UNIVERSITY OF OSLO

COMPUTATIONAL PHYSICS

Project 5

UiO : University of Oslo

Authors:

Birgitte Madsen, 66

Magnus Isaksen, 14

Soumya Chalakkal, 51

Autumn 2015

UiO : University of Oslo

Department of Physics
University of Oslo
Sem Sælands vei 24
0371 Oslo, Norway
+47 22 85 64 28
http://www.mn.uio.no/fysikk/english/

Course:

Computational Physics

Project number:

5

Link to GitHub folder:

`https://?????`

Hand-in deadline:

Friday, December 11, 2015

Project Members:

Birgitte Madsen, 66
Magnus Isaksen, 14
Soumya Chalakkal, 51

Copies: 1
Page count: ??
Appendices: 0
Completed: ????, 2015

ABSTRACT

TABLE OF CONTENTS

Chapter 1 Introduction **1**

Chapter 2 Method **3**
2.1 Transformation between units 3
2.2 Newtonian two-body problem and greater systems in three dimension 4
2.2.1 Velocity-Verlet method 6
2.2.2 Fourth Order Runge-Kutta Method 7
2.2.3 A smoothing function 8
2.3 Generating Mass and Position for Cluster Particles 9
2.3.1 Gaussian Distributed Mass 9
2.3.2 Uniformly Distributed Position 11
2.4 Computing the Energy 12

Chapter 3 Results and Discussion **15**
3.1 Stability of the 2-body System using Runge-Kutta and Velocity-Verlet 15
3.2 Testing Runge-Kutta and Velocity-Verlet for Sun-Earth-Mars System 18
3.3 Time Step Length in the N-body Problem 19
3.4 Stability and Equilibrium of $N = 100$ System 22
3.5 Energy of the N-body Runge-Kutta Method 23
3.6 Distribution of Potential and Kinetic Energy of Bound Particles after a Finite Time 24

Chapter 4 Conclusion **27**

Bibliography **29**

CHAPTER

1 INTRODUCTION

CHAPTER

2 METHOD

The methods introduced in this chapter, are used to study the evolution of bodies, also referred to as particles, in the solar system or in the galaxy. Firstly, the fourth order Runge-Kutta method and the Velocity-Verlet method are introduced for a 2-body problem in 3 dimensions, however, since the aim of this project is to investigate the time evolution of a star cluster consisting of N particles, the code is extended to N bodies by incorporating suitable changes.

To generate the N bodies, a function for generating the position coordinates of the N bodies as uniformly distributed particles within a sphere is introduced as discussed in sec Sec. 2.3, along with a function for generating masses that are randomly distributed by a Gaussian distribution around ten solar masses with a standard deviation of one solar mass.

In the Github folder `https://github.com/birgimad/Project5_BM_MI_SC`, the source codes for the algorithms described in this chapter can be found.

2.1 Transformation between units

When considering at a planetary system or a larger system like a galaxy it is inconvenient to use the SI units for length and time. Instead, to investigate the evolution of astronomical systems, it is an advantage to use days, years (yr) or even longer time periods as the unit of time, and astronomical units (AU) or light-years (ly) as the unit of distance. The change in unit system, evidently changes the considered constant, the gravitational constant G, which in SI units is given as

$$G = 6.67 \cdot 10^{-11} \frac{\mathrm{Nm}^2}{\mathrm{kg}^2}$$

For a planetary system like the Earth-Sun system it is better to consider distances in AU instead of meters, and use days as a measure of time, as the planet doesn't move far on its orbit in a second. Furthermore it's an advantage to express the masses in the system in units of solar mass. Hence the constants have to be transformed into these unit systems.

The gravitational constant G is transformed using

$$1\ \mathrm{AU} = 1.495 \cdot 10^{11}\ \mathrm{m} \qquad \text{and} \qquad 1\ \mathrm{M}_\odot = 1.989 \cdot 10^{30}\ \mathrm{kg}$$

giving the gravitational constant for the planetary system

$$G = 2.96 \cdot 10^{-4} \frac{\mathrm{AU}^3}{\mathrm{days}^2 \mathrm{M}_\odot}$$

which is convenient when considering a planetary system.

For a star cluster the distances are greater and the time scales are larger than for the planetary system. Therefore it's more convenient to use years as the unit of time and light-years (ly) as the unit of distance.

$$1 \text{ yr} = 3.1536 \cdot 10^7 \text{s} \qquad \text{and} \qquad c = 2.008 \cdot 10^8 \frac{\text{m}}{\text{s}}$$

in which c is the speed of light. This yields that 1 ly is

$$1 \text{ ly} = 9.45 \cdot 10^{15} \text{ m}$$

Giving the gravitational constant for stellar distances

$$G = 1.536 \cdot 10^{-13} \frac{\mathrm{ly}^3}{\mathrm{yr}^2 \mathrm{M}_\odot}$$

2.2 Newtonian two-body problem and greater systems in three dimension

The problem of solving the time-evolution of a two-body system in three dimensions can reasonably be considered in two different coordinate systems: one coordinate system with one of the bodies in rest compared to the frame of reference in which the other body is moving, and one coordinate system with both of the bodies moving relative to the frame of reference. Both of these reference systems are depicted in Fig. 2.1.

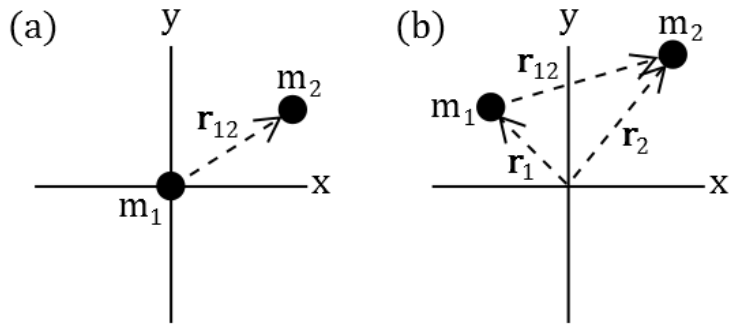

Figure 2.1: Two-dimensional illustration of the three-dimensional problem of determining determining the relative distance and relative velocity between two bodies. In (a) body 1 with mass m_1 is considered stationary in position- and velocity-space, whilst body 2 with mass m_2 moves relative to body 1. In (b) both body 1 and 2 moves relative to the frame of reference in position and time, yielding that the position vector between body 1 and 2 is given as $\mathbf{r}_{12} = \mathbf{r}_2 - \mathbf{r}_1$.

In the codes presented in this section, solving the problem in coordinate system (a) will first be considered for simplicity. Thereafter, the codes will be extended to include the movement of body 1 relative to the coordinate system, since this will be useful when extending the codes to an N body system.

In the problem, $\mathbf{r}(t)$ is the three-dimensional space vector consisting of the coordinated $(x(t), y(t), z(t))$, whilst $\mathbf{v}(t)$ is the three-dimensional velocity vector with coordinates $(v_x(t), v_y(t), v_z(t))$, both of which are dependent on time.

In general, the considered differential equation is

$$\frac{dy}{dt} = f(t,y) \tag{2.1}$$

Which yields that

$$y(t) = \int f(t,y)dt \tag{2.2}$$

For the two bodies in a three dimensional Newtonian gravitational field this corresponds to six coupled differential equations given by the vector equations

$$\frac{d\mathbf{r}}{dt} = \mathbf{v} \qquad \text{and} \qquad \frac{d\mathbf{v}}{dt} = -\frac{GM_2}{r^3}\mathbf{r} \tag{2.3}$$

in which M_2 is the masses of the the other bodies, whilst r is the distance between the bodies. The equations in (2.3) are computed by the script given below in which *drdt* corresponds to the derivative of the coordinates of the position, and *dvdt* corresponds to the derivative of the velocity coordinates.

```
void Derivative(double r[3], double v[3], double (&drdt)[3], double (&dvdt)[3], double
    G, double mass){
    drdt[0] = v[0];
    drdt[1] = v[1];
    drdt[2] = v[2];
    double distance_squared = r[0]*r[0] + r[1]*r[1] + r[2]*r[2];
    double newtonian_force = -G*mass/pow(distance_squared,1.5);
    dvdt[0] = newtonian_force*r[0];
    dvdt[1] = newtonian_force*r[1];
    dvdt[2] = newtonian_force*r[2];
}
```

When including movement of both bodies relative to the frame of reference, the *Derivative* function must be slightly modified, since then the relative position of the two bodies will be given as $\mathbf{r}_{12} = \mathbf{r}_2 - \mathbf{r}_1$. For a general case with N particles, the *distance_squared* between body i and j, and the acceleration in x, y and z due to the Newtonian force between body i and j can be determined by the following lines of code.

```
for (int j=0; j<number_of_particles; j++)
    {
        if (j!=i)
        {
            distance_squared = 0;
            for (int k=0; k<3; k++)
            {
                distance_squared += (r(i,k)-r(j,k))*(r(i,k)-r(j,k));
            }
            force_between_particles = m(j) * pow(distance_squared,-1.5);
            acc_x += G*force_between_particles*(r(j,0)-r(i,0));
```

```
            acc_y += G*force_between_particles*(r(j,1)-r(i,1));
            acc_z += G*force_between_particles*(r(j,2)-r(i,2));
        }
    }
```

The if statement in the for loop over all bodies adds up the acceleration in all three dimensions of all particles due to the presence of other particles. Hence, for the two body problem, the argument in the if statement will only be true once for each of the two particles.

