Solving the Gravitational N-Body Problem Using Finite Difference Methods

Karlo Krakan — 31451123

Department of Physics and Astronomy, University of British Columbia, Vancouver BC CAN V6T1Z1

Abstract

Exact solutions to the gravitational N-body problem are known only for $N = 2$ and specific cases of $N = 3$. In this paper, a solution to the N-body problem for arbitrary N will be established through the use of finite difference methods. The correctness of the code is tested via a convergence test and a test of conservation of total energy of a system. After the correctnes is established, a few numerical experiments are conducted including the orbits of Earth and Jupiter around the Sun, a binary star system, and a gravitational slingshot.

Contents

1 Introduction

The N-body problem is one of the oldest and most difficult problems to solve in physics. The first advancements in this area started almost immediately after Newton laid out his theory of gravitation. Newton believed the problem was impossible to solve, and as far as we know, he was correct. However, strides were made when the problem was solved for $N = 2$ by Johann Bernoulli. In this case, Bernoulli was able to solve the problem, but only after he had made the simplifying assumption that one of the bodies remains fixed in space. The solution was a good approximation, but evidently not enough to claim that the problem was "solved," since Newton's law of gravitation does not allow for any of the bodies to be fixed. The exact solution was found later by considering the rotation of the bodies about the center of mass of the system.

$N \geq 3$ is a remarkably more difficult problem to solve exactly. In fact, no exact solutions for this problem are known and almost 400 years after the problem was first laid out, physicists are still looking for both analytic and numerical solutions to this problem. Despite this, major advancements were made with the discovery of choreographies, first by Lagrange in 1772 and later extended considerably with the discovery of new choreographies in the 2000s. This process was of course considerably aided by the use of computers, which were unavailable to the classical physicists. Numerical techniques have allowed us to solve many problems which would be seemingly impossible to the likes of Newton. We are following that trend.

2 Mathematical Formulation

2.1 Equations of Motion

Consider a system of N points each with mass m_i, $i = 1, ..., N$. We know that the gravitational force (with units nondimensionalized such that G=1) exerted on each particle by every other particle is given by Newtons law of gravitation

$$m_i\mathbf{a}_i = \sum_{j=1,j\neq i}^{N} \frac{m_i m_j}{r_{ij}^3}\mathbf{r}_{ij}$$

where

$$\mathbf{r}_{ij} = (x_j - x_i)\hat{x} + (y_j - y_i)\hat{y} + (z_j - z_i)\hat{z}$$

and r_{ij} is simply the magnitude of $\mathbf{r}_{ij}$. Expressed as a differential equation

$$\frac{d^2\mathbf{r}_{ij}}{dt^2} = \sum_{j=1,j\neq i}^{N} \frac{m_j}{r_{ij}^3}\mathbf{r}_{ij}$$

The total energy of the system at time t is given by

$$E(t) = T(t) + V(t)$$

where

$$T(t) = \frac{1}{2}\sum_{i=1}^{N} m_i\mathbf{v}_i^2$$

is the total kinetic enery of the system at time t and

$$V(t) = -\sum_{i=1}^{N}\sum_{j=i+1}^{N}\frac{m_i m_j}{r_{ij}}$$

is the total potential energy of the system at time t.

2.2 Finite Difference Approximation

We discretize a finite continuous time domain $0 \leq t \leq t_{max}$ using a level parameter l in way such that

$$n_t = 2^l + 1$$
$$\Delta t = \frac{t_{max}}{n_t - 1} = \frac{t_{max}}{2^l}$$
$$t^n = (n-1)\Delta t$$

where n_t is the number of time steps, Δt is the discrete time step, and t^n is is the nth time step[1]. In order to approximate the time evolution of the system of N bodies to the second order, it will be necessary to use the second order Taylor expansions of the first and second order derivatives[1], given belowu

$$\frac{d^2\mathbf{r}_i^n}{dt^2} = \frac{\mathbf{r}_i^{n+1} - 2\mathbf{r}_i^n + \mathbf{r}_i^{n-1}}{\Delta t} + O(\Delta t^2)$$

$$\frac{d\mathbf{r}_i^n}{dt} = \frac{\mathbf{r}_i^{n+1} - \mathbf{r}_i^{n-1}}{2\Delta t} + O(\Delta t^2)$$

where $\mathbf{r}_i^n = \mathbf{r}_i(t^n)$ [1].

