{j=1 \\ i\neq j}}^{N} \dfrac{\tilde{m}_j\,(\tilde{\vec{r}}_j - \tilde{\vec{r}}_i)}{(\tilde{r}_{ij}^2 + \varepsilon^2)^{3/2}}, \\ \tilde{\vec{v}}_i = \dfrac{d\tilde{\vec{r}}_i}{d\tilde{t}}, \qquad \text{where} \quad \tilde{r}_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \end{cases}$$

Here we added the term ε in the force of gravity to avoid too large accelerations.

Integration of equations of motion

Our goal is to produce a numerical method of integration of equations:

$$\begin{cases} \dfrac{d\vec{v}_i}{dt} = \vec{g}_i(\vec{x}) \\ \dfrac{d\vec{x}_i}{dt} = \vec{v}_i(t) \end{cases}, \quad \vec{x}_i = \vec{x}_i(t)$$

Along trajectory of each particle one can treat acceleration $\vec{g}_i(\vec{x}(t))$ simply as a function of time: $\vec{g}_i(t)$.
To make our arguments more transparent we omit subscripts i:

(1) $$\frac{dv(t)}{dt} = g(t), \quad \frac{dx}{dt} = v(t)$$

If we know coordinates and velocities of particles at moment t_0, we can integrate eqs (1) to find x's and v's at moment $t_1 = t + dt$:

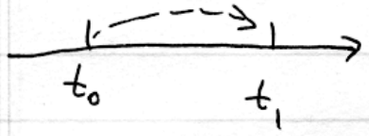
(2) $$x_1 = x_0 + \int_{t_0}^{t_1} v(t)\,dt$$
$$v_1 = v_0 + \int_{t_0}^{t_1} g(t)\,dt$$

We can use Taylor expansion around $t = t_0$ to estimate the integrals:

$$v(t) = v_0 + \left.\frac{dv}{dt}\right|_{t_0} dt + \frac{1}{2}\left.\frac{d^2v}{dt^2}\right|_0 dt^2 + \dots$$

The same is for $g(t) = g_0 + \left.\frac{dg}{dt}\right|_0 dt + \dots$

Simple (first order Euler) scheme of integration



$$x_1 \simeq x_0 + v_0 dt$$
$$v_1 \simeq v_0 + g_0 dt$$

Using the Taylor expansion we find the accuracy of these approximations:

$$x_1 = x_0 + v_0 dt + \frac{1}{2}\frac{dv}{dt}\Big|_0 dt^2 + \frac{1}{3!}\frac{d^2v}{dt^2}\Big|_0 dt^3 + \dots \simeq$$

$$\simeq x_0 + v_0 dt + \varepsilon, \text{ where } \varepsilon \approx \frac{1}{2} g dt^2 \propto O(dt^2)$$

$$v_1 = v_0 + g_0 dt + \varepsilon$$

The scheme of integration:

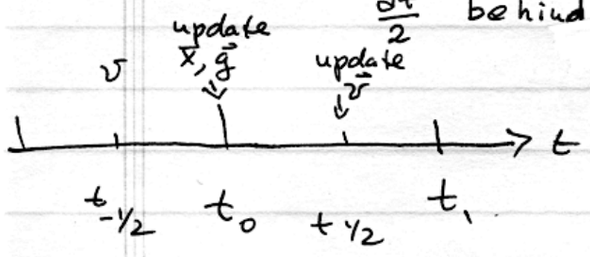
1) read initial conditions: $\vec{x}_i$ and $\vec{v}_i$ for $t = t_i$.
2) Find acceleration for each particle:
3) update coordinates and velocities; advance time: $t = t + dt$
4) go back to step 2

Note that we do not need to store vectors $\vec{x}_1$ and $\vec{v}_1$. They will replace old values $\vec{v}_0$, $\vec{x}_0$,
Accuracy of this approximation is low

$$\varepsilon \sim O(dt^2)$$

Leap-frog scheme

velocities are given at time, which is $\frac{dt}{2}$ behind the moment of coordinates



$$\begin{cases} v_{1/2} = v_{-1/2} + g_0\, dt \\ x_1 = x_0 + v_{1/2}\, dt \end{cases} \qquad (*)$$

Accuracy: we start by eliminating v's from eq(*).

$$x_1 = x_0 + v_{-1/2}\, dt + g_0\, dt^2$$

Now use $x_{-1} = x_0 - v_{-1/2}\, dt$ to eliminate $v_{-1/2}$:

$$x_1 = x_0 + (x_0 - x_{-1}) + g_0\, dt^2$$

This can be written as:

$$x_1 - 2x_0 + x_{-1} = g_0\, dt^2 \qquad (1)$$

Use the Taylor expansion for x_1 and x_{-1} up to the 4th order:

$$x_{\pm 1} = x_0 \pm v_0\, dt + g_0 \frac{dt^2}{2} \pm \frac{1}{3!}\frac{dg}{dt} dt^3 + \frac{1}{4!}\frac{d^2g}{dt^2} dt^4$$