2.2.1 Velocity-Verlet method

The basic idea of the Velocity-Verlet algorithm is to write the Taylor expansion of the position in Newtons equation, one forward step and one backward step in time with step length δt as

$$\mathbf{r}(t_i \pm \delta t) = \mathbf{r}(t_i) \pm \mathbf{v}(t_i)\delta t + \mathbf{a}(t_i)\frac{\delta t^2}{2} \pm \frac{\delta t^3}{6}\frac{d^3\mathbf{r}(t_i)}{dt^3} + \mathscr{O}(\delta t^4) \tag{2.4}$$

in which $\mathbf{v}(t_i) = d\mathbf{r}(t_i)/dt$ is the velocity, and $\mathbf{a}(t_i) = d^2\mathbf{r}(t_i)/dt^2$ is the acceleration at time t_i. [1, p. 248] Adding the two expressions in Eq. (2.4) gives

$$\mathbf{r}(t_i + \delta t) = 2\mathbf{r}(t_i) - \mathbf{r}(t_i - \delta t) + \mathbf{a}(t_i)\delta t^2 + \mathscr{O}(\delta t^4) \tag{2.5}$$

which has a truncation error that goes as $\mathscr{O}(\delta t^4)$. Now, using $\mathbf{r}(t_i - \delta t) = \mathbf{r}(t_i) - \mathbf{v}(t_i)\delta t + \mathbf{a}\delta t^2/2 + \mathscr{O}(\delta t^3)$, yield that the position at time $t_i + \delta t$ can be determined as

$$\mathbf{r}(t_i + \delta t) = \mathbf{r}(t_i) + \mathbf{v}(t_i)\delta t + \frac{1}{2}\mathbf{a}(t_i)\delta t^2 \tag{2.6}$$

Since the velocity is not included in Eq. (2.5), it is computed through the Velocity-Verlet scheme where position, velocity and acceleration at time $t_i + \delta t$ is computed from the Taylor expansion as

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \frac{1}{2}(\mathbf{a}(t) + \mathbf{a}(t + \delta t))\delta t \tag{2.7}$$

The velocity at time $t_i + \delta t$ is in the algorithm computed by first calculating

$$\mathbf{v}_{part1}(t + \delta t) = \mathbf{v}(t) + \frac{1}{2}\mathbf{a}(t)\delta t \tag{2.8}$$

and then use the *Derivative* function to determine $\mathbf{a}(t + \delta t)$, which is then used to compute the remaining term of Eq. (2.7) as

$$\mathbf{v}_{part2}(t + \delta t) = \frac{1}{2}\mathbf{a}(t + \delta t)\delta t \tag{2.9}$$

The velocity-Verlet method uses the algortihm *Derivative* described in Sec. 2.2, to generate the six differential equations, in the following while-loop that runs until reaching the final time in time steps of length $\delta t = (t_{initial} - t_{final})/(\#timesteps)$.

```
while(time<=t_final){
Derivative(r,v,drdt,dvdt,G,mass);
for(int i=0; i<6 ; i++){
```

```
    r[i] = r[i]+dt*drdt[i] + 0.5 * dt * dt * dvdt[i];
    v_partly[i] = drdt[i] + 0.5 * dt * dvdt[i];
    dvdt[i] = v_partly[i];
    }
    Derivative(r,v,drdt,dvdt,G,mass);
    for(int i=0; i<n ; i++){
    v[i] = v_partly[i] + 0.5 * dt * dvdt[i];
    }
    time += dt;
    }
```

2.2.2 Fourth Order Runge-Kutta Method

The Runge-Kutta method is based on Taylor expansions, with the next function value after a times step $\delta t = t_i - t_{i+1}$ being computed from four more or less improved slopes of the function in the points t_i, $t_i + \delta t/2$ and t_{i+1}, and has a truncation error that goes as $\mathscr{O}(\delta t^4)$. [1, p. 252]

The first step of the RK4 method is to compute the slope k_1 of the function in t_i by

$$k_1 = \delta t f(t_i, y_i)$$

Then the slope k_1 at the midpoint is computed from k_1 as

$$k_2 = \delta t f(t_i + \delta t/2, y_i + k_1/2)$$

The slope at the midpoint is then improved from k_2 by

$$k_3 = \delta t f(t_i + \delta t/2, y_i + k_2/2)$$

from which the slope k_4 at the next step y_{i+1} is predicted to be

$$k_4 = \delta t f(t_i + \delta t, y_i + k_3)$$

From the computed slopes k_1, k_2, k_3 and k_4, the function value at $t_i + \delta t$ is computed as

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.10}$$

When implementing this for the two-body problem in three dimensions, it boils down to a continuous call of two functions, namely the function *Derivative* given in Sec. 2.2 and the function *updating_dummies* given below.

```
void updating_dummies(double dt, double drdt[3], double dvdt[3], double (&r_dummy)[3],
    double (&v_dummy)[3], double number, double (&kr)[3], double (&kv)[3], double
    r[3], double v[3])
{
    for (int i = 0; i<3; i++){
        kr[i] = dt * drdt[i];
        kv[i] = dt * dvdt[i];
        r_dummy[i] = r[i] + kr[i]/number;
        v_dummy[i] = v[i] + kv[i]/number;
    }
}
```

The function *updating_dummies* computes the values of k_1, k_2, k_3 and k_4 for all three space coordinates and velocity coordinates from the derivatives *drdt* and *dvdt* computed by the *Derivative* function. To compute the next step given by Eq. (2.10), the following succession of function calls are made until the time reaches the final time t_{final} after $(t_{final} - t_{inital})/\delta t$ time steps.

```
while(time<=t_final){
    Derivative(r,v,drdt,dvdt,G,mass);
    updating_dummies(dt,drdt,dvdt,r_dummy,v_dummy,2,k1r,k1v,r,v);
    Derivative(r_dummy,v_dummy,drdt,dvdt,G,mass);
    updating_dummies(dt,drdt,dvdt,r_dummy,v_dummy,2,k2r,k2v,r,v);
    Derivative(r_dummy,v_dummy,drdt,dvdt,G,mass);
    updating_dummies(dt,drdt,dvdt,r_dummy,v_dummy,1,k3r,k3v,r,v);
    Derivative(r_dummy,v_dummy,drdt,dvdt,G,mass);
    for (int i = 0; i<n; i++){
        k4r[i] = dt*drdt[i];
        k4v[i] = dt*dvdt[i];
    }
    for (int i=0; i<n;i++){
        r[i] = r[i] +(1.0/6.0)*(k1r[i]+2*k2r[i]+2*k3r[i]+k4r[i]);
        v[i] = v[i] +(1.0/6.0)*(k1v[i]+2*k2v[i]+2*k3v[i]+k4v[i]);
    }
    time += dt;
    }
```

When including the movement of both bodies relative to the reference system or adding more bodies to the system, **r**'s, **v**'s, **k**'s etc. must be generated for all of the particles, yielding introduction of a for loop over all particles.

2.2.3 A smoothing function

Simulating two stars moving around each other in space, the method used is looking at them as point particles. This method is good to some extent, but it allows the stars to pass inside each other without this being a hindrance. In order to simulate that the stars are not in fact point particles, but have a radius R, as illustrated in Fig. 2.2.

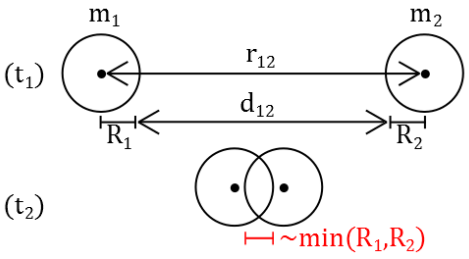

Figure 2.2: Stars in motion

To avoid this problem it is possible to introduce a small constant ε that acts as a smoothing function, giving a minimum distance between the stars. This prevents the stars from moving in to close to each other. Meaning that they now acts not like point particles but as objects with an extent in space. This epsilon changes Eq. (2.3) into

$$\frac{\partial v}{\partial t} = -\frac{GM_2}{r^2 + \varepsilon^2} \tag{2.11}$$

2.3 Generating Mass and Position for Cluster Particles

When considering the time evolution of an N-particle star cluster, in which the mass and position of each single particle is not known from a catalogue or of special significance, it is inappropriate to hard code the values for each single particle. Instead, it is an advantage to use probability distribution functions to generate random positions and masses for each of the particles. It is decided that the mass of the N particles in the cluster should follow a Gaussian distribution around ten solar masses with a standard deviation of one solar mass. For the position, it is decided that the density of particle initially is uniform in a sphere with a radius of twenty solar masses. The is, however, not equivalent to saying that the particles are uniformly distributed in the Cartesian coordinates x, y and z and neither in the spherical coordinates r, ϕ and θ, which complicates the problem a bit. The sections below present the functions for generating random masses an positions of N particles with the desired distributions. Each of the functions are tested for a system with $N = 100,000$.