3 Code Correctness

3.1 The Test Case

Naturally, it is expected that any numerical solution conform to a known exact solution. Consider two bodies, m_1 and m_2. Suppose that at some instant in time they are located on the same axis, then let $R = r_1 + r_2$ be a conserved quantity. The center of mass of this system is located at (on the same axis as the bodies)

$$R_{CM} = \frac{m_1 r_1 + m_2 r_2}{M} \tag{1}$$

where M is the total mass of the bodies. With the center of mass at the origin, the bodies then are located at

$$r_1 = \frac{m_2}{M} R \tag{2}$$

$$r_2 = -\frac{m_1}{M} R \tag{3}$$

Now, imagine that the bodies are rotating in circular orbits about the center of mass. In this case, the speeds of the bodies are constant and the only objects exerting forces on the bodies are the objects themselves. Therefore, in natural units, Newton says that

$$\frac{m_1(m_1+m_2)}{m_2}\frac{v_1^2}{R} = \frac{m_2(m_1+m_2)}{m_1}\frac{v_2^2}{R} = \frac{m_1 m_2}{R^2} \tag{4}$$

Suppose also that the center of mass is not in motion, then total momentum of the system is 0 and we have

$$m_1 v_1 + m_2 v_2 = 0 \tag{5}$$

Then, solving the system of equations given by (4) and (5) and using (2) and (3), we get

$$v_1 = \frac{\sqrt{m_2 r_1}}{R} \tag{6}$$

$$v_2 = \frac{\sqrt{m_1 r_2}}{R} \tag{7}$$

So the initial conditions given by (2), (3), (6) and (7) will produce a pair of bodies rotating in circular orbits about the center of mass.

This simple system will be used to test the correctness of our code. We will choose the system such that the initial conditions and masses are

$$m_1 = m_2 = 1$$

$$r_1 = (1, 0, 0)$$

$$r_2 = (-1, 0, 0)$$

$$v_1 = (0, 1/2, 0)$$

$$v_2 = (0, -1/2, 0)$$

3.2 Conservation of Energy

For any physical system, the total energy of the system should remain constant, unchanging in time. As we have computed the energy of the system at every timestep, we are able to plot the energy versus time. A straight line is expected and any substantial deviation from a straight line indicates a problem with the code. However, with certain initial conditions, energy conservation is expected to be broken due to the limitations of our code (for example, two bodies approaching eachother head on will break energy conservation when they meet in the middle). With resepect to the test case, this type of conservation breaking is not expected and conservation of energy may be used as a convincing code correctness test.

3.3 Convergence Tests

We also expect that the error in the calculated energy of the system, $dE(t) = E(t) - E(0)$, will grow as $O(\Delta t^2)$ as the system evolves. We also expect that the error in the calculated energy would be reduced by a factor of four whenever Δt is reduced by two. Therefore, we will also determine whether or not the system shows this kind of behaviour.

4 Numerical Experiments

4.1 Orbits of Jupiter and Earth

We will now perform a few numerical experiments. First, we wish to approximate the motion of the Earth and Jupiter around the Sun. To avoid complexity in choosing initial conditions, we will approximate the planets orbits as circular. The sun is placed at the origin and the Earth and Jupiter are scaled such that the earth's distance from the sun is 1. The masses of the planets are scaled such that the mass of the sun is 1. Finally, the initial velocities of the planets are scaled such that the Earths orbital velocity is 1. The planets are placed on the x-axis with their velocities pointing upwards in the y-direction.

4.2 Binary Star System

Binary star systems are systems of two stars rotating about their center of mass. In section 3.1, we were able to derive the initial conditions that create such a system (for circuar orbits, at least) and the results used to establish the code correctness. However, we used bodies with the same mass and in this case, we will vary the mass of the bodies. We choose $m_1 = 3.0$ and $m_2 = 1.4$, which may be taken to represent a small black hole and a white dwarf if the scale is such that the mass of the Sun is 1. Though, unlike in the preceeding experiment, it can not be known whether this system corresponds to a real physical system since the distance between the bodies was chosen arbitrarily. On the other hand, the motion is, of course, exactly analagous. The velocities of the stars are chosen according to the formula outlined in Section 3.1

4.3 Gravitational Slingshot

When the total energy of two body system is greater than zero, a body may approach a more massive body in such a way that the smaller body is slingshotted into another direction without being coupled to the larger body (since the total energy is > 0). Thus, we choose our initial conditions such that the total energy is greater than zero, the more massive body, M, is at the origin with zero initial velocity and the less massive body, m, is displaced upwards and to the right compared to M and given a leftwards initial velocity.

