Format for delivery of report and programs

The format of the project is that of a printed file or hand-written report. The programs should also be included with the report. Write **only your candidate number** on the first page of the report and state clearly that this is your report for the final project of FYS3150/FYS4150, fall 2014. If you have collaborated with other students, please state also whom you have worked together with by stating the candidate number of your collaborator(s).

There will be a box marked 'FYS3150/FYS4150' at the reception of the Department of Physics (room FV128). If you opt for this project, the final report should also include project 3 on the solar system. Your final report counts 50% of the final grade.

Final part of astronomy project, N-body simulation of an open galactic cluster, deadline Monday December 1, 12pm (noon)

The goal in this project is to develop a code that can perform simulations of an open cluster using Newtonian gravity. First, however we will compare the stability of two different methods. This is because when we are looking at a system with a large number of particles, we are more interested in the statistical properties of the system than in the individual motion of each of the particles. This means that the stability of the solution method is more important than its short term accuracy. This project is inspired by the preprint by Joyce *et al.*, see Ref. [1] below. When solving this project, we recommend downloading this article. It is a good companion to understand the physics discussed here.

In the first part of this project we will explore the stability of two well-tested numerical methods for solving differential equations. Here you can use the algorithms and program you developed for the solar system in project 3. The algorithms to test and implement are the fourth-order Runge-Kutta method and the Velocity-Verlet method.

a) Implement the Newtonian two-body (you can choose masses and dimensionalities as you wish) problem in three dimensions using the fourth order Runge-Kutta method and the Velocity-Verlet method discussed in the lecture notes.

   You can build on the code you developed for project 3. Compare the stability of the two different methods. How do they work for large time steps? How do they work for very long times? Compare also the time used to advance one timestep for the two different methods. Comment your results. Which algorithm would you use for simulating systems that require long times?

b) Develop a method for adaptive time steps that can be used for a simulation with many particles (planets, stars, moons etc). A good idea is to let the different objects have different values for the time step, dt. The most straightforward way of doing this is to try to advance all particles using the largest time step, and if the time step is too large for some of the particles, you go back and advance these particles using half the time step. If this time step is still too large you can go on to a quarter of the original time step, and so on. (This method of halving the timesteps can be used to get a much greater accuracy using Richardson's extrapolation in the optional part of the project). (Hint: When calculating the forces on the particles with small time steps you should extrapolate the position of the particles with a larger time-step to the appropriate time.)

   How do you decide what is a good time step? Think about this!

   Two quick and dirty estimates that are used in some N-body simulations are

$$dt_i \sim \min_j \left(\frac{|\vec{r}_{ij}|}{|\vec{v}_{ij}|} \right),$$

   and

$$dt_i \sim \frac{1}{a_i},$$

where $|\vec{r}_{ij}|$ and $|\vec{v}_{ij}|$ are the distance and relative velocity between objects i and j, and a_i is the acceleration of object i. dt_i is then an estimate (up to some factor) of the time step needed for object i.

Are these good estimates? Why/why not? What is actually problematic with having too large time steps?

Other ways to estimate the time step needed are comparing different order schemes (like RK-4 and RK-5) or comparing results at different step sizes.

Is it also reasonable to let the largest step size vary in time? Why/why not?

A good way to test your adaptive time step scheme would be to impement it on a system containing the sun, the earth and the moon.

Is the adaptive scheme better (less time-consuming/more accurate) than the fixed time step scheme?

We will now try to build a simple model of an open cluster, see Ref. [2]. An open cluster is a group of up to a few thousand gravitationally bound stars created from the collapse of a molecular cloud. This collapse leads to a flurry of star formation. Open clusters are usually found in the arms of spiral galaxies, or in irregular galaxies. Since stars in an open cluster have roughly the same age, and are made from the same material, they are interesting in the study of stellar evolution, since many of the variable parameters we have when comparing two stars are kept constant.

Once open clusters are formed they gradually dissipate as members get ejected from the cluster due to random collisions, this means that open clusters generally last only a few hundred million years. In figure 1, we see the Hertzsprung-Russell diagrams for two open clusters.

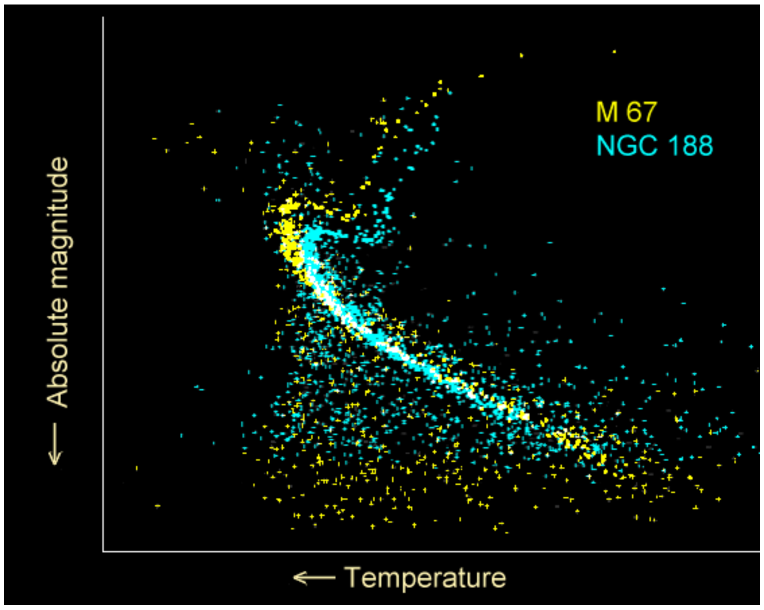

Figure 1: Hertzsprung-Russell diagrams for two open clusters, M67 and NGC 188. We see that most of the stars are on the main sequence. In the older cluster, NGC 188, we see that the heaviest stars are just now leaving the main sequence, while the younger cluster, M67, is following closely after.