When we substitute $x_{\pm 1}$ into (1) all terms with "$\pm$" will be zero. What is left is:

$$x_1 - 2x_0 + x_{-1} = g_0\, dt^2 + \varepsilon,$$

where error of approximation is

$$\varepsilon = \frac{1}{12}\left.\frac{d^2g}{dt^2}\right|_0 dt^4 \propto O(dt^4)$$

Stability of this approximation

Equation (1) is not exact. The error at some initial step is ε. Assuming that subsequent iterations do not make their own errors, how the initial error ε propagates through the solution? It can be shown that the error does not grow if $\left|\frac{\partial g}{\partial x}\right| dt^2 < 2$

Much better approach: variable time-step

In this case every particle moves with its own time-step. We also improve our approximation by including the first derivative of $\vec{g}$ with time:

$$\vec{g}_i = \sum_{\substack{j=1 \\ i \neq j}}^{N} \frac{m_j(\vec{r}_j - \vec{r}_i)}{(r_{ij}^2 + \varepsilon^2)^{3/2}},$$

$$\dot{\vec{g}}_i = \sum_j \frac{m_j(\dot{\vec{r}}_j - \dot{\vec{r}})}{(r_{ij}^2 + \varepsilon^2)^{3/2}} - \sum_j \frac{3m_j}{(r_{ij}^2 + \varepsilon^2)^{5/2}} (\vec{r}_i - \vec{r}_j, \dot{\vec{r}}_i - \dot{\vec{r}}_j)(\vec{r}_j - \vec{r}_i)$$

↑ scalar product

Now for each particle we store the following variables: $\vec{r}_i, \vec{v}_i, \vec{g}_i, \dot{\vec{g}}_i, t_i, \Delta t_i$. Here Δt_i is the time-step for this particle.

In order to advance the particle from moment t_i to moment $t_i + \Delta t_i$, we use:

$$\begin{cases} \vec{r}_i = \vec{r}_i + \vec{v}_i \Delta t_i + \vec{g}_i \frac{\Delta t_i^2}{2} + \dot{\vec{g}}_i \frac{\Delta t_i^3}{6} \\ \vec{v}_i = \vec{v}_i + \vec{g}_i \Delta t_i + \dot{\vec{g}}_i \frac{\Delta t_i^2}{2} \end{cases}$$

Once the particle was advanced to its new time, we must decide what time-step should it use when it will move next time. The time-step is (or can be) defined by the condition that the acceleration $\vec{g}_i$ should not change much from one moment of integration to another:

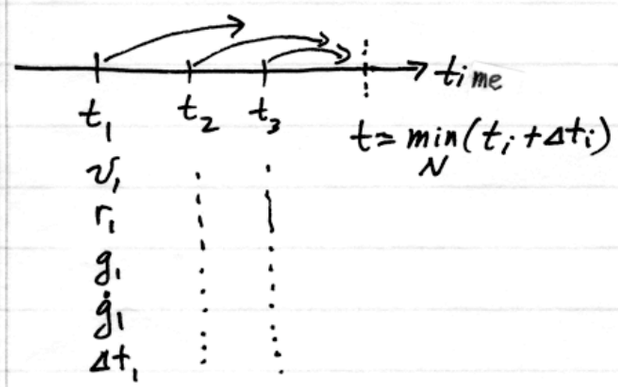
$$|\dot{\vec{g}}_i| \Delta t \ll |\vec{g}_i|.$$