4.4 Elliptical Orbit

This section is included as a precursor to an unexpected result. As we were varying initial conditions for the gravitational slingshot, a peculiar elliptic orbit was found. This will be discussed further in the next section.

5 Results

5.1 Correctness Tests

The code we produced may be viewed in the Appendix for independent examination. The main program is called *nbody.m* and it depends on another function called *nbodyaccn.m* which may also be viewed in the Appendix. The program was written in the MATLAB programming language. The program is able to take an arbitrary number of bodies with their initial conditions and produce the solution as a multidimensional position array. This array is formatted such that it may be fed into Matt Choptuiks *nbodyout.m*[1] function for use in the simulation program xfpp3d. The correctness is established via a convergence test and a test of conservation of energy as outlined in Sections 3.2 and 3.3, the results of which are shown below.

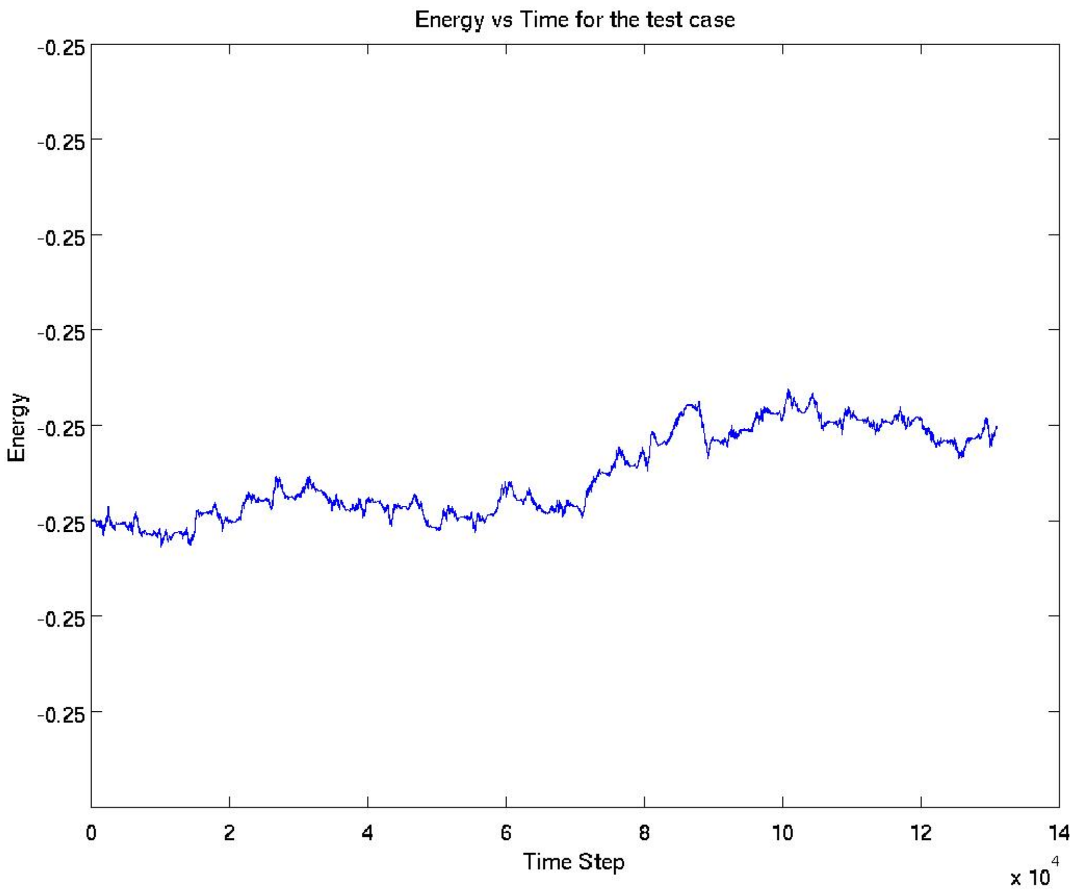

Figure 1: Energy vs Time for the test case

Inspecting Figure 1 it is clear that the program produces the expected result, namely that energy is conserved. Of course there is some error but it is evidently extremely small. This, however, is to be expected as we do expect the error in the calculated energy to grow as $O(\Delta t)$ along with inherent computational errors. The maximum error can be simply calculated and we find it to be 7.2769e-08, which is astonishingly small and only serves as a testament to the accuracy of the program. The following figure, Figure 2, shows the results of the convergence test.

