Numerical study of the solar system

Timofey Golubev, Hao Lin, Xingze Mao

April 6, 2017

Abstract

Codes were developed to study the motion of planets in the solar system, in the spirit of object-oriented programming. Two ordinary differential equation (ODE) solution methods, Euler's method and Velocity Verlet method, were implemented. We started with a simple two-body Sun-Earth system to test the correctness of our implementations. It was then followed by a study of the three-body Sun-Earth-Jupiter system and a full-scale simulation of the evolution of the solar system. It was found that the Velocity Verlet method is sufficiently accurate to ensure the closedness of the orbits and to conserve total energy and angular momentum. Finally, we also looked at the perihelion precession of Mercury and confirmed that a relativistic correction is key to the explanation of this phenomenon.

1 Introduction

In the solar system, all 9 planets are orbiting around the Sun which results in a 10-body problem. In this project we want to simulate the trajectories of all the planets by considering the impact of the pairwise gravitational force on their movements. The time evolutions of positions and velocities are described by a set of coupled first order ODEs. Their numerical solutions are found by either Euler's method or the Velocity Verlet method. These methods are implemented in a solver class in C++ and are applied to study different subsystems in the solar system.

Section 2 discusses the methods we use in the project, including the unit conversions used to simplify the differential equations, the Euler's method and the Verlet method used to solve the differential equations. A short description of our code is also listed in this part. In Section 3, we study the evolutions of different subsystems of the solar system, including the Sun-Earth system, the Sun-Earth-Jupiter system, the entire solar system, and the perihelion precession of Mercury. Plots of orbits are shown and discussed. Further remarks and conclusions are drawn in Section 4.

2 Methods

First we consider the Earth orbiting around the Sun. The gravitational force experienced by the Earth is

$$F_G = \frac{GM_\odot M_{Earth}}{r^2}$$

where $M_\odot$ is mass of the Sun and M_{Earth} is the mass of the Earth, r is the distance between the Earth and the Sun, and G is the gravitational constant. We choose the xy-plane for the motion of the Earth with the Sun located at the origin. With Newton's second law, the above equation can be written separately for the x and y components of acceleration:

$$\frac{d^2x}{dt^2} = \frac{F_{G,x}}{M_{Earth}} = \frac{GM_\odot}{r^2}\frac{x}{r}$$

$$\frac{d^2y}{dt^2} = \frac{F_{G,y}}{M_{Earth}} = \frac{GM_\odot}{r^2}\frac{y}{r}$$

2.1 Unit Conversion

SI units will give us some inconvenience since the distance between the Earth and the Sun is about 1.5×10^{11}m, which is an astronomically large number. The equations can be much simplified if we adopt the astronomical units: 1AU= 1.5×10^{11}m, which is the average distance between the Sun and the Earth. Considering the time it takes for the planet to finish one orbit, we will use years as the unit of time.

If we assume the trajectory of the Earth is a circle, the following relations will hold:

$$F_G = \frac{M_{Earth}v^2}{r} = \frac{GM_\odot M_{Earth}}{r^2} \tag{1}$$

Since $v = 2\pi\frac{1\text{AU}}{1\text{yr}}$, we will have the following equation:

$$v^2 r = GM_\odot = 4\pi^2\frac{\text{AU}^3}{\text{yr}^2} \tag{2}$$

Therefore if we use astronomical units for length and year as the unit for time, $GM_\odot$ can be just replaced by $4\pi^2$. The differential equations we need to solve become:

$$\begin{aligned}\frac{d^2x}{dt^2} &= \frac{4\pi^2}{r^2}\frac{x}{r} = \frac{4\pi^2}{x^2+y^2}\frac{x}{\sqrt{x^2+y^2}} \\ \frac{d^2y}{dt^2} &= \frac{4\pi^2}{r^2}\frac{y}{r} = \frac{4\pi^2}{x^2+y^2}\frac{y}{\sqrt{x^2+y^2}}\end{aligned} \tag{3}$$

2.2 Euler's method

If we use Taylor expansion to expand $f(x+h)$ around x, we will have

$$f(x+h) = f(x) + hf'(x) + f''(x)\frac{h^2}{2} + O(h^3), \tag{4}$$

A first order approximation for the derivative is $f'(x) = \frac{f(x+h)-f(x)}{h}$ which is the Euler method. If we want to have higher precision in our calculation dealing with the differential equation, we can choose a smaller step size h.

