

Gravitational N-Body Simulations

Gudbrand Tandberg

14. oktober 2014

- 1 Abstract
- 2 Introduction to the N-body problem
- 3 The first NBodySolver class
- 4 Introduction to the numerical methods
 - 4.1 Euler's Method
 - 4.2 Runge Kutta 4
- 5 Results for the extended solar system
- 6 Extending NBodySolver to allow for different timesteps
- 7 Refined results for the extended solar system
- 8 Further development of the algorithm
 - 8.1 Parallellizing the code
 - 8.2 Faster gravity evaluations
 - 8.3 Close encounter regularization
- 9 Starclusters and beyond...
 - 9.1 The initial configurations
 - 9.2 Results
- 10 Graphics
- 11 Conclusion

12 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.

13 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