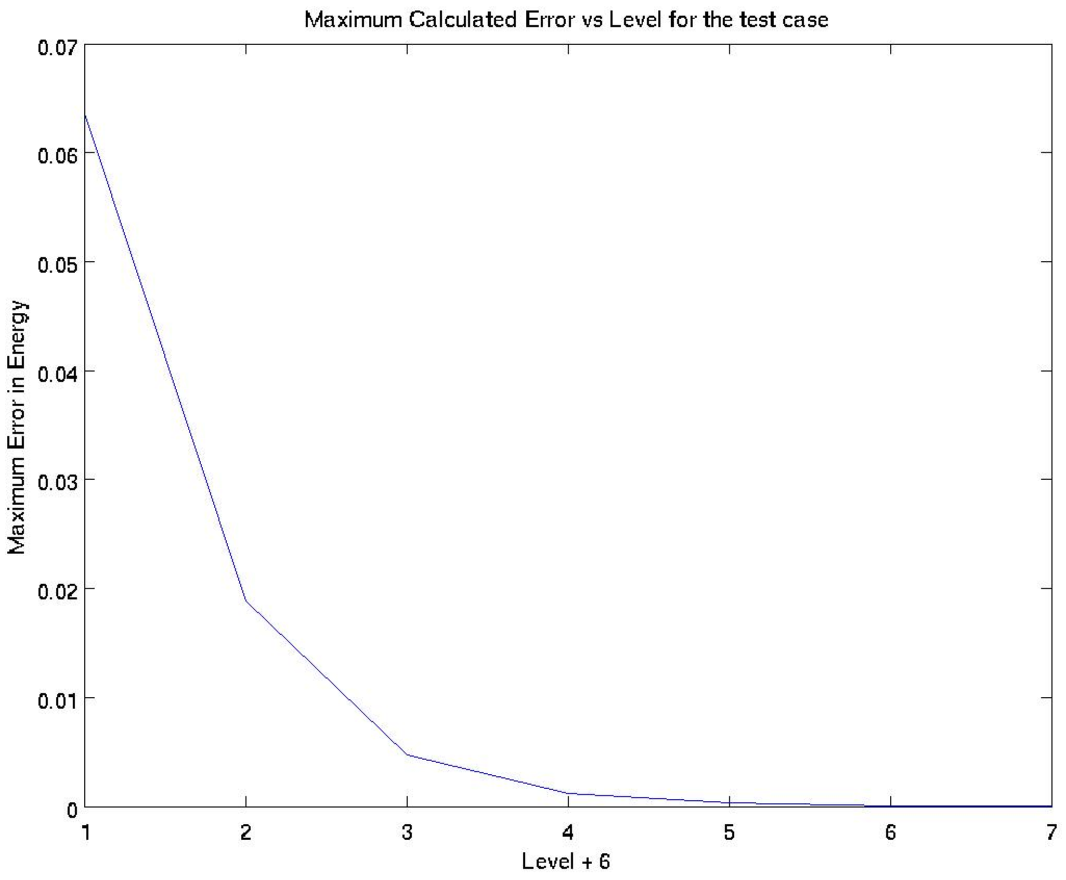

Figure 2: Maximum Error in Energy vs Level for the test case

We performed this test by varying Δt, where every unit increase of the level corresponds to a halving of Δt. It may be hard to see in Figure 2, but every halving of Δt does correspond to a reduction by a factor of four in the error. This is the behaviour we were expecting and shows that, along with the energy conservation test, that the program is accurate. No further testing is needed, as the results are general and we have no reason to expect that the program would be innacurate for any other set of initial conditions (so long as the particles do not collide which is a limitation of the program).