As for the second order differential equation, for example Newton's law, $\frac{d^2x}{dt^2} = a$, we can define $v = \frac{dx}{dt}$, rewrite it as two coupled first order equations:

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= a \end{aligned} \tag{5}$$

With Euler's method[1], we can write the equations as

$$\begin{aligned} x_{i+1} &= x_i + hv_i + O(h^2) \\ v_{i+1} &= v_i + ha_i + O(h^2) \end{aligned} \tag{6}$$

Code implementation:

Acceleration is first calculated by summing up all the forces from other planets in the system.

```
for(int n=0; n<total_planets; n++){
        if(n==current_index)continue;
        planet &other = all_planets[n];
        Accel_x += current.X_Acceleration(other,
Gconst,corrections);
        Accel_y += current.Y_Acceleration(other,
Gconst,corrections);
        Accel_z += current.Z_Acceleration(other,
Gconst,corrections);
}
```

Next both the positions and the velocities are updated by the method mentioned above.

```
//update positions by 1 time step
for(int j=0;j<3;j++){
current.position[j]+=TimeStep*current.velocity[j];
}
//update velocities by 1 time step
```

```
//(NOTE: the forces already have the proper sign)
current.velocity[0]+=Accel_x*TimeStep;
current.velocity[1]+=Accel_y*TimeStep;
current.velocity[2]+=Accel_z*TimeStep;
```

The positions, velocities, and total system energies are written to the output files at each time step. Also, as unit tests of the algorithm, total system energies are calculated and output at each time step and initial and final angular momenta are compared.

2.3 Velocity Verlet method

Another popular method used to solve differential equations is the Verlet method[1]. Here we will use the Taylor expansion twice to expand both $f(x+h)$ and $f(x-h)$ around x.

$$f(x+h) = f(x) + hf'(x) + f''(x)\frac{h^2}{2} + O(h^3)$$
$$f(x-h) = f(x) - hf'(x) + f''(x)\frac{h^2}{2} + O(h^3)$$

Summing up the two equations above, we have the relation between values of $f(x)$ at three adjacent points:

$$f(x+h) = 2f(x) - f(x-h) + f''(x)h^2 + O(h^4) \tag{7}$$

The error is of 4^{th} order, because all the odd terms vanish when we sum up the two equations. The problem about this algorithm is that it is not self-starting. In other words, we need both $f(x)$ and $f(x-h)$, which is usually not available, to calculate $f(x+h)$.

One alternative way is the Velocity Verlet method[1] explained below. The Taylor expansion of both the velocity and the displacement are

$$\begin{aligned} x_{i+1} &= x_i + hx'_i + x''_i\frac{h^2}{2} + O(h^3) \\ v_{i+1} &= v_i + hv'_i + v''_i\frac{h^2}{2} + O(h^3) \end{aligned} \tag{8}$$

If we consider again the Taylor expansion $v'_{i+1} = v'_i + v''_i h + O(h)$, we will have the $h^2 v'' = h(v'_{i+1} - v'_i) + O(h^3)$. If we plug this relation into equation (8), we will have:

$$\begin{aligned} v_{i+1} &= v_i + hv'_i + \frac{h}{2}(v'_{i+1} - v'_i) + O(h^3) \\ &= v_i + \frac{h}{2}(v'_{i+1} + v'_i) + O(h^3) \end{aligned} \tag{9}$$

Thus we will have the Velocity Verlet method:

$$\begin{aligned} x_{i+1} &= x_i + hv_i + \frac{h^2}{2}v'_i + O(h^3) \\ v_{i+1} &= v_i + \frac{h}{2}(v'_{i+1} + v'_i) + O(h^3) \end{aligned} \tag{10}$$

If we have the position and velocity of the first point, with this vesion of the Verlet method, we will be able to calculate the velocity and position of next point.

