FYS4150 Project 5: N-body simulation of an open galactic cluster

Marie Foss (# 56), Maria Hammerstrøm (# 59)

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Github: *https://github.com/mariahammerstrom/Project5*

1 Introduction

An open cluster is a group of up to a few thousand stars that are gravitationally bound to each other. These stars are created from the same giant molecular cloud,

These stars have roughly the same age and are created from the same giant molecular cloud. Open clusters are interesting subjects in the study of stellar evolution, because of the member's similar age and composition, making their properties such as distance, age, metallicity and extinction more easy to determine.

Once an open cluster has been formed, it will gradually dissipate as its members get ejected from the cluster due to random collisions, lasting only a few hundred million years.

Open clusters are usually found in the arms of spiral galaxies or in irregular galaxies.

(Images which shows an open cluster, and a galaxy with visible clusters - HII regions.)

In this project we looked at a simple model for how an open cluster is made from a gravitational collapse and for the interactions among a large number of stars. We have simulated a "cold collapse", meaning the particles will have little or no initial velocity.

...

First we study the Newtonian two-body problem in three dimensions, then extend this to an N-body problem in three dimensions. The time evolution of the system is described through **Newton's law**, given by:

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M} \tag{1}$$

where $F_{G,x}$ is the **gravitational force** in the x direction, given by

$$F_{G,x} = -\frac{GM_1M_2}{r^3}x, \tag{2}$$

where M_1 and M_2 are the masses of the two objects and r is the distance between them. In three dimensions, $r = \sqrt{x^2 + y^2 + z^2}$. We have similar expressions for the y and z direction.

Eq. (1) is a second-order differential equation, which we can rewrite as a set of coupled first-order differential equations:

$$\frac{dx}{dt} = v_x \quad \text{and} \quad \frac{dv_x}{dt} = F_{G,x}. \tag{3}$$

These equations can be used to determine the position and velocity of an object.

When solving this problem we will convert the units of mass and length to dimensionless variables using solar masses $M_\odot$ and light years.

... (describe this and introduce new equations)

When setting up the N-body problem we will let the particle start at rest with masses randomly distributed by a Gaussian distribution around $10M_\odot$ with a standard deviation of $1M_\odot$. The particles will start with uniformly randomly distributed positions within a sphere of a given radius R_0.

2 Methods

The numerical algorithms we wish to implement and compare are the fourth-order Runge-Kutta method and the Velocity-Verlet method. Both methods assume a known initial value for the position (x_i, y_i, z_i) and the velocity $(v_{i,x}, v_{i,y}, v_{i,z})$.

2.1 4th order Runge-Kutta method

Using the 4th order Runge-Kutta method (RK4), we can estimate the next value x_{i+1} (and similarly y_{i+1} and z_{i+1}) through the formula

$$x_{i+1} \approx x_i + \frac{h}{6} + 2v_{i+1/2,x} + 2v_{i+1/2,x} + v_{i+1,x} \tag{4}$$

where $v_{i+1/2,x}$ is the velocity in the x direction at time $t_{i+1/2}$ and position $x_{i+1/2}$ after taking a step $h/2$, which marks the midpoint between $v_{i,x}$ and $v_{i+1,x}$. The midpoint is evaluated two times to get a better estimate of the function value at this point, which in turn will give a better estimate for the final point $v_{i+1,x}$. h denotes the time step, defined as $h = (t_f + t_0)/N$ where t_f is the end-time of our simulation, t_0 is the start-time (usually chosen as zero), and N is the number of integration points (not to be confused with the number of particles in the simulation!).

The algorithm goes as follows:

- Compute $K_1 = h\, v_x(t_i, x_i)$, which gives the slope at t_i.
- Compute the slope at the midpoint using Euler's method, which gives $K_2 = h\, v_x(t_i + h/2, x_i + K_1/2)$.
- The estimate of the slope at the midpoint is then further improved by $K_3 = h\, v_x(t_i + h/2, x_i + K_2/2)$.
- Finally compute the end-point through: $x_{i+1} = x_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K4)$.

These equations must be deployed for all three dimensions x, y and z.

Using this method gives a global truncating error which goes like $O(h^4)$.

2.2 Velocity-Verlet method

This method is derived by performing a Taylor expansion of $x(t + h)$ and $x(t - h)$, and add the two equations. This results in the formula

$$x_{i+1} = 2x_i - x_{i-1} + a_x(t_i, x_i)h^2 \quad i = 1, 2, \dots \tag{5}$$

where a_x is the acceleration, which is the second derivative of x. Velocity is not included in the equation, but can be computed using the well-known formula

$$v_{i,x} = \frac{x_{i+1} - x_{i-1}}{2h} + O(h^2) \tag{6}$$

where x_{i+1} has been estimated by Eq. (5).

We see from both Eq. (5) and Eq. (6) that this algorithm is not self-starting as it depends on x_{i-1}. The first time we calculate Eq. (5), meaning x_2, we need to know both x_0 and x_1. The known initial position gives us x_0 and we have to use a different algorithm to make an estimate for x_1. The simplest solution is to use Euler's method, which approximates x_1 by:

$$x_1 = x_0 + v_{0,x}h^2 \tag{7}$$

3 Results

3.1 Stability of methods

...

4 Conclusions

...

5 List of codes

The codes developed and used for this project are:

...