5.2 Numerial Experiments

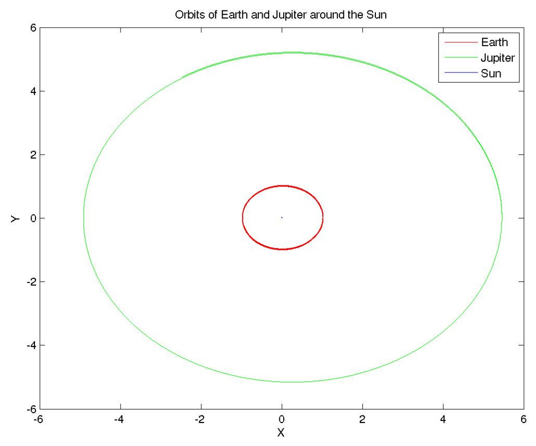

Figure 3: Sun, Earth and Jupiter system. The barely visible blue dot at the center is the motion of the sun.

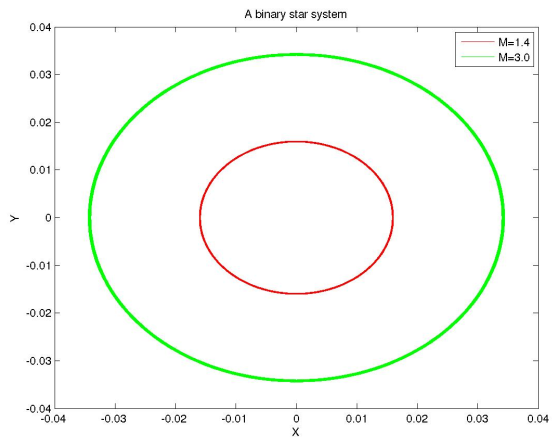

Figure 4: A binary star system

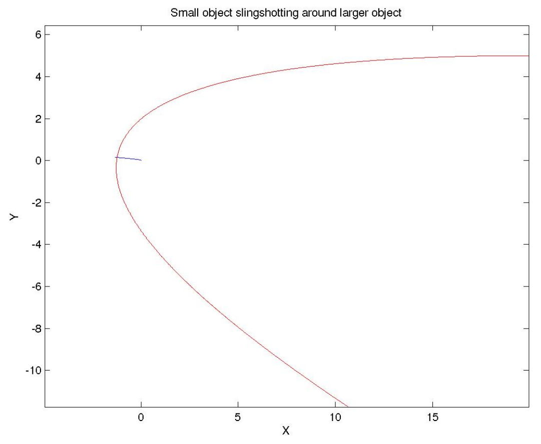

Figure 5: A gravitational slingshot

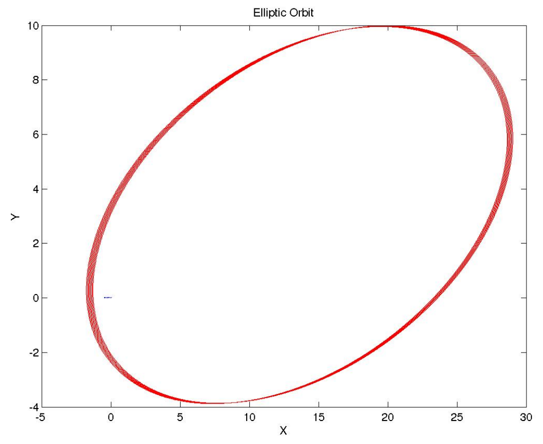

Figure 6: An odd elliptical orbit.

Figures 3 to 6 show the results of the numerical experiments. From the graphs we can see that all of the experiments produced the expected results. The initial conditions for these numerical experiments may viewed in the Appendix. In Figure 3, it should be noted that the motion of the Sun at the center of the graph is barely visible. Evidently in this model, the motion of the sun is almost nill. This may indicate that models of the solar system with the Sun remaining fixed at the center can be viable and produce very good approximations to the real motions of the objects in the solar system.

Figure 6 shows a very peculiar and unexpected result. That is, an elliptical orbit such that the more massive body is not at a focus of the ellipse.

3 dimensional animations of these simulations may be viewed at http://omega.phas.ubc.ca/ karlokr/. The download links are below.