Code implementation:

First the acceleration due to gravitational forces exerted by all other planets `other` is calculated for each planet `current`. `Corrections` passes a boolean value to the `Acceleration` function to allow to add relativistic corrections which were used for calculating precession of Mercury.

```
//Calculate pairwise gravitational acceleration
for(int n=0; n<total_planets; n++){
            if(n==current_index)continue;//skip this case
            planet &other = all_planets[n];
            ith_accel[current_index][0] +=
current.X_Acceleration(other, Gconst, corrections);
            ith_accel[current_index][1] +=
current.Y_Acceleration(other, Gconst, corrections);
            ith_accel[current_index][2] +=
current.Z_Acceleration(other, Gconst, corrections);
 }
```

Then the position is updated with current position, velocity and acceleration

```
for(int j=0; j<3; j++) {
            current.position[j] += current.velocity[j]
*TimeStep + 0.5*TimeStep_sqrd*ith_accel[current_index][j];
 }
```

The accelerations are recalculated again at the new positions by summing contributions from all other planets

```
for(int n=0; n<total_planets; n++){
            if(n==current_index)continue;
            planet &other = all_planets[n];
            next_accel[0] += current.X_Acceleration
(other, Gconst, corrections);
            next_accel[1] += current.Y_Acceleration
(other, Gconst, corrections);
            next_accel[2] += current.Z_Acceleration
(other, Gconst, corrections);
 }
```

The new velocity is updated based on the velocity at previous points and the accelerations (`ith` and `next`) at the two adjacent points

```
// Calculate new velocity components for current planet
for(int j=0; j<3; j++) {
        current.velocity[j] += 0.5*TimeStep*
(ith_accel[current_index][j] + next_accel[j]);
}
```

The positions, velocities, and total system energies are written to output files at each time step. Also initial and final angular momenta are compared.

2.4 Mercury Precession

We study the orbit of Mercury around the Sun with no other planets present when a general relativistic correction is added to the Newtonian gravitational force. The force is now

$$F_G = \frac{GM_{Sun}M_{Mercury}}{r^2}\left(1 + \frac{3l^2}{r^2c^2}\right) \tag{11}$$

where $l = |r \times v|$ is the angular momentum per unit mass and c is the speed of light. This results in the perihelion of Mercury's elliptical orbit to precess with a precession angle $tan(\theta_p) = \frac{y_p}{x_p}$ where x_p, y_p are coordinates of the perihelion.

Code Implementation:

In order to calculate the perihelion precession of Mercury, it is necessary to use a very high time resolution. Therefore, in order to reduce computational cost, we reduce the writing to output file to only the positions for the last simulated orbit of Mercury. The perihelion of the last orbit was found by finding the position along the orbit which had the minimum radius from the origin. Then the angular change of the perihelion was calculated. Since if z-component of velocity is small compared with x,y-component in the initial condition, setting z-component to zero will not change the trajectory much, however, lots of computational time can be saved. The trajectory calculation was implemented in C++ using our solver class and the perihelion calculation was implemented in Matlab.

```
last_orbit_start = total_steps-steps_per_orbit;
for i=last_orbit_start:total_steps
    x=data(mercury_indices(i),3);
    y=data(mercury_indices(i),4);
    final_radii(j) = sqrt(x^2+y^2);
    j=j+1;
end
[final_perihelion, index]=min(final_radii);
perihelion_index = index-1+last_orbit_start;
x_final_peri = data(mercury_indices(perihelion_index),3);
y_final_peri = data(mercury_indices(perihelion_index),4);
perihelion_angle = atand(abs(y_final_peri/x_final_peri));
```

3 Results and discussion

3.1 Sun-Earth system

The Sun-Earth system is studied first with the Euler and Velocity Verlet algorithms. As the Sun has mass much larger than the earth, the center of mass for Sun-Earth system is almost inside the Sun, we assume the Sun is fixed and the Earth is circulating around the Sun under the force of gravity. The stability of the two algorithms is studied and also comparison between two algorithms' timing is made. The energy and angular momentum of such a system is calculated.