2.3.1 Gaussian Distributed Mass

Pseudo-random numbers, corresponding to masses, randomly distributed by a Gaussian distribution is generated following the Box-Muller transform. The basic form of the Box-Muller transform gives

$$X_1 = \sqrt{-2ln(V_1)}sin(2V_2) = Rcos(\theta)$$
$$X_2 = \sqrt{-2ln(V_1)}cos(2V_2) = Rsin(\theta)$$

where $R^2 = -2ln(V_1)$ and $\theta = 2V_2$ [2]. X_1 and X_2 are random numbers distributed according to a Gaussian distribution of mean 0 and variance 1.

In polar form the above two equations becomes

$$X_1 = u\sqrt{\frac{-2ln(s)}{s}}$$
$$X_2 = v\sqrt{\frac{-2ln(s)}{s}}$$

where $s = R^2 = u^2 + v^2$. Here u and v are uniformly distributed in the interval [-1,1] and points only within the unit circle is admitted (see Fig. 2.3). Therefore, only those pairs of u and v which gives a value for s in the interval (0,1) is considered. Value of s is similar to that of V_1 and $\theta/2\pi$ is similar to that of V_2 in the basic form. The lines of code below shows the implementation of the generation of masses normally distributed with a mean of $10M_\odot$ and a standard deviation of $1M_\odot$.

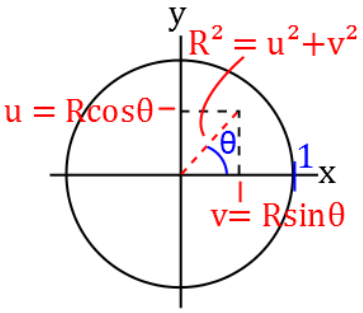

Figure 2.3: The values of u and v will always be in the interval $[-1;1]$.

```
void gaussian_mass_generator(vec
(&mass), int number_of_particles)
{
  srand(time(NULL));
  for (int i = 0; i < number_of_particles; i++)
  {
  static int iset = 0;
```

```
  static double gset;
  double fac, rsq, v1, v2;
    do{
   //generate two random numbers uniformly distributed in
       the interval [-1,1]
      v1 = 2.*((double) rand() / (RAND_MAX)) -1.0;
      v2 = 2.*((double) rand() / (RAND_MAX)) -1.0;
      //Radius of the numbers (within the unit circle)
          squared
      rsq = v1*v1+v2*v2;
    } while (rsq >= 1.0 || rsq == 0.);
    //computing the gaussian distributed numbers
    fac = sqrt(-2.*log(rsq)/rsq);
    gset = v1*fac;
    iset = 1;
    mass(i) = v2*fac;
    mass(i) += 10;
  }
}
```

In the function *gaussian_ mass_ generator*, v1 and v2 are random numbers uniformly distributed in the interval [-1,1]. $rsq = v1^2 + v2^2$ corresponds to $s = R^2 = u^2 + v^2$. Using the do-while loop only those pairs of v1 and v2 that produces an s equal to 0 or greater than or equal to 1 is generated so that the points are inside unit circle. Variable gset correspond to X1.

To test whether the generated masses are actually normally distributed around $10M_\odot$ with a standard deviation of $1M_\odot$, 100,000 masses are generated, by the presented code, and plotted in a histogram below.

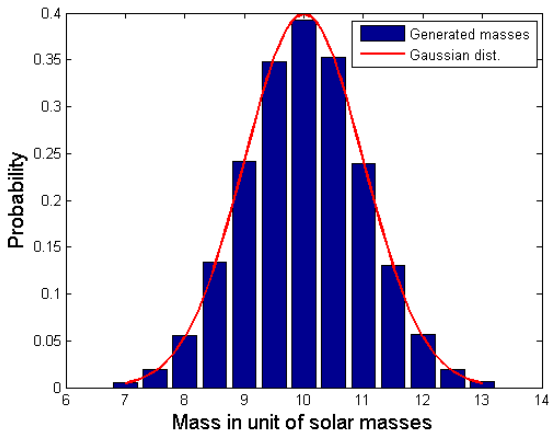

Figure 2.4: Histogram of the mass of 100,000 particles generated by the c++ code introduced above together with a Gaussian distribution of mean $10R_\odot$ and standard deviation $1R_\odot$ generated in MatLab using the *normpdf* function.

2.3.2 Uniformly Distributed Position

To generate positions uniformly distributed inside a sphere of radius R_0, random numbers generated using the *rand* function in c++ are used and converted into coordinates of uniformly distributed particles within the sphere. In order to get a uniform density of particles three variables v, w and u corresponding to random numbers uniformly distributed between 0 and 1 are introduced. The spherical coordinates θ, ϕ and r are, according to [3], then linked to these variables using the equation

$$\theta = cos^{-1}(1-2v)$$
$$\phi = 2\pi w$$
$$r = R_0(u)^{1/3}$$

The following equations are then used to get back to the Cartesian coordinate system.

$$x = rsin(\theta)cos(\phi)$$
$$y = rsin(\theta)sin(\phi)$$
$$z = rcos(\theta)$$

After performing these steps, a uniform distribution of N particles within a sphere of radius R_0 is achieved. Below, the code for generating this uniform distribution within a sphere of radius $R_0 = 20$ly is introduced.

```
void uniform_pos_generator(mat (&position), int N)
{
double pi=3.14159, c = 2*pi, R = 20;
vec phi(N), r(N), theta(N), x(N), y(N), v(N);

srand(time(NULL));

for (int i=0;i<N;i++){

        x(i) = ((double) rand() / (RAND_MAX)); //random numbers generated in the
            interval(0,1)
        y(i) = ((double) rand() / (RAND_MAX));
        v(i) = ((double) rand() / (RAND_MAX));
   }
for (int i=0;i<N;i++){
        phi(i)=c*x(i);
        r(i)=R*pow(y(i),1.0/3.0);
        theta(i)=acos(1.0-2.0*v(i));
        position(i,0)=r(i)*sin(theta(i))*cos(phi(i));
        position(i,1)=r(i)*sin(theta(i))*sin(phi(i));
        position(i,2)= r(i)*cos(theta(i));
   }
}
```

To test whether the generated positions within the sphere of radius 20 ly, the density of particles in the cross-sectional area of each x-value is determined and plotted as a histogram in Fig. 2.6 for $100,000$ particles with position generated by the introduced lines of code. The density of particles in the cross-sectional area of each x-value is found by dividing the total number of particles with that x-value with

the cross-sectional area of the sphere in that x-value (see Fig. 2.5). The cross-sectional area of the sphere in a specific area is found from a little trigonometry, by first considering that the radius of the circle that makes of the cross-sectional area in a point x_i is given by $r_i = 20sin\theta_i$ ly. This yields that the area A_i of the cross-sectional area, in ly, is given as

$$A_i = 400\pi sin^2\theta_i = 400\pi(1 - cos^2\theta)$$

in which the last equal sign stems from $1 = cos^2\theta + sin^2\theta$. But $x_i = 20cos\theta_i$ ly, giving

$$A_i = \pi(400 - x_i^2)$$

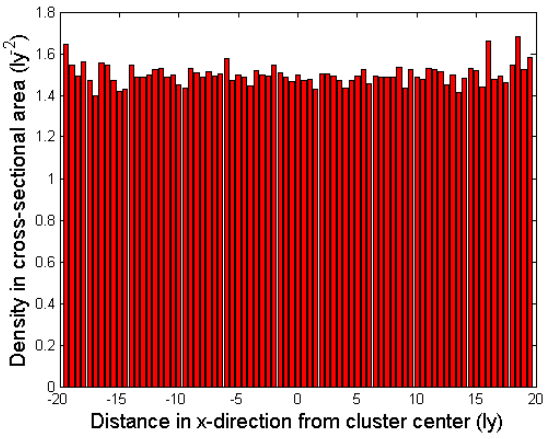

Figure 2.6: Histogram of density of 100,000 particles with position generated by the code introduced in [1] as a function of the x-coordinate of the particles. The histogram is made with bins in the interval $[-19.5; 19.5]$ and a bin-size of 0.5. The distance $x = \pm 20$ from the cluster center is not considered, since the cross-sectional area in that point is zero.

2.4 Computing the Energy

In order to test whether the energy is conserved, the initial energy of the system can be calculated and printed together with the final energy after a specific time interval. According to the conservation of energy, these are equal.