Sun, Earth, Jupiter: http://omega.phas.ubc.ca/ karlokr/sunearthjupiter.mpg

Binary Star: http://omega.phas.ubc.ca/ karlokr/binarystar.mpg

Slingshot: http://omega.phas.ubc.ca/ karlokr/slingshot.mpg

Elliptic Orbit: N/A

6 Conclusion

Evidently, our solution suffers from a number of flaws. The first is that it is unable accurately model many gravitational systems. This can be seen when attempting to simulate a system where any two bodies collide. Since the bodies are point particles in our model, collisions between two particles will break energy conservation when the bodies collide. Furthermore, it could very well be the case that when two bodies are very close to each other, the positions of the bodies will update in a manner that places the bodies in positions that would not be allowed in a real system. These issues may be resolved in a future update that would upgrade the bodies from point particles to actual, rigid spheres. Also, it may be advantageous to update the program with a collision checking system such that if the bodies approach eachother at an angle they scatter and if they approach head on they become one new body with a new radius that is a function of the masses of the bodies.

Another annoyance of our solution is the speed at which the calculation are performed. For many systems requiring great accuracy and extremely small time steps, MATLAB just does not perform the calculations at an optimal speed. In the future it may advantegous to port the program to a faster language such as C. This would allow us to simulate much more systems, especially those that require very high accuracy, and at a higher speed.

Finally it may be necessary to really consider the implications of the odd elliptic orbit that was found during the numerical experimentation. Since the sun is not at a focus, it may be of interest to theorists to prove whether or not such an orbit is possible. If such an orbit is possible, which the simulation shows that it indeed is, then it may also be of interest to astronomers to track asteroids and comets and determine whether these objects orbit the Sun in the expected elliptic orbits or whether they orbit in these types of orbits where the Sun is not at a focus of the orbit.

7 References

[1] Choptuik, M. (2012). *Phys 210: Intro to Computational Physics: Course Website.* http://laplace.physics.ubc.ca/210/Notes.html

A Appendix

A.1 nbody.m

```
function [t, r, v, m, E, T, V] = nbody(tmax, level, r0, v0, m0, tracefreq)
% nbody Solves the n-body problem using an O(deltat^2) FDA
%
% Input arguments
%
%      tmax:      (real scalar) Final solution time.
%      level:     (integer scalar) Discretization level.
%      r0:        (nb*3 matrix) Initial positions of the n bodies in
%                 Cartesian coordinates, nb is the number of bodies
%      v0:        (nb*3 matrix) Initial velocities of the n bodies in
%                 Cartesian coordinates, nb is the number of bodies
%      m0:        (1*nb array) Masses of the n bodies, nb is the number
%                  of bodies
%      tracefreq: (optional integer scalar) Frequency of tracing output,
%                  0 disables tracing.
%
% Output arguments
%
%      t:       (real vector) Vector of length nt = 2^level + 1 containing
%               discrete times (time mesh).
%      r:       (nb*3*nt matrix) Matrix of dimension nb*3*nt, where nt is the
%               total number of timesteps and nb is the number of bodies,
%              containing computed values of the positions of the n bodies
%               at discrete times t(n).
%      v:       (nb*3*nt matrix) Matrix of dimension nb*3*nt, where nt is the
%               total number of timesteps and nb is the number of bodies,
%               containing computed values of the positions of the n bodies
%               at discrete times t(n).
%      m:       (real vector) Vector of length nb, where nb is the number of
%               bodies, containing the masses of the n bodies
%      E:       (real vector) Vector of length nt = 2^level + 1 containing
%               the total energy at discrete times t(n).
%      T:       (real vector) Vector of length nt = 2^level + 1 containing
%               the total kinetic energy at discrete times t(n)
%      V:       (real vector) Vector of length nt = 2^level + 1 containing
%               the toal potential energy at discrete times t(n).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  % Tracing control: if 6th arg is supplied base tracing on that input,
  % otherwise use local defaults.
  if nargin > 5
    if tracefreq == 0
      trace = 0;
    else
      trace = 1;
    end
  else
    trace = 1;
    tracefreq = 100;
  end

  if trace
    fprintf('In nbody: Argument dump follows\n');
    tmax, level, r0, v0, m0
  end

  % Number of bodies
  nb = length(m0);

  % Number of time steps; create t, r, v,  m, E, K and T matrices
  nt = 2^level + 1;
  t = linspace(0.0, tmax, nt);
  r = zeros(nb, 3, nt);
  v = zeros(nb, 3, nt);
  m = zeros(nb, 1);
  E = zeros(1, nt);
  T = zeros(1, nt);
  V = zeros(1, nt);

  % The time step
  deltat = t(2) - t(1);

  % Initialize the masses of the particles
  m = m0;

  % Initialize the first value of the particles velocities
  v(:,:,1) = v0;

  % Initialize the first two values of the particles positions
  r(:,:,1) = r0;
  r(:,:,2) = r0 + deltat * v0 + 0.5 * deltat^2 * nbodyaccn(r0, m);

  if trace
    fprintf('deltat=%g \n', deltat);
    fprintf('r(:,:,1)=\n');
    fprintf([repmat('%f\t', 1, size(r(:,:,1), 2)) '\n'], r(:,:,1)');
    fprintf('r(:,:,2)=\n');
    fprintf([repmat('%f\t', 1, size(r(:,:,2), 2)) '\n'], r(:,:,2)');
  end

  %---------------------------
  % Perform the simulation
  %---------------------------

  for n = 2:nt-1
    % Compute position of the particles at later times
    r(:,:,n+1) = 2 * r(:,:,n) - r(:,:,n-1) + deltat^2 * nbodyaccn(r(:,:,n), m)
        ;

    % Compute the velocities of the particles
    v(:,:,n) = ( r(:,:,n+1) - r(:,:,n-1) ) / (2 * deltat);
  end

  % Determine the value of the velocity at the final time step
  v(:,:,nt) = 2 * v(:,:,nt-1) - v(:,:,nt-2);

  % Compute the total kinetic energy at all times
  for s=1:length(t)
    for i=1:nb
      T(s) = T(s) + 0.5 * m(i) * ( (v(i,1,s))^2 + (v(i,2,s))^2 + (v(i,3,s))^2
          );
    end
  end

  % Compute the total potential energy at all times
  for l=1:length(t)
    for i=1:nb
                  if i+1 <= nb
                          for j=i+1:nb
                                  V(l) = V(l) - ( (m(i) * m(j)) / ( (r(j,1,l) -
                                      r(i,1,l))^2 + (r(j,2,l) - r(i,2,l))^2 + (r
                                      (j,3,l) - r(i,3,l))^2 )^(1/2) );
                          end
                  end
      end
    end

  % Compute the total energy at all times
  for l=1:length(t)
    E(l) = T(l) + V(l);
  end

end
```

