Mini-Projet : Celestial Mechanics

Students :
Advisers : Ruben Di Battista and Louis Reboul

1 Introduction

In this mini-projet, we focus on the simulation of a N-body problem in celestial mechanics. The purpose of the mini-project is to combine the Barnes-Hut algorithm with a symplectic integrator in order to simulate the interaction of two galaxies.

2 Setting of the N-body problem

In physics, the N-body problem consists of the computation of all pair interactions in a system consisting of N particles. The two best-known cases are gravitational interactions, e.g. between the stars in a galaxy, and electrostatic interactions between atoms represented as point charges. The interactions can be described by a potential energy or by the forces acting on each particle, i.e. the derivatives of the potential energy with respect to the positions. The latter case is practically more relevant in simulation algorithms.

The N-body problem can be formulated as

$$F_i = \sum_{j \neq i} w_i w_j f(x_i - x_j) \tag{1}$$

where x_i are the positions of the particles, F_i is the total force acting on particle i, w_i is a parameter describing particle i (i.e. mass or charge), and $f(d_{ij})$ describes the functional form of the interactions, which depend only on the distance vector d_{ij} between two particles.

In the following, we will be interested in simulating a galaxy consisting of many bodies (stars, dusts, planets, ...). These bodies are represented by their position, their velocity and their mass. We use Newton's second law which says that mass times acceleration is equal to the total force on each mass point. The forces are computed using Newton's law of universal gravitation:

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

where m_i is the mass, x_i the position of the particle i. The distance between particles i and j is denoted $|x_j - x_i|$.

The acceleration is then given by

$$m_i \frac{d^2 x_i}{dt^2} = \sum_{j \neq i} F_{ij}. \tag{3}$$

As you can see, we have to solve a system of ODE. There are several methods to solve this equation numerically. But before we choose one of theses schemes, let us take a look at the computation of the forces.

Suppose that we have n bodies. A naive algorithm to compute the forces is of complexity n^2 !! It is a really slow algorithm. It is possible to improve the computation of the interactions between particles by using the Barnes-Hut algorithm. The idea is to divide the n bodies recursively into groups according to spatial proximity, by storing them in a quad-tree (for a 2D problem) and to say that if a group of bodies is sufficiently far away from any other bodies, we can approximate its gravitational effects by using its center of mass. The center of mass of a group of bodies is the average position of a body in that group, weighted by mass. This algorithm is of complexity $n \log n$.

2.1 Building the tree

Let's take the example given on this excellent post `http://arborjs.org/docs/barnes-hut`. We encourage you to read this article carefully to acquire a deeper understanding of the algorithm.

Figure 1: Example space.

The generated tree is the following

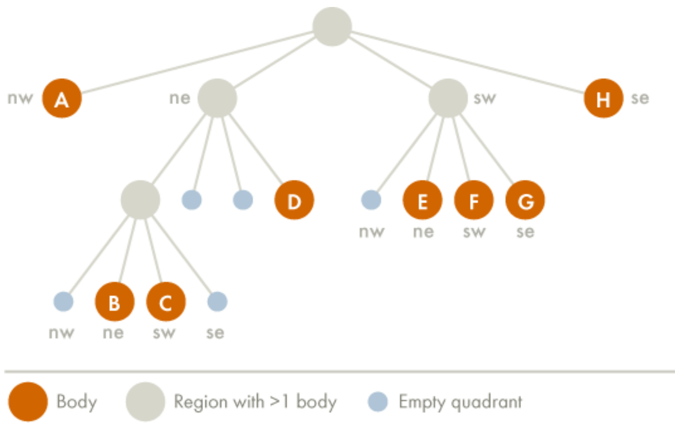

Figure 2: Example tree.

The algorithm for the construction of the tree is to insert the bodies one after another. We use a recursive procedure to insert a body b into the tree at node x.

- if the node x is empty, put the new body b here,
- if the node x is a non-empty quadrant, find the right sub-quadrant for body b and apply the procedure recursively substituting this sub-quadrant for x,
- if the node x is a body c, subdivide the region until bodies b and c are in different quadrants. Then, insert bodies b and c in the right quadrants.