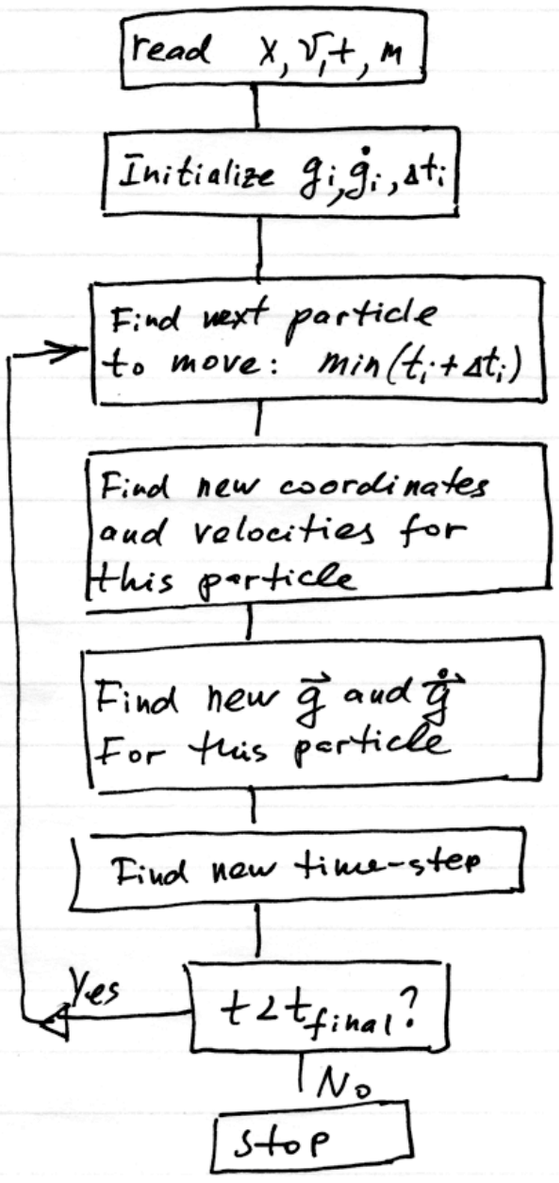
We introduce a small parameter η, which defines how small the change in the acceleration should be:

$$\Delta t_i = \eta \frac{|\vec{g}_i|}{|\dot{\vec{g}}_i|}$$

We also limit the value of the new time step to avoid too large time-step in rare situations, when $\dot{\vec{g}}_i$ gets too small: $\Delta t_i = \min\left(\eta \frac{|\vec{g}_i|}{|\dot{\vec{g}}_i|}, \Delta t_{i,old} \cdot \sqrt{2}\right)$

The algorithm for integration of N-body problem with variable time-step





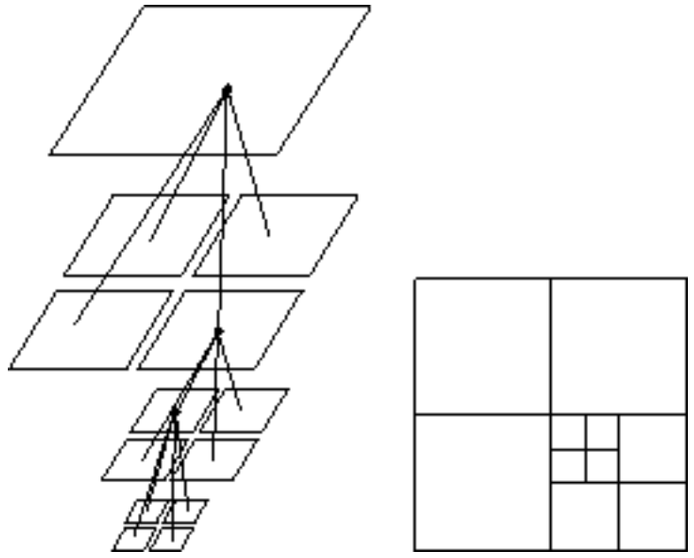
Different types of N-body codes

Finding grav. acceleration is the most time consuming part of any large N-body code. This is why code are classified by methods used to find accelerations:

1) Direct summation $\sim N^2$
2) Solve the Poisson equation on a mesh:
   $\nabla^2 U = 4\pi G\rho$
   a) need to design mesh
   b) find density on the mesh
   c) solve Poisson equation
   d) differentiate U to get $\vec{g}$
3) TRee codes:
   a) generate a hierarchical mesh
   b) multipole expantion for mass distribu-
      tion inside each cell
   c) sum contribution using only large-
      -enough cells

Simplest code of this type is called Particle-Mesh (PM): a constant-size cubic mesh is overlayed on computational volume. (1) Calculate density in each cell. (2) Solve the Poisson equation using FFT. (3)Numerically differentiate grav.potential to find acceleration for each particle. (4) Move particles by one small time-step. Repeat the procedure.

Oct-Tree



This is a 2D example of a quad-tree. Every square, which has too many particles is split into 4 squares, which have 1/2 of original size. In 3D every cubic cell is split into 8 smaller cells. If any of new cells still have too many particles, they are also split into 8 even smaller cells.

The structure is adaptive: the level of the tree depends on local density. When particles move, density changes and so does the structure.

Oct Trees are used for TREE codes and for some types of Adaptive-Mesh-Refinement (AMR) codes.