The total energy E_{tot} of the system is found by summing up the potential energy E_{pot} and kinetic energy E_{kin} of the N bodies that constitutes the system. The total potential energy is calculated as

$$E_{pot} = \sum_{1=0}^{N} \sum_{j \neq i} \frac{Gm_i m_j}{r_{ij}} \tag{2.12}$$

in which m_i and m_j are the masses of the i'th and j'th body, respectively, $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ is the distance between the two bodies, and G is the gravitational constant. The total kinetic energy of the system is calculated as

$$E_{kin} = \frac{1}{2} \sum_{1=0}^{N} m_i v_i^2 \tag{2.13}$$

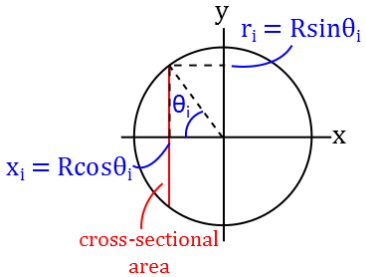

Figure 2.5: Two-dimensional illustration of the three-dimensional problem of determining the density of particles in each x-value.

with v_i being the speed of the i'th particle calculated as $v_i = \sqrt{v_{ix}^2 + v_{iy}^2 + v_{iz}^2}$, and m_i is the corresponding mass of that body.

The c++ code for computing the total energy of the system is given here below. When the kinetic energy is calculated, v_i is not explicitly calculated. Instead v_i^2 is calculated to reduce the number of floating point operations.

```
for (int i=0; i<number_of_particles; i++){
   for (int k=0; k<3; k++){
     kin_en(i) += v(i,k)*v(i,k);
    }
    kin_en(i) = 0.5*m(i)*kin_en(i);
    for (int j=0; j<number_of_particles; j++){
        if (j != i){
         pot_en(i) += pow(distance_between_particles(i,j),-1.0)*m(j);
        }
    }
    pot_en(i) = pot_en(i)*G*m(i);
    tot_en(i) = kin_en(i)+pot_en(i);
}
```

CHAPTER

3 RESULTS AND DISCUSSION

The starting point is solving the two body system, where Earth- and Sun-like forms the two bodies with masses $1M_\odot$ and $3 \cdot 10^{-6}M_\odot$, respectively, for which $M_\odot$ is the solar mass. With the help of the Runge-Kutta method and the Velocity-Verlet method introduced in Sec. 2.2, the problem is solved both with a stationary Sun-like body relative to the frame of reference, and with both Earth and Sun moving relative to the coordinate system, with an initial velocity (0,0,0) and initial position an (1,1,1) for the Sun. For earth the initial position is assigned to be (2,1,1) and initial velocity (0,0.017,0).

For an N body system, the movement of the bodies with the evolution of time is estimated with the fourth order Runge-Kutta method. From the results and analysis the behaviour of the system is unfold.

The results from running the codes described in Chap. 2 for simulating a two-body problem and a star cluster consisting of N particles interacting with the Newtonian force can be found in the GitHub folder `https://github.com/birgimad/Project5_BM_MI_SC`, together with data for various runs of the algorithms and functions.

3.1 Stability of the 2-body System using Runge-Kutta and Velocity-Verlet

The considered two-body system consists of an Earth-like body that orbits a Sun-like body at an initial distance of 1 AU. The initial velocity is chosen so that the orbital period of the Earth-like body is 365 days, and due to the choice of the initial position of the Earth-like body relative to the Sun-like body and the initial direction of the velocity of the Earth-like body, the orbit of the Earth-like body is purely in the $x-y$-plane. Two cases are considered: the situation when the Sun is at rest compared to the frame of reference at all times, and the situation with both the Sun and the Earth moving relative to the frame of reference. The chosen position and velocity vectors for the Earth and Sun in these two situations are shown in Tab. 3.3.

Table 3.1: Initial position and velocity for Earth and Sun in the Sun-Earth-like two-body system. F1 refers to the frame of reference at which the Sun is at rest in origo at all times, whilst F2 refers to the frame of reference in which both the Sun and the Earth moves relative to the coordinate axis. The mass of the Earth is given in solar masses, that is $M_E = 3.0 \times 10^{-6} M_\odot$, and the gravitational constant is given is $2.96 \cdot 10^{-4} \frac{AU^2}{days^2 M_\odot}$ (see Sec. 2.1).

	$\mathbf{r}_{initial}$ [AU]	$\mathbf{v}_{initial}$ [AU/day]
Earth (F1)	$(1.0, 0.0, 0.0)$	$(0.0, 0.017, 0.0)$
Sun (F2)	$(1.0, 1.0, 1.0)$	$(0.0, 0.0, 0.0)$
Earth (F2)	$(2.0, 1.0, 1.0)$	$(0.0, 0.017, 0.0)$

Fig. 3.3 shows the final distance between the two bodies as a function of time step length for the two situations: one in which sun is stationary relative to frame of reference and another in which sun is moving relative to the frame of reference. In both cases the final distance after 100 years is calculated between the Sun- and the Earth-like bodies and plotted as a function of length of time steps using both the fourth order Runga Kutta method and the Velocity-Verlet method.

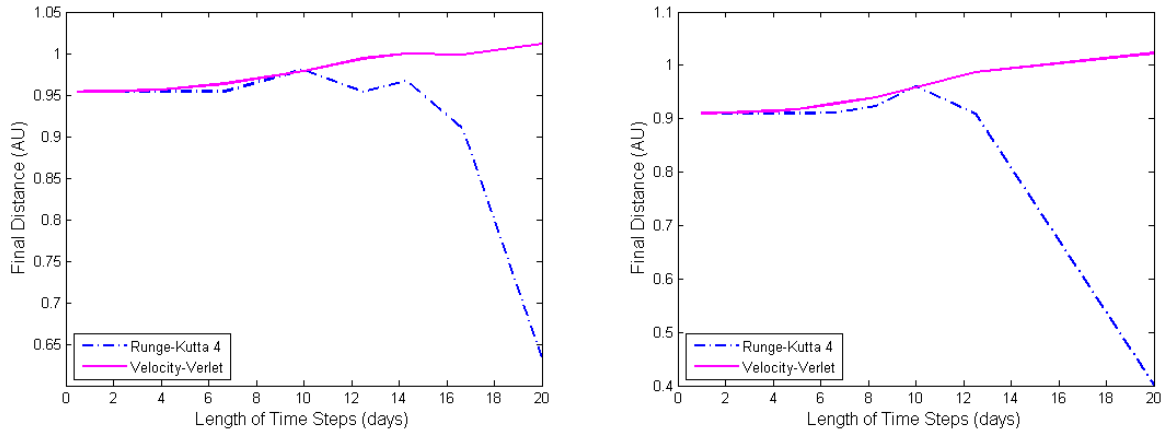

Figure 3.1: Distance between bodies after 100 years as a function of time step length for the Earth-Sun-like two-body system using both the forth order Runge-Kutta method and the Velocity-Verlet method. The leftmost plot do not allow for Sun motion relative to the frame of reference, whilst the rightmost allows for movement of both the Earth and the Sun relative to the frame of reference.

The result using the Velocity-Verlet method shows a gradual increase in the final distance between the two bodies after 100 years, as the length of time steps increases, whereas the final distance after 100 years gained by the fourth order Runga-Kutta method shows fluctuations for shorter time step lengths, which is slightly more when sun is stationary relative to the frame of reference. As time step length reaches about 10 days, the distance, determined by the Runge-Kutta method, between the two bodies decreases, and the final distance after 100 years between the Earth-like body and the Sun-like body starts varying a lot with a change in time step length, meaning that the Runge-Kutta method is very unstable for large time steps, greater than approximately 10 days for this situation. However, both the Velocity-Verlet and forth order Runge-Kutta method seems to have stabilized for time steps smaller than or equal to 5 days, both for the situation with a stationary Sun and for the situation with both the Sun and Earth moving relative to the

coordinate system. Hence, the time step length of 5 days is used to study the stability of the two methods for long time periods in Fig. 3.2.

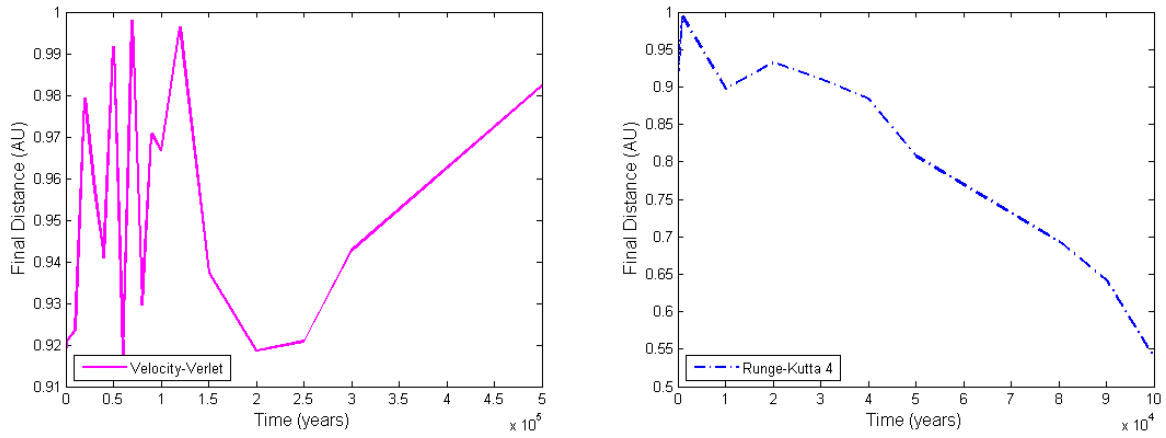

Figure 3.2: The final distance as a function of time with a time step length of 5 days for both the Velocity-Verlet and Runge-Kutta method with both the Earth and Sun moving relative to the frame of reference. After 2.5×10^4 years, the Earth continuously moves towards the Sun, in the Runge-Kutta method, whilst the distance between the Earth and Sun still fluctuates between 0.92 AU and 1 AU for the Velocity-Verlet method after 5×10^5 years.