3.1.1 Stability of Euler and Velocity Verlet method

In all numerical calculations, the discretization is done with some level of approximation. Higher order approximations will give us a smaller error. The velocity Verlet method has the error in the 4^{th} order ($O(h^4)$) and Euler method gives us a 2^{nd} order error ($O(h^2)$). With smaller step size, the derivative is approximated better. If we have too large a step, the error might result in inaccuracies of the algorithm. Different sizes of steps are chosen to calculate the trajectory of the Earth. Figure 1 shows the results with two different time steps: 0.05 yr and 0.005 yr for two methods. The Velocity Verlet method is stable in both cases, while Euler's method has unclosed trajectory in both case.

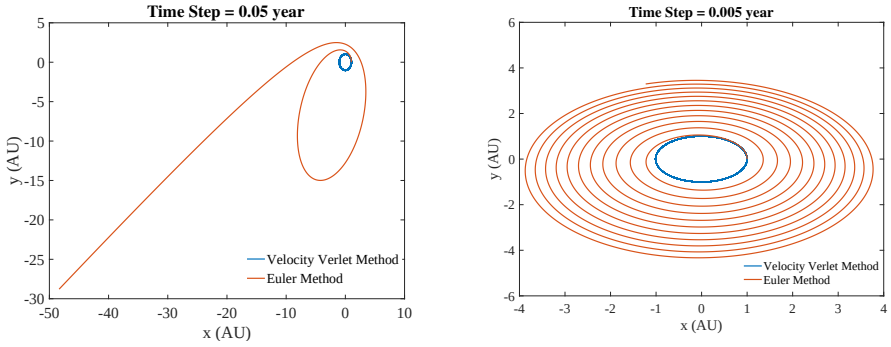

Figure 1: Stability of the Sun-Earth system using Euler's method

3.1.2 Timing of Euler and Velocity Verlet method

Trajectory calculation is the calculation intensive part of the code. Based on equation (6), 2 summations and 2 multiplications are needed to update the velocity and displacement in one direction. In the xy-plane, when we have N points in the calculation, we have about $8N$ FLOPS per iteration for Euler's method. With equation (10), there are 4 summations and 6 multiplications to update the velocity and displacement in one direction. So when we apply the Velocity Verlet method to the Sun-Earth system, the FLOP count per iteration

is about $20N$ where N is the number of mesh points in our calculation. With the time step $h = 0.00005yr$, the elapsed time for our Euler method is 26.4563s and the Velocity Verlet method is 26.9962s. Note that this includes the writing out all positions and velocities at each time-step to file, and energy and momentum conservation checks which are imbedded in our solver class. For practical real-time calculations, the Velocity Verlet method is almost as fast as Euler's method, but has much higher precision.

In order to test the true CPU time of the algorithms, we removed the writing to file and all conservation checks. In this case, for the same time step of $h = 0.00005yr$, the elapsed time for Euler's method is 0.1121s and Verlet method is 0.2034s. Therefore most of the CPU time in our simulations was used for the print to file and conservation checks. We used this knowledge to enable running of order 10^9 mesh points when calculating precession of Mercury.

3.1.3 Conservation of Energy and angular momentum

With the right initial conditions, the gravitational force will always be perpendicular to the velocity of the Earth and the Earth moves in a circle. Gravitational force does no work to the Earth, so kinetic energy stays the same. Also the distance between the Earth and the Sun stays the same, so potential energy is also conserved. The results for the Sun-Earth system calculated using both the Euler and Velocity Verlet methods are shown in Figure 2.