We will look at a simple model for how an open cluster is made from the gravitational collapse and interaction among a large number of stars. We want to study this collapse, and the statistical properties of the collapsed system.

One particle in our model represents one or a few stars, and we will work with a few hundred particles. We will simulate what is called a "cold collapse", this means that we start the particles with little or no initial velocity.

c) Extend your code to an arbitrary number of particles, N, starting with a uniform (random) distribution within a sphere of a given radius R_0. Start the particles at rest, with masses randomly distributed by a Gaussian distribution around ten solar masses with a standard deviation of one solar mass. Use solar masses and light years as units of mass and length and make your equations dimensionless. The function *GaussPDF* included with this project can be used to generate random numbers which follow a Gaussian (or normal) distribution. The function for calculating these random numbers can be found at the webpage of the course together with the project files.

   How large time steps are required given $R_0 = 20ly$ (light years), and a $N = 100$? Do we have any units of time that fit this timescale? In the limit where $N \to \infty$, keeping ρ_0 constant, we get a continuous fluid. In this case the system collapses into a singularity at a finite time $\tau_{crunch} = \sqrt{\frac{3\pi}{32G\rho_0}}$. (For the especially interested (Not required!): Can you derive this result? Hint: recall the Friedman equations [3]).

   Why do we not observe this singularity in our model? Use τ_{crunch} as the unit of time, and find G in these units (G will become a function of the number of particles N, and the average mass of the particles, μ).

   You should run these calculations with both the fourth-order Runge-Kutta algorithm and the Velocity-Verlet method. Which method would you prefer? Give a critical discussion.

   For the remaining exercises, you should use only one of the above methods.

d) Run the system for a few τ_{crunch}. Save the positions of the particles at different times to file. Does the system reach an equilibrium? How long time does this take ?

e) Make a function that calculates the kinetic and potential energy of the system. Is the energy conserved? Some of the particles are ejected from the system, how can we identify these particles from the energies we have calculated? How much of the energy of the system is taken away by particle ejection? How does this change with different values of N? Are there still particles being ejected after the system reaches equilibrium?

f) We will now introduce a smoothing function to take care of the numerical instability that arises when two of the particles come very close. There are a lot of ways of inserting such a smoothing, but we will just look at a very simple one. We will modify the Newtonian force law to make it finite at short ranges

$$F_{mod} = -\frac{GM_1M_2}{r^2 + \epsilon^2}.$$

   The parameter ϵ is a "small" real constant. What should the value of this parameter be? Try out different values (which one gives you the best energy conservation?). Can we justify this correction to the pure Newtonian force by noting that our particles do not represent actual point particles but rather mass distributions of some finite extent? Does the addition of this correction change any of the results from part e) ?

g) Now we will look at the particles that are bound (not ejected). What is the distribution of potential and kinetic energy?

   The virial theorem says that for a bound gravitational system in equilibrium we have

$$2\langle K \rangle = -\langle V \rangle,$$

   where $\langle K \rangle$ is the average (over time) kinetic energy of the system and $\langle V \rangle$ is the average potential energy.

   By the ergodic hypothesis we can take an ensamble average (average over a large system) instead of the time average.

   Are your results consistent with the virial theorem?

h) Your task here is to parallelize using either MPI or OpenMP the differential solver you have developed for this project. What kind of speed-up do you get, that is, how does the speed-up of your code scale with the number of processes? Make sure that your previous results are reproduced.

i) This part is optional but gives you an additional 30% on the final score! Try to plot the radial density of the particles (the particle density as a function of radius) in the equilibrium state. How would you extract such an information from your calculations? (Hint: make a histogram for the radial particle density) What is the average distance? What is the standard deviation? Plot the radial distribution of particles.

   Run the code for different number of initial particles, keeping the total mass constant. What is the average distance as a function of N?

   The radial distribution of particles in this kind of cold collapse can often be fit very well with the simple expression

$$n(r) = \frac{n_0}{\left(1 + \left(\frac{r}{r_0}\right)^4\right)}.$$

   Try to fit your data to this curve, what is the value n_0 and r_0? Can you find how these values depend on N?

   How many particles can you simulate?

   Compare your results with those found in Ref. [1].

   If you want, you can also compare your results to the well-known Navarro-Frenk-White profile

$$\rho(r) = \frac{\rho_0}{\frac{r}{r_0}\left(1 + \left(\frac{r}{r_0}\right)^2\right)}.$$

   Does this fit better?

References

[1] M. Joyce, B. Marcos, and F. Sylos Labini, Cold uniform spherical collapse revisited, arXiv1011.0614 (2011), http://arxiv.org/abs/1011.0614.

[2] P. J. E. Peebles, *The Large-Scale Structure of the Universe*, Princeton University Press, 1980. See also C. Payne-Gaposchkin, *Stars and clusters*, (Cambridge, Harvard University Press, 1979).

[3] A. Friedman, *On the Curvature of Space*, General Relativity and Gravitation **31**, 1991 (1999).