When allowing for motion of both the Earth and the Sun relative to the frame of reference, both the Velocity-Verlet method and the forth order Runge-Kutta method show that the distance between the Earth-like body and the Sun-like body will fluctuate with time, which seems reasonable from the fact that Earth's orbit around the Sun is not circular but elliptical. From the Runge-Kutta method it is found that after approximately 2.5×10^4 years the Earth will start moving rapidly towards the Sun, and after 10^5 years the distance between the Sun-like object and the Earth-like object is only around 0.55 AU. This is, however, not seen in the Velocity-Verlet method. In the real solar system, the orbit of the Earth is in addition to the gravitational pull from the Sun also affected by the presence of the other planets. In the absence of these remaining planets and other objects in the solar system, the movement of the Earth-like object considered in this two-body system will, obviously, be different from the orbit known from astrophysics. However, it is estimated that the absence of other objects in the solar system, will not cause this rapid motion of Earth and Sun towards each other after only in the order of $10,000$ years, and hence it is concluded that the rapid motion seen in the rightmost figure of Fig. 3.2 is due to instabilities in the fourth order Runge-Kutta method presented in Sec. 2.2.2, yielding a greater stability in the Velocity-Verlet method than in the Runge-Kutta method for solving this two-body problem.

The table below shows the respective computational times for the fourth order Runge-Kutta method and the Velocity-Verlet method for computing the final position after 1 year using different time step length.

Table 3.2: Computational time for the fourth order Runge-Kutta method and the Velocity-Verlet method for different time steps during 1 year.

# time steps	Comp. time RK4	Comp. time VV
1	6	2
10	25	8
10^4	8.4×10^3	4.6×10^3
10^6	8.0×10^5	3.6×10^5

From the table it is evident that the computational time of both the fourth order Runge-Kutta method and the Velocity-Verlet method is more or less proportional to the number of time steps. Furthermore, the Velocity-Verlet method seems to be faster than the Runge-Kutta mthos for all investigated number of time steps. Together with the grater stability of the Velocity-Verlet method than the Runge-Kutta method, this yields that it is an advantage to use the Velocity-Verlet method to study this two-body system.

3.2 Testing Runge-Kutta and Velocity-Verlet for Sun-Earth-Mars System

Before using the codes to calculate the movement of the stars in a star cluster, the codes are tested on a known system like the Sun-Earth-Mars system. This is because these bodies move in a very predictable and stable fashion. So when increasing the time the orbits should remain almost equal to the initial orbit, and almost circular. As it is possible to see in Fig. 3.3 that it's the case. Where the mass, initial position and, velocity is set as given in Tab. 3.3.

Table 3.3: Mass, initial position and initial velocity of Sun, Earth and Mars when running the Runge-Kutta 4 algorithm for this three-body problem. The Earth and Mars are set to orbit in the $x-y$-plane at $z = 1$ AU with the distance 1 AU and 1.5 AU to the Sun, respectively, which is not physically true. However, this initialization of position and velocity is reasonable to illustrate the validity of the Runge-Kutta method and Velocity-Verlet method presented in Sec. 2.2.

	mass [$M_\odot$]	$\mathbf{r}_{initial}$ [AU]	$\mathbf{v}_{initial}$ [AU/day]
Sun	1.0	$(1.0, 1.0, 1.0)$	$(0.0, 0.0, 0.0)$
Earth	3.0×10^{-6}	$(2.0, 1.0, 1.0)$	$(0.0, 0.017, 0.0)$
Mars	3.2×10^{-7}	$(-0.5, 1.0, 1.0)$	$(0.0, 0.014, 0.0)$

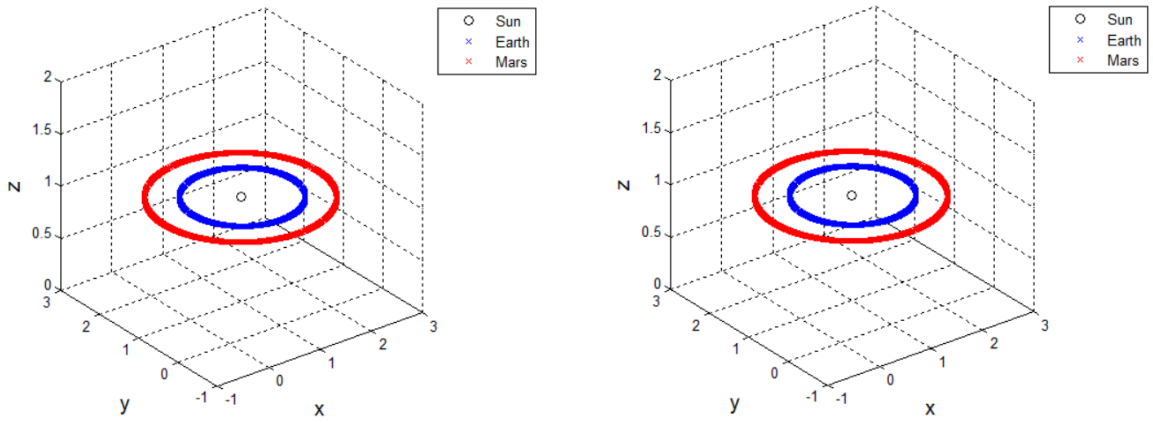

Figure 3.3: Time evolution of the simplified system of Sun-Earth-Mars over a time period of 20 years using Runge-Kutta (leftmost) and Velocity-Verlet (rightmost) method with a time step length of 1 day. The masses, initial positions, and initial velocities of the three objects are given in Tab. 3.3.

Calculating the initial and final energy of the Sun-Earth-Mars system according to the source code presented in Sec. 2.4, gives the Tab. 3.4 for different time periods with step length of 1 day.

Table 3.4: The final energy after different time periods computed by both the first order Runge-Kutta method and the Velocity-Verlet method for the Sun-Earth-Mars-like system with initial energy of $2.37 \times 10^{-9} \mathrm{M}_{\odot}\mathrm{AU}^2/\mathrm{days}^2$.

Time period (years)	Final energy (RK4)	Final energy (VV)
1	-1.44×10^{-9}	-1.44×10^{-9}
10	-1.47×10^{-9}	-1.47×10^{-9}
100	-1.48×10^{-9}	-1.48×10^{-9}
1000	-1.44×10^{-9}	-1.44×10^{-9}

From the results in Tab. 3.4 the energy is not changing much over time, and with the precision of the constants, e.g. the gravitational constant $G = 2.96 \times 10^{-4}\mathrm{AU}^3/\mathrm{days}^2 \cdot \mathrm{M}_{\odot}$, and the time steps, this gives a good picture of the evolution of the energy. And it gives the impression that the methods used are giving a good estimates to the real movement of the particles. And since it is working for this model with three particles it is possible to assume that it will work for N bodies as well.

3.3 Time Step Length in the N-body Problem

When estimating the time step length required to study the evolution of the star cluster consisting of $N = 100$ particles initially uniformly distributed in a sphere of radius 20 ly with normal distributed masses with mean $10\mathrm{M}_{\odot}$, the initial and final distribution of particles in the radial direction for a finite time period is studied for different time step lengths. The time period, considered, is chosen to be of the order of the characteristic time τ_{crunch}, given in Eq. (3.1). τ_{crunch} is the finite time at which the system with $N \to \infty$

particles with no or low initial velocity collapses into a singularity [4].

$$\tau_{crunch} = \sqrt{\frac{3\pi}{32G\rho_0}} \tag{3.1}$$

With the units light years, years and solar masses $M_\odot$, the value of the gravitational constant is, according to Sec. 2.1, $G = 1.536 \cdot 10^{-13} \text{ly}^3/\text{yr}^2\text{M}_\odot$, whilst the mass density of $N = 100$ particles with a mean mass of $10\text{M}_\odot$ within the sphere of radius 20ly is $\rho_0 = 10\text{M}_\odot/(4/3\pi(20\text{ly})^2)$, yielding that the τ_{crunch} becomes

$$\tau_{crunch} = \sqrt{\frac{12\pi^2 \cdot 20^3}{32 \cdot 1.536 \cdot 10^{-13}} \cdot 10 \cdot 3} \text{ yr} \approx 8.0 \times 10^7 \text{ yr} \tag{3.2}$$

The histograms in Fig. 3.4 and 3.5 show the initial and final radial position of 100 particles in a sphere of radius 20 ly, initially at rest, interacting only through the Newtonian force between all particles, after 10^7 years with different step lengths. The masses and initial position of the 100 particles are generated by the functions given in Sec. 2.3. the final positions given in the histograms of Fig. 3.4 are computed with the Velocity-Verlet method for N bodies, whilst the final positions given in the histograms of Fig. 3.5 are computed with the fourth order Runge-Kutta method for N bodies.