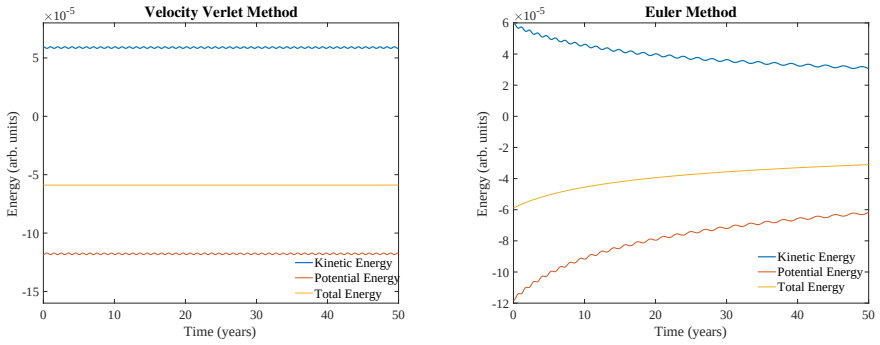

Figure 2: Sun-Earth system total energies as a function of time for time step = 0.0005 yr.

In the Velocity Verlet method simulation, both potential and kinetic energy are almost constant over time. The small oscillation is believed to come from the numerical stability error resulting in a not perfectly circular orbit. The oscillation decreases when we use smaller time steps. The flat line in the center of Figure 2 is the total energy for the Sun-Earth system which is conserved with a relative total energy change of only $-4.7 \times 10^{-6}\%$ for a time step of 0.0005yr. When the trajectory is almost a perfect circle, both kinetic and potential energy are conserved. The angular momentum of the earth is also calculated and found

to be conserved for the Verlet method with relative change of only $-10^{-12}\%$ for the same time step. Since there are no external forces acting on the Sun-Earth system, the angular momentum of the system is expected to be conserved and since the Sun is fixed, the angular momentum of the earth should also be conserved.

In the Euler method results, the energies are clearly not conserved with a relative total energy change of -47.3% for time step of 0.0005yr. The angular momentum is also not conserved with a relative total change of 37.7% for the same time step. This is due to the numerical errors from the method giving unstable trajectories as seen in Figure 1. Comparing the Euler method energies plot with the trajectories we can understand the observed changes in energies. As the orbital radius increases, the kinetic energy decreases since the planet is losing velocity. At the same time, Earth's potential energy is becoming more positive since the gravitational attraction becomes weaker with larger radius.

3.2 Escape velocity

The escape velocity for a planet which has a distance of 1 AU from the Sun is calculated and compared with analytical results. The least kinetic energy that the planet must have to escape is equal to the gravitational potential energy

$$\frac{GM_\odot m}{r} = \frac{mv^2}{2} \tag{12}$$

where m is the mass of the planet, v is the velocity, and r is in the unit of AU. With equation (2), and $GM_\odot = 4\pi^2$, the escape velocity is

$$v_{escape} = \sqrt{2GM_\odot} = \sqrt{2 \times 4\pi^2} = 2\sqrt{2}\pi AU/yr \tag{13}$$

The trajectories of a single planet with various starting velocities are plotted in Figure 3. When the initial velocity is 2π, the planet follows a nearly circular orbit. When initial velocity becomes $> 2\pi$, the orbit becomes elliptical, with the eccentricity increasing with increasing velocity. When the velocity reaches v_{escape}, the planet escapes the gravitational interaction with the Sun.