A.2 nbodyaccn.m

```
function [a] = nbodyaccn(r, m)
  % The number of particles
  n = length(m);

  % Initiate the acceleration matrix
  A = zeros(size(r));

  % Compute the acceleration matrix
  for i = 1:n
    l = 1:n;
    l(i) = [];
    for j = l
      direction = r(j,:) - r(i,:);
      magr = sum(direction.^2)^(1/2);
      coupling = m(j) / magr^3;
      A(i,:) = coupling * direction(:);
    end
    end

    a = A;

end
```

A.3 Initial Conditions

```
%--------------------
% Sun, earth, jupiter
%--------------------
%nb = 3;
%m0 = [0.000003, 0.001, 1.00];
%r0 = [1.02, 0, 0; 5.46, 0, 0; 0, 0, 0];
%v0 = [0, 0.98, 0; 0, 0.417, 0; 0, 0, 0];

%[t r v m E T V] = nbody(100, 18, r0, v0, m0, 1);

%--------------------
% Binary star
%--------------------
%nb = 2;
%m0 = [3.0, 1.4];
%r0 = [-(1.4/4.4)*0.05, 0, 0; (3.0/4.4)*0.05, 0, 0];
%v0 = [0, -sqrt(1.4*(1.4/4.4)*0.05)/0.05, 0; 0 sqrt(3.0*(3.0/4.4)*0.05)/0.05
    0];

%[t r v m E T V] = nbody(100, 17, r0, v0, m0, 1);

%------------------
% Elliptical Orbit
%------------------
%nb = 2;
%m0 = [1000, 1];
%r0 = [0 0 0; 20 10 0];
%v0 = [0 0 0; -5 0 0];

%[t r v m E T V] = nbody(100, 16, r0, v0, m0, 1);

%------------------
% Slingshot
%------------------
%nb = 2;
%m0 = [1000, 1];
%r0 = [0 0 0; 20 5 0];
%v0 = [0 0 0; -10 0 0];

%[t r v m E T V] = nbody(100, 16, r0, v0, m0, 1);
```