The most "pythonic" way to represent a tree is to define a class representing a node which contains a list of subnodes. However, such a representation is difficult to use efficiently with Cython, Pythran or Numba. It is preferable to store the tree as an array as follows:

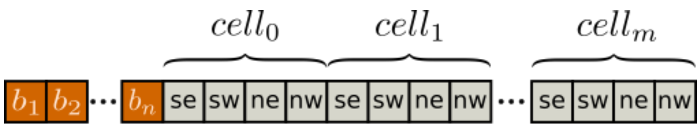

Figure 3: Tree array.

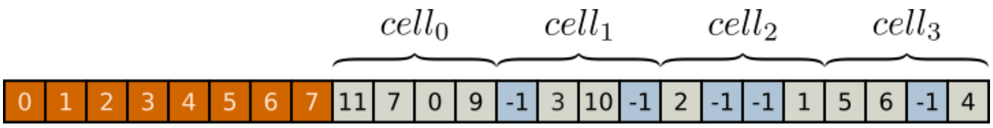

Figure 4: Example tree array.

In our example, the construction of the tree will give us the array where $0, 1, \ldots, 7$ represent the bodies $A, B, \ldots, H$, -1 means that there is no body on this quadrant. The cell index is represented by an integer greater than $n - 1$, where n is the number of bodies. To access the cell k in this array, we just have to compute the index $n + 4(k + n)$.

2.2 Calculating the mass and the center of mass

Now that we have constructed our tree, we can calculate the center of mass and the total mass of the cell using this algorithm

1. initialize the arrays 'total_mass' and 'center_of_mass' with size $n_{bodies} + n_{cell}$
2. store the mass and the coordinates of the bodies at the beginning of the arrays 'total_mass' and 'center_of_mass'
3. loop over the cells starting at the end (i.e. the top-level node)
   (a) find the elements of the cell
   (b) sum the masses of all elements that are bodies or cells
   (c) compute 'center_of_mass' as the sum of the coordinates multiplied by the mass of each element
   (d) normalize 'center_of_mass' by 'total_mass'

2.3 Calculating the forces

To calculate the total force acting on body b, we use the following recursive procedure, starting with the root of the quad-tree

1. If the current node is an external node (and it is not body b), calculate the force exerted by the current node on b using Newton's law of universal gravitation, and add this amount to b's total force.
2. Otherwise, calculate the ratio $\frac{s}{d}$. If $\frac{s}{d} < \theta$, treat this internal node as a single body, and calculate the force it exerts on body b by using the 'center_of_mass' for the position and the 'total_mass' for the mass using again Newton's law of universal gravitation, and add this amount to b's total force.
3. Otherwise, run the procedure recursively on each of the current node's children.

2.4 Solving the ODE with a standard scheme

Now that we can compute the forces of our system, we can solve the ODE by using an iterative method. Suppose that you want to solve the following ODE

$$y'(t) = f(t, y).$$

The Adam Bashforth of order 6 solves this equation numerically via the formula

$$y_{k+1} = y_k + \Delta t \sum_{j=0}^{5} a_j f(t_{k-j}, y_{k-j}), \tag{4}$$

where

$$\begin{aligned} a_0 &= \frac{4277}{1440}, \qquad a_1 = \frac{-7923}{1440}, \qquad a_2 = \frac{9982}{1440}, \\ a_3 &= \frac{-7298}{1440}, \qquad a_4 = \frac{2877}{1440}, \qquad a_5 = \frac{-475}{1440}. \end{aligned} \tag{5}$$

To initialize this scheme, we will use a Runge Kutta method of order 4 to calculate the first solution steps. In the N-body problem, the unknowns are the coordinates and the velocities of the system. Calculating the forces gives us the acceleration of the system. With this acceleration, we can calculate the velocities at the time step $k+1$. The new positions at time step $k+1$ are calculated using the velocities at time step $k, k-1, \ldots, k-5$.

3 Solar system

Using the orignal Adams-Bashforth scheme and the newly implemented composition method we have studied in the PC9 (optimized 8-15), propose a study of the simple problem of the dynamics of the solar system already implemented in the notebook and compare the two methods.

4 Galaxy interactions

Relying on the notebook provided in collaboration with Loic Gouarin and Roland Denis, provide a simulation of the interaction of two Galaxies using the two schemes and analyze the observed dynamics.

The students will also be able to rely on `http://beltoforion.de/article.php?a=barnes-hut-galaxy-simulator`.