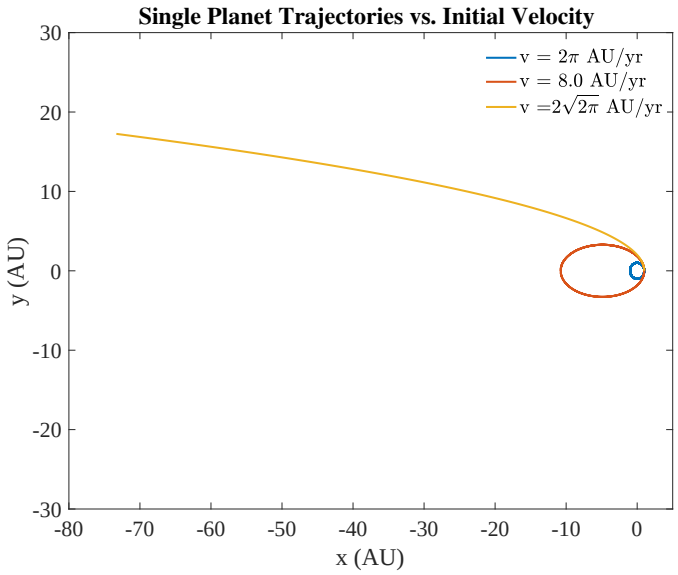

Figure 3: Trajectories for a planet which has a distance of 1Au from the Sun.

3.3 Sun-Earth-Jupiter system

Besides the gravitational attraction from the Sun, the Earth's motion is also affected by other planets. As Jupiter is the heaviest planet in the solar system, we add the Jupiter into our system and modify the code to solve a three-body system. The center of this three-body system is fixed in the Sun. With the gravitational attraction from Jupiter, the trajectory of the earth will be affected. Three different masses ($M_{Jupiter}$, $100M_{Jupiter}$, and $1000M_{Jupiter}$) are given to Jupiter in our calculations and the results are shown in Figure 4. The Sun is in the center, the smaller circle is trajectory of the Earth, and Jupiter is circulating in the outer circle. With increasing mass of Jupiter, we can see the trajectory of the Earth getting wider as shown in the upper two plots in Figure 4. The plots in the bottom show the trajectories when we set the mass of Jupiter to be 1000 times its real mass. The left one is the calculation with time step $h = 0.05$yr and the right one has time step of $h = 0.0005$yr. The trajectories are not stable in both cases and both will go to infinity eventually.