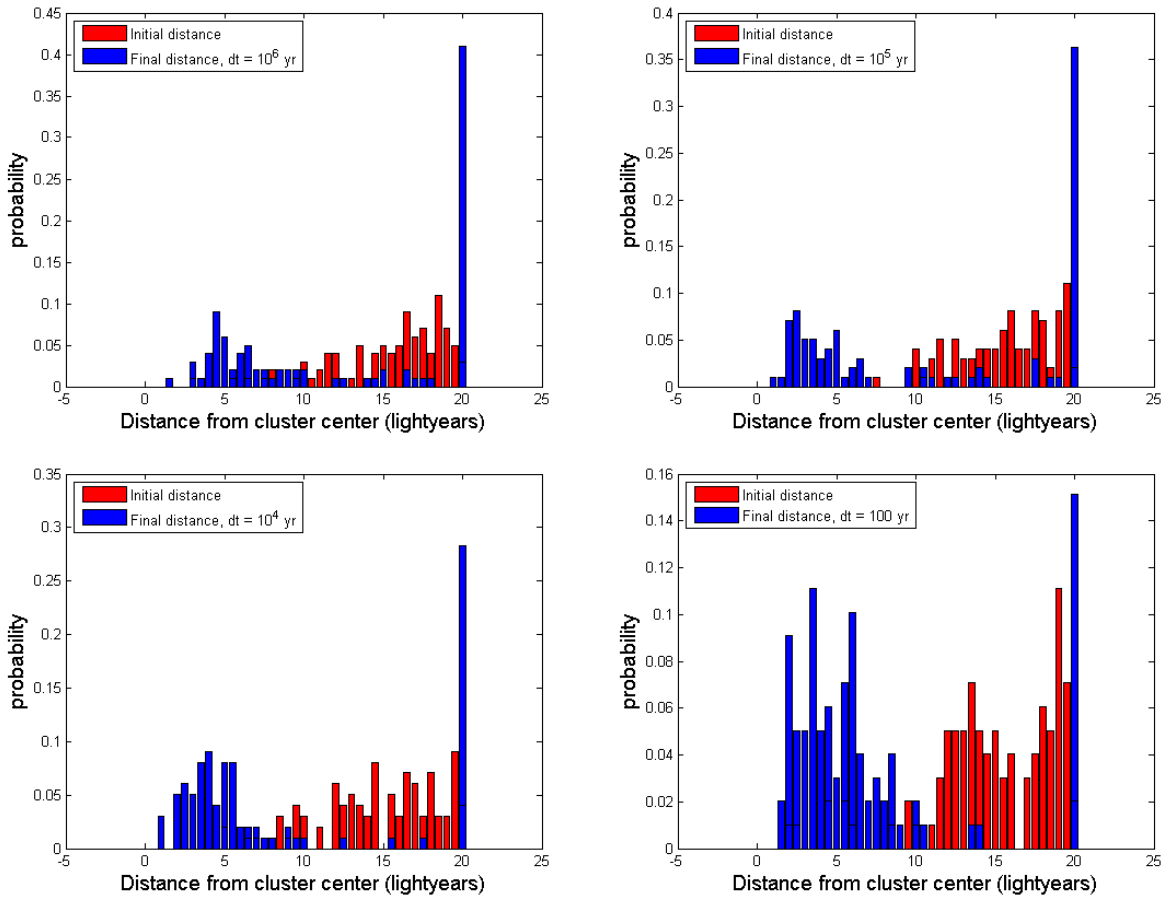

Figure 3.4: Initial and final position position over a time period of 10^7 years for 100 particles in a sphere of radius 20 ly computed by the Velocity-Verlet method, for different time step lengths dt. The masses of the particles are normal distributed around $10\text{M}_\odot$ with a standard deviation of $1\text{M}_\odot$, whilst the initial positions are computed from an initial uniform density within the sphere. The depicted particle probability at the distance $r = 20$ ly correspond to particles that are actually at around 20 ly away from the cluster center after 10^7 years, but it also accounts for the particles that has escaped the cluster, that is $R > 20$ ly.

By comparing the four histograms in Fig. 3.4 for different time step lengths, in seems that the final radial distribution of the particles after 10^7 years consists of three distinct features. Close to the center, at a distance smaller that $5-10$ ly, a high density seems to appear. In addition a great part of the particles seem to be ejected from the cluster. This is seen by the high probability bar at a distance $R = 20$ ly from the cluster center. This bar represents both particles at the edge of the cluster and particles that have been ejected from the cluster. For long time steps, this ejection seems to be greater than for short time steps. That is, for a step length of $dt = 10^6$ yr, the amount of particles that has been ejected from the cluster after 10^7 yr is roughly 40%, whilst the amount of ejected particles with a step length of $dt = 100$ years is only around 15%. This is similar to the result gained in Sec. 3.1, where the distance between the bodies after 100 years in the Sun-Earth-like system was computed to be greater for larger time step lengths than for short time step lengths for the Velocity-Verlet method, as well, as seen in Fig. 3.1. The third feature of the histograms in Fig. 3.4 is the small density of particles at more than 10 ly away from the cluster center. All of the histograms with different time step lengths show these same features, apart from the number of ejected particles, which vary rapidly with decreased step length.

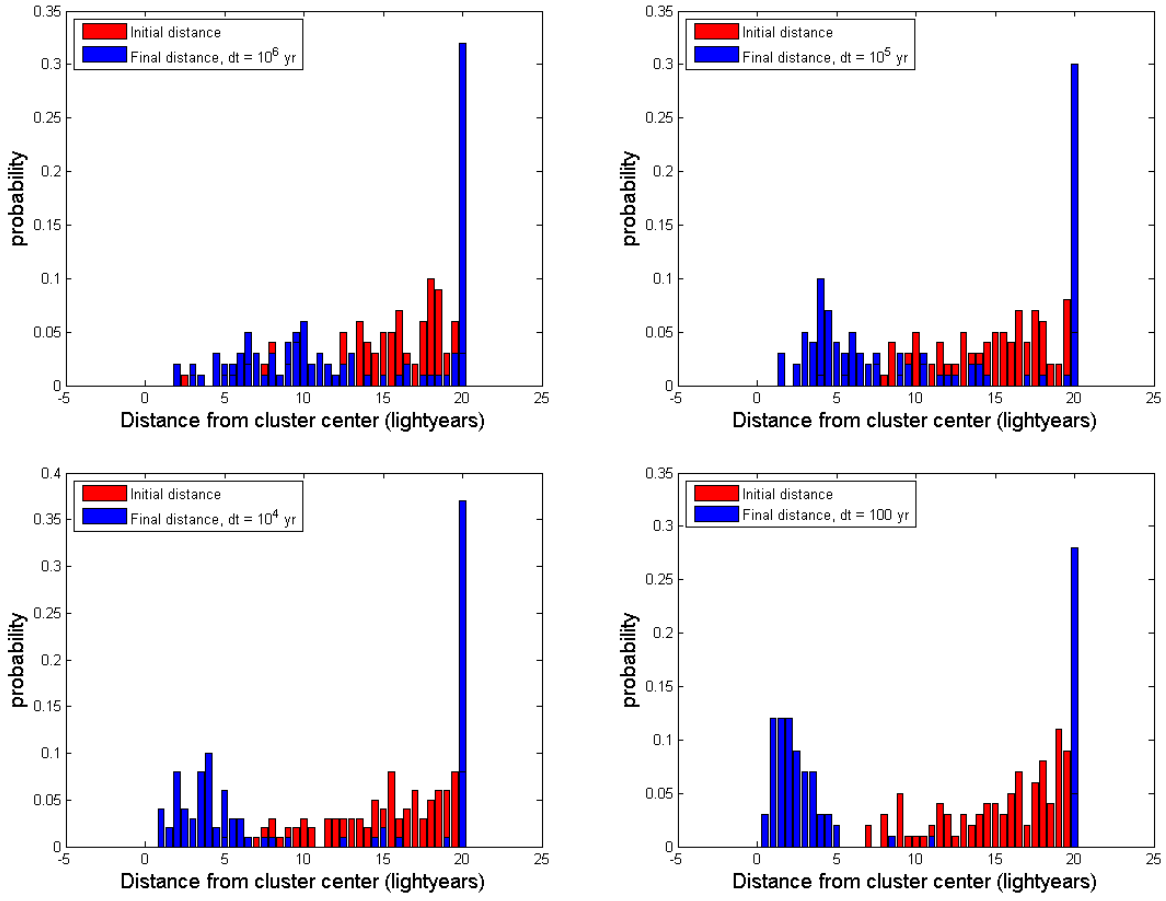

Figure 3.5: Initial and final position position over a time period of 10^7 years for 100 particles in a sphere of radius 20 ly computed by the Fourth order Runge-Kutta method, for different time step lengths dt. The masses of the particles are normal distributed around $10M_\odot$ with a standard deviation of $1M_\odot$, whilst the initial positions are computed from an initial uniform density within the sphere. The depicted particle probability at the distance $r = 20$ ly correspond to particles that are actually at around 20 ly away from the cluster center after 10^7 years, but it also accounts for the particles that has escaped the cluster, that is $R > 20$ ly.