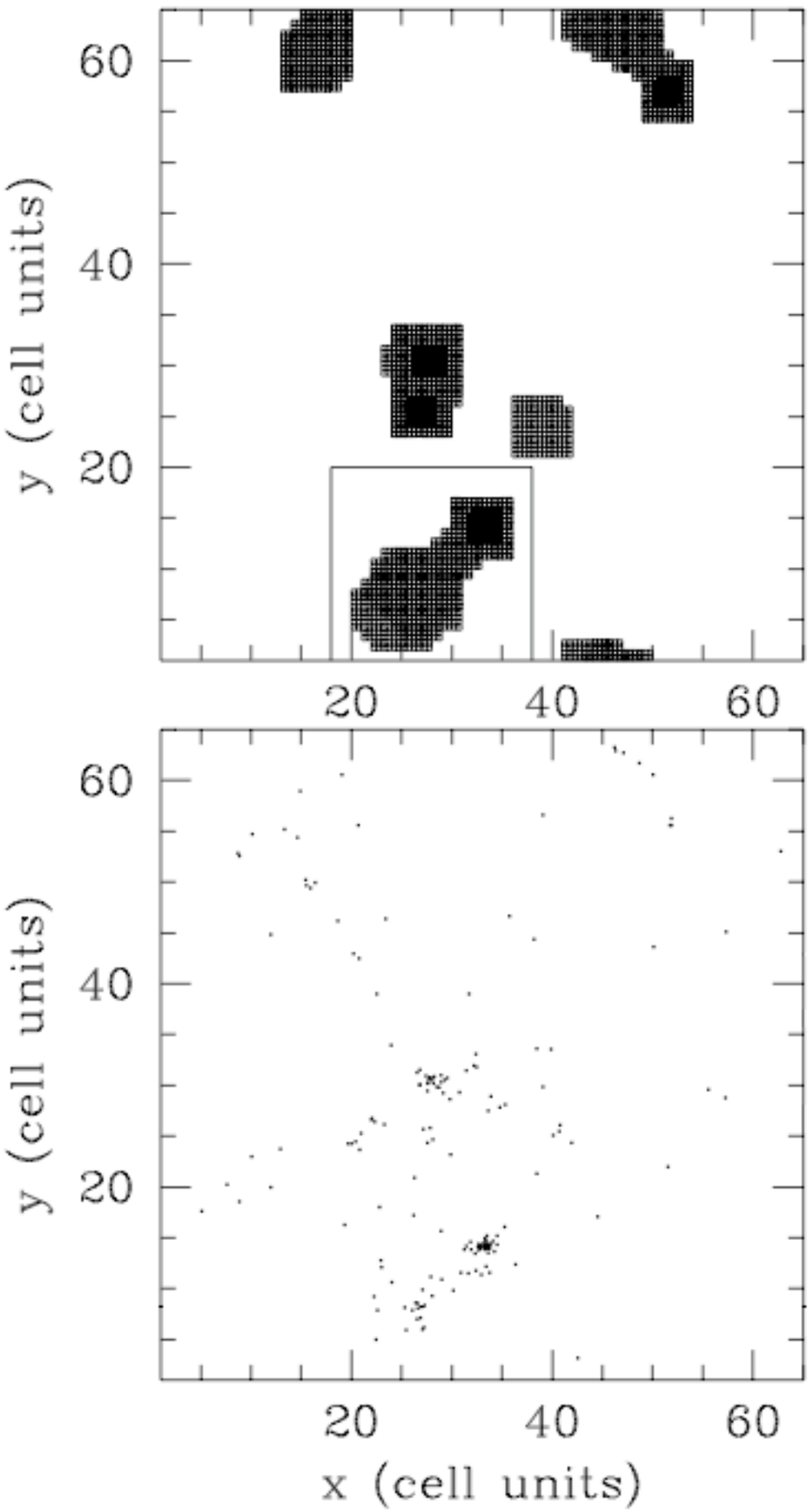
Once the structure is created, we can find grav.potential or grav acceleration using different techniques:

- Sum contributions from different nodes. Use 'opening angle' criterion to select appropriate level of refinement. This is for TREE codes.
- Solve the Poisson equation by different iterative schemes. This is for AMR codes.

Modern N-body codes are typically combinations of Particle-Mesh code (very fast) with either TREE or AMR additions for high resolution in dense environments.

Here is an example of construction of refinement structure.

The bottom plot shows a slice through a distribution of particles. The top panel shows a slice through refinement structure. For clarity only refined cells are displayed.



Nbody :

cpu requirements: $\text{cpu-time} = 5\text{sec} \times N_{steps} \times \left(\frac{N_{particles}}{10^4}\right)^2$

energy conservation : 2000 particles, $\left.\frac{E_{kin}}{E_{pot}}\right|_{init} = -0,15$

$\varepsilon = 3e-3$

$R = 1$

dt	Error in Energy_tot
$3e^{-3}$	2%
1,5e-3	1%
0,75e-3	0.5%

$t_{dyn} \simeq 0.3$

$\varepsilon = 1,5e-3$

dt	error
1,5e-3	1%

$\Rightarrow$ 1000 steps / $t_{dynamical}$

20,000 particles cpu-time/step = 18sec

30,000 particles =