Gravitational N-Body Simulations

Gudbrand Tandberg

October 31, 2014

1 Abstract

2 Introduction to the N-body problem

2.1 The equations

2.2 A brief history of N-Body simulations

3 The first NBodySolver class

4 Introduction to the numerical methods

4.1 Runge Kutta 4

4.2 Verlet method

5 Results for the solar system

5.1 Three Body Problem

OpenGL graphics of SEM-system.

5.1.1 Stability and accuracy

5.1.2 Energy considerations

6 Extending NBodySolver to allow for different timesteps

7 Refined results for the extended solar system

Discussion on the efficiency of this the solver. OpenGL graphics of inner Solar System (sun,...,callisto).

8 Further development of the algorithm

8.1 Parallellizing the code

8.2 Smoothing factor

8.3 Faster gravity evaluations

9 Application: cold cluster collapse

9.1 Introduction

9.2 Results

OpenGL animation, discussion on t_{crunch}, virial energy computation, different smoothing &c.

10 Conclusion

11 Logbook

8.10.2014. Initialized git repo. Created files main.cpp, NBody_functions.cpp/h, ODESolver.cpp/h. Started shell implementation of ODESolver, helper functions and a possible main-functions. Spent time contemplating some major design issues.

9.10.2014. Started coding. Discussed many design choices with the group teachers. Renamed ODESolver to NBodySolver and wrote the class Body. Wrote stub implementations of key methods. The flow of the program is unravelling as I work. Plan for the nearest future: get NBodySolver to work using Eulers method and a simple 2-body initial configuration.

13.10.2014. Wrote matlab script that generates initial condition files for the solar system. Wrote methods for reading initial conditions and initializing the Solver. Wrote the eulerAdvance()-method and implemented brute force gravitational calculator. Ended up with promising plots with matlab of the solar system (albeit quite inaccurate..). Problem: allow gravity() to live in seperate file.

14.10.2014 Added Pluto and Halley's comet. Wrote the method advanceRK4() with great success. Achieved stable trajectories for 11 bodies with T = 1000 weeks, dt = 0.05 weeks.

14.10.2014 Added Phobos & Deimos. Phobos requires at most hourly timestep! This calls for adaptivity! Added the Giovian planets. Initial configuration is now complete. initial_writer.m can write any selection of initial conditions for any part of the solar system to be sent to NBodySolver. Extremely satisfying results for the Giovian system. Stable trajectories.

15.10.2014 Regroup at the lab and start to think about the next steps. The next steps will be: 1) clean up & comment the code 2) Start writing individual timestep implementation in new branch 3) Test the project again. [After that: parallellization].

Finished cleaning up and commenting. Pushed the project. Read several papers on methods of N-Body simulations, and the physics behind. Need to visit the think-tank for some time now.

23.10.2014 Restructured the project this week. Extremely frustrating problem held me back some days. It is now taken care of. Wrote a first implementation of adaptive stepsize today at the lab. Will have to restructure the program again to allow Richardson method.

12 TODO

Write python script that generates the following initial conditions (and more!)

- Sun-earth-moon system
- Solar system (with/without moons)
- Spaceship launch from the earth
- Halleys comet enters orbit
- randomly placed inside a disk with 'correct' orbital velocity
- randomly placed (weighted in the center) inside a disk with 'correct' orbital velocity
- randomly placed inside a sphere with tangential velocity/no velocity

13 Lenker

http://en.wikipedia.org/wiki/Barnes%E2%80%93Hut_simulation
http://en.wikipedia.org/wiki/N-body_simulation
http://trekto.info/n-body-simulation
http://en.wikipedia.org/wiki/Plummer_model
http://burtleburtle.net/bob/math/multistep.html
http://www.artcompsci.org/ http://www.ifa.hawaii.edu/faculty/barnes/treecode/treeguide.html
https://www.ids.ias.edu/ piet/act/comp/algorithms/starter/