As for the radial distribution after 10^7 years computed by the Velocity-Verlet method, the final distribution shown in Fig. 3.5, computed by the fourth order Runge-Kutta method, show the three features: high density close to the center, low density at a more than 10 ly from the center, and an ejection of particles from the cluster. However, for long time steps, that is $dt = 10^6$ yr, these features are not as distinct, yielding that this step length is too long for the Runge-Kutta method. Unlike the Velocity-Verlet method, the amount of particles that are ejected from the cluster using the Runge-Kutta method seems, however, to be stable for all depicted step lengths at around 30%.

In the two particle case the Velocity-Verlet method seemed like it was the most stable method over time. Looking at the N particle case the Velocity-Verlet is more dependent of the choice of time step, whilst the Runge-Kutta 4 method is more stable towards the change in time-step. This might have a lot to say for the stability of the system and the computational time. Therefore the Runge-Kutta 4 method seems like the better choice when looking at times greater than τ_{crunch}.

After finding the positions of the particles after a time in the order of a τ_{crunch}, the star cluster have not collapsed into a mass in the middle. Therefore using τ_{crunch} as a unit of time to study the further behaviour is a natural conclusion. For this to work the gravitational constant has to be transformed into units that fits and for the N particle case. Using Eq. (3.1) it's possible to find the gravitational constant as a function of particles N.

$$G = 986.96 \frac{1}{N} \frac{\mathrm{ly}^3}{\tau_{crunch}^2 \mathrm{m}_{mean}} \tag{3.3}$$

Now when looking at timescales larger than τ_{crunch}, the time steps have to be calculated from the new unit of time τ_{crunch}. From the histograms in Fig. 3.5 the best time step length to choose is 10^5 years. If this is then transformed into the new time scale it is

$$dt = 10^{-3} \tau_{crunch}$$

3.4 Stability and Equilibrium of $N = 100$ System

From the histograms in Fig. 3.5, it is expected that the extend of the cluster will vary with time. In the histograms of Fig. 3.6, the final position of 100 uniformly distributed particles with Gaussian distributed masses, as previous, is computed with the fourth order Runge-Kutta method after $1\tau_{crunch}$ and $4\tau_{crunch}$ with the same step length, $dt = 10^{-3}\tau_{crunch}$. Similar histograms are made for the final position after $0.5\tau_{crunch}$, $1.5\tau_{crunch}$, $2\tau_{crunch}$, $2.5\tau_{crunch}$, $3\tau_{crunch}$, as well. However, these histograms are not explicitly shown in this work.

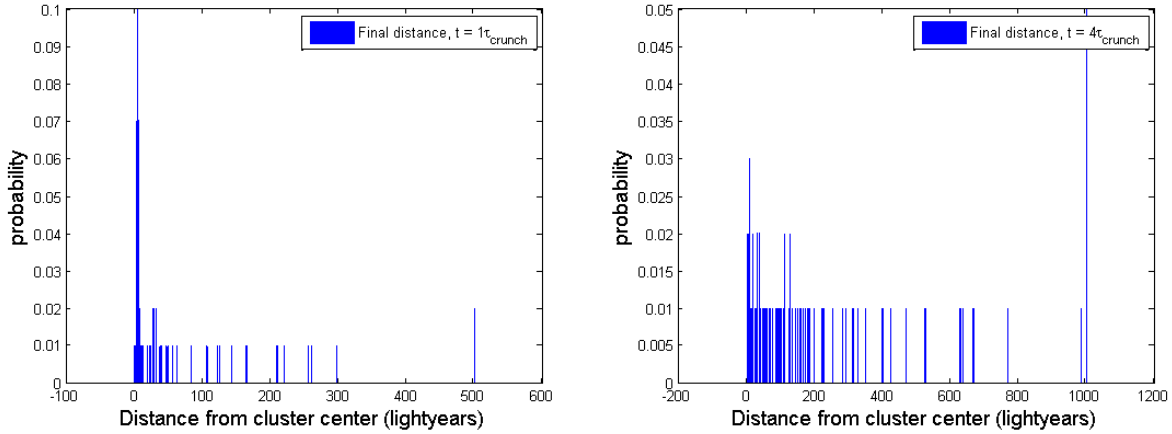

Figure 3.6: Final radial position of 100 particles after $1\tau_{crunch}$ and $4\tau_{crunch}$ with a step length of $10^{-3}\tau_{crunch}$. The "high" densities at 500 ly and 1000 ly represent particles further away from the cluster center than 500 ly or 1000 ly, respectively. The extend of the cluster is here defined as the first particle than has a distance of 100 ly to the next particle further from the cluster center. Hence after $1\tau_{crunch}$, the extend of the cluster is 166 ly, whilst after $4\tau_{crunch}$, the cluster extend is 525 ly.

The knowledge of the cluster extend as a function of time gained from Fig. 3.6 is plotted in the figure below. The plotted data show reaching of an equilibrium after approximately $2\tau_{crunch}$. Another argument for reaching the equilibrium after about $2\tau_{crunch}$ is the number of particles within a sphere of radius of 10 ly, which is inspired by the great density of particles within this sphere compared to density outside this sphere after 10^7 years, shown in the histograms of Fig. 3.5.

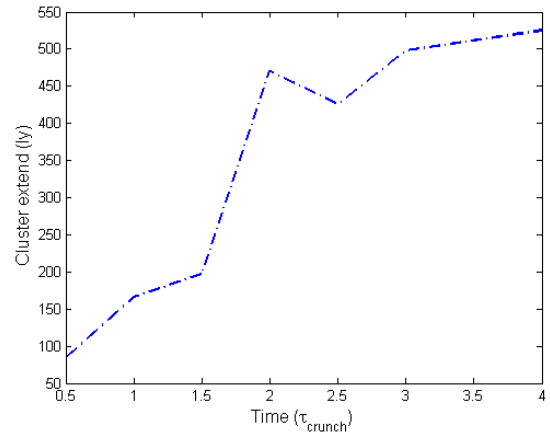

Figure 3.7: Plot of the cluster extend as a function of the final time. The cluster extend is determined from histograms similar to the ones shown in Fig. 3.6.

Time [τ_{crunch}]	# particles
0.5	31
1	62
1.5	38
2	23
2.5	17
3	6
4	12

Table 3.5: Number of particles within a sphere of radius 10 ly after various times.

3.5 Energy of the N-body Runge-Kutta Method

To study the energy inside the system, only the particles that have not escaped are of interest. To find out which stars are still in the system, the kinetic energy K of the particle has to be smaller than the potential

energy to remain bound to the system. These are the bound particles.

$$E_{kin} \leqslant -E_{pot} \tag{3.4}$$

Giving a positive energy total if the kinetic energy is greater than the potential, and thus allowing the star to escape.

In Tab. 3.6 the initial energies, the number of bound particles, the final energies of the bound particles and, energy loss in percentage is shown. Her we can see how much energy is lost and if there is a connection between the number of particles in the system to begin with. There is an energy loss in the system, this i due to the ejected particles taking some energy away from the system. Yielding that the Energy is not conserved in the system.

N =	Number of bound particles	$E_{initial}$	E_{Bound}	deviation %
30	22	-184057	-231578	-0.25
50	40	-307864	-484146	-0.5725
70	51	-421925	-475997	-0.1281
100	82	-592577	-807914	-0.3634

Table 3.6: The total initial energy and the total final energy of the bound particles and the percentage energy loss after a time period of a time period $1\tau_{crunch}$, for different numbers of initial particles.

From the energy loss there is no apparent connection between number of particles and how much energy the system looses in percentage. But as the number of particles is not larger than 100 and the system has not been tested several times for these values, it's not possible to make a conclusion if the energy loss is dependent on the number of particles.

3.6 Distribution of Potential and Kinetic Energy of Bound Particles after a Finite Time

The virial theorem states the following relation for the kinetic energy E_{kin} and potential energy E_{pot} of a bound gravitational system in equilibrium.

$$2\langle E_{kin}\rangle = -\langle E_{pot}\rangle \tag{3.5}$$

$\langle E_{kin}\rangle$ and $\langle E_{pot}\rangle$ represents the time average of the kinetic and potential energy, respectively, but by the ergodic hypothesis, this can be made the ensamble average [4].

To study whether the virial theorem of Eq. (3.5) is fulfilled with the computed algorithms, the final kinetic and potential energy of the bound particles of a system initially consisting of 70 particles after a time period of τ_{crunch}. It is expected that the fraction of the mean of the kinetic energy and the mean of the potential energy will be close to -2, even though it in Sec. 3.4 is found that the equilibrium is not reached before than after a time period of $2\tau_{crunch}$.

In addition, to get rid of some of the numerical instabilities, a smoothing function, as argued for in Sec. 2.2.3, is introduced. To check the consistency with the virial theorem of the fourth order Runge-Kutta method when including the smoothing function, similar histograms for the distribution of the kinetic

energy and potential energy after a finite time period as shown in Fig. 3.8 is generated to determine the mean of the kinetic energy and potential energy. The bound particles are found by the same argument as in [1] that a particle is bound if the total energy of that particle is negative, that is the particle is bound if the absolute value of the potential energy is greater than the absolute value of the kinetic energy.

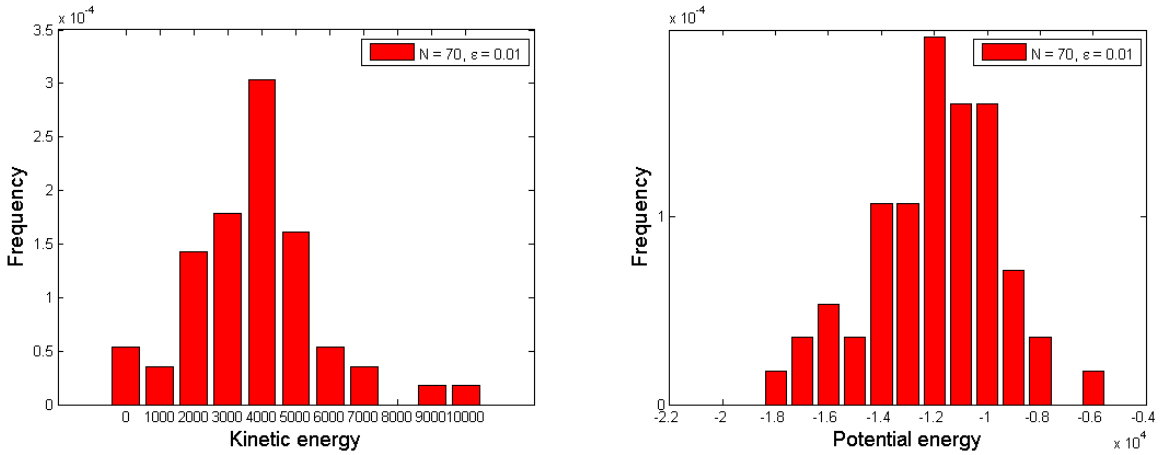

Figure 3.8: Distribution of the kinetic and potential energy of bound particles in the star cluster after a time period of $1\tau_{crunch}$ with a step length of $1\times10^{-4}\tau_{crunch}$ and $\varepsilon = 1\times10^{-2}$ ly for 70 particles with normal distributed masses, uniformly distributed within a sphere generated by the functions presented in Sec. 2.3.

The kinetic and the potential energy of the bound particles seem to follow a more or less uniform distribution after a finite time period. This is also seen when computing histograms for different values of ε. The mean of these distributions for the kinetic energy and potential energy after are shown in the table below for various values of ε.

ε (ly)	$\langle E_{kin}\rangle$	$\langle E_{pot}\rangle$	$\langle E_{pot}\rangle/\langle E_{kin}\rangle$
0	952.0	-2409	-2.53
0.1	4036	-7886	-1.95
10^{-2}	1119	-2846	-2.54
10^{-3}	2548	-4860	-1.91
10^{-4}	2531	-4489	-1.77
10^{-5}	2129	-3677	-1.73
10^{-6}	567.5	-1774	-3.13
10^{-7}	1333	-2444	-1.83
10^{-8}	1368	-2635	-1.93

Table 3.7: Mean of kinetic energy and potential energy for bound particles after a time period of $3\tau_{crunch}$ with a step length of $10^{-4}\tau_{crunch}$ for a star cluster with initially 70 particles uniformly distributed in a sphere of radius $20R_{\odot}$. The computed potential energy is divided by two, since in the computation the potential energy is doubled when summing up for all particles.

From Tab. 3.7 it is seen that for no values of ε, the virial theorem is fulfilled. However, the fraction $\langle E_{pot}\rangle/\langle E_{kin}\rangle$ approaches the value -2, when ε is introduced for all values except $\varepsilon = 10^{-6}$. The fraction

[1] FiXme Note: ref to magnus-sec

of the energies is in the figure below plotted as a function of ε, and it is seen from the figure, as well from Tab. 3.7 that most of the choices of ε will be an improvement of the result compared having $\varepsilon = 0$, according to the virial theorem. Since 10^{-7}ly corresponds to $1.36R_{\odot}$ this is chosen as the optimal ε, due to the fact that is has the physical meaning of being similar to what one can expect the radius of the considered particles to be.

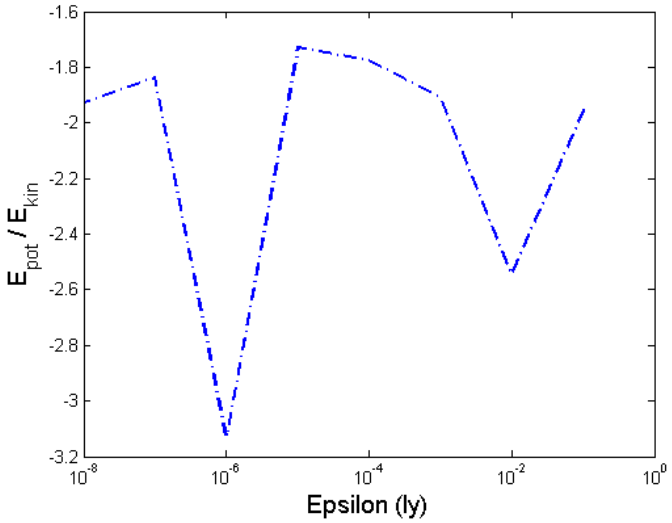

Figure 3.9: The value of the energy fraction $\langle E_{pot} \rangle / \langle E_{kin} \rangle$, that according to the virial theorem should equal -2, as a function of ε introduced in Eq. (2.11).

The histograms in Fig. 3.10 below show the distribution of the kinetic energy and potential energy of the bound particles after a time period of $3\tau_{crunch}$, hence after reached equilibrium. The distribution of the energies of the bound particles seems no longer to be Gaussian as was seen for a time period of $1\tau_{crunch}$ in Fig. 3.8.

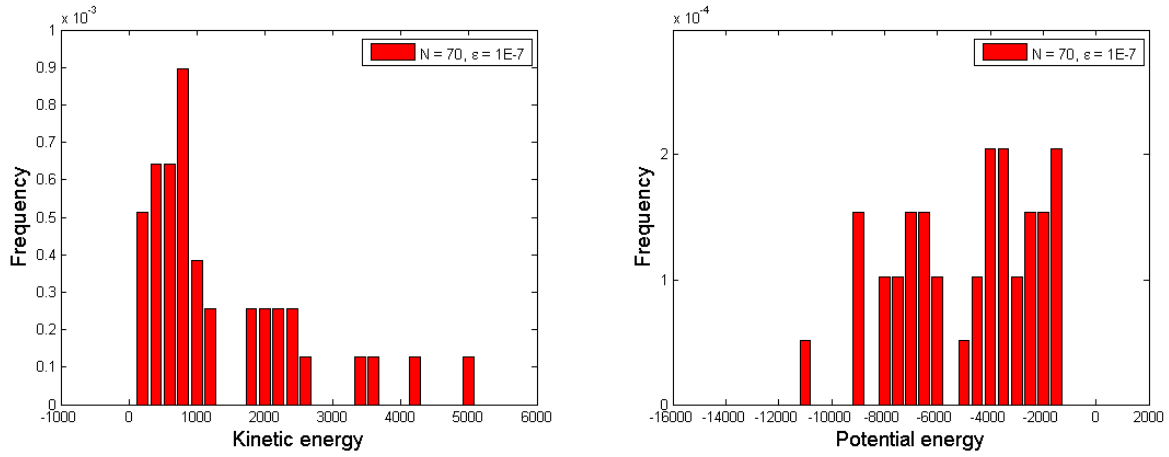

Figure 3.10: Distribution of the kinetic and potential energy of bound particles in the star cluster after a time period of $1\tau_{crunch}$ with a step length of $1 \times 10^{-4}\tau_{crunch}$ and $\varepsilon = 1 \times 10^{-2}$ ly for 70 particles with normal distributed masses, uniformly distributed within a sphere generated by the functions presented in Sec. 2.3.

CHAPTER

4 CONCLUSION

BIBLIOGRAPHY

[1] M. Hjorth-Jensen, "Computational physics - lecture notes fall 2015," August 2015.

[2] R. Toral and A. Chakrabarti, "Generation of gaussian distributed random numbers by using a numerical inversion method," *Computer Physics Communications*, vol. 74, pp. 327–334, 1993.

[3] H. T. Ihle, "On the, not entirely trivial, issue of getting uniformly distributed random coordinates within a sphere."

[4] "Project 5, astronomy project, n-body simulation of an open galactic cluster." URL: `https://github.com/CompPhysics/ComputationalPhysics1/blob/gh-pages/doc/Projects/Project5/project5_astro.pdf`. [Online access; 11/12-2015].