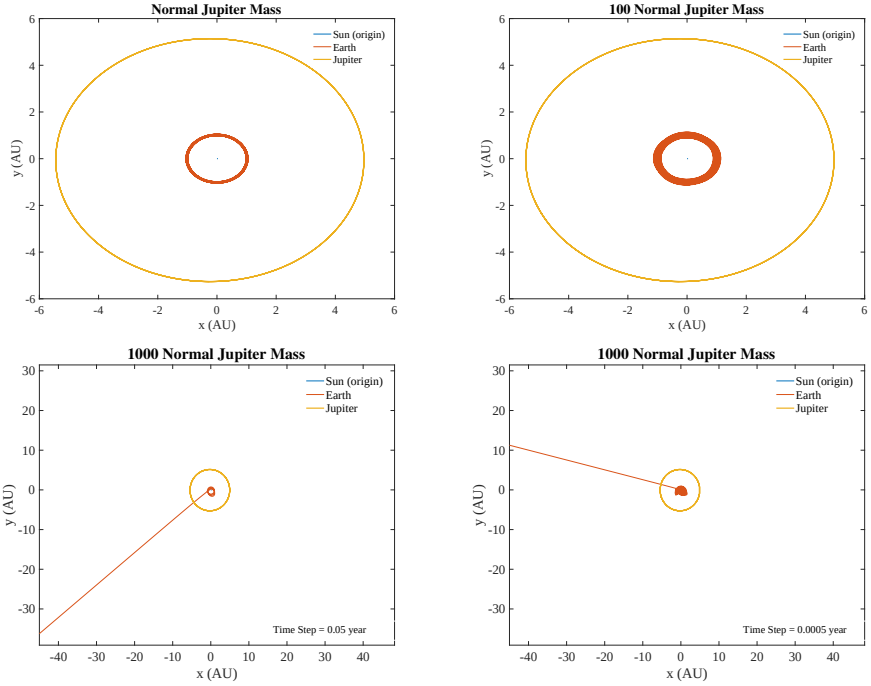

Figure 4: Stability of the Sun-Earth-Jupiter system using velocity Verlet method

3.4 Full solar system

Trajectories of all planets in the solar system are calculated with our Verlet solver. Center of mass for the solar system is fixed at the origin of our coordinate system. The initial conditions for all planets and the Sun are obtained from NASA[2]. Figure 5 shows the top view and the side view of the trajectories of all planets and Figure 6 shows the planets close to the Sun, which are the small trajectories in the center of Figure 5. From these figures, Pluto's trajectory shows many differences from other planets. Side view shows all other planets are almost in the same plane, while Pluto's orbit has a large angle relative to that plane. The top view shows that Pluto's trajectory intersects Neptune's so that in some positions, Pluto is closer to the Sun than Neptune.

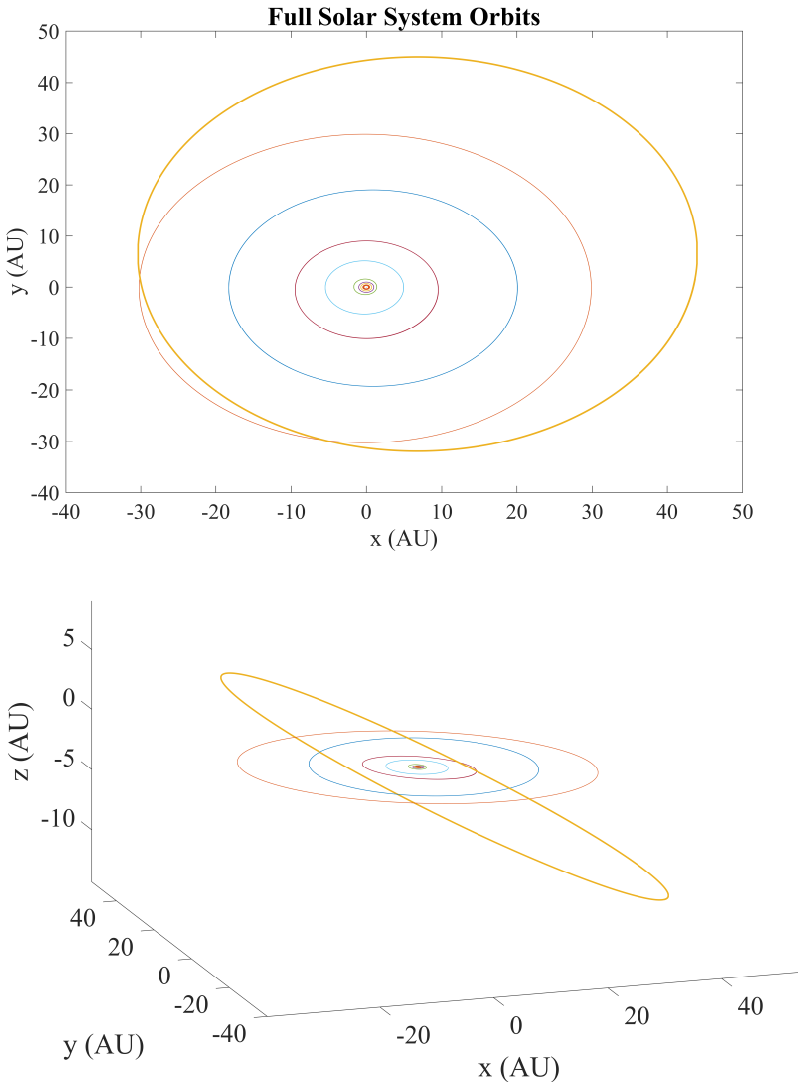

Figure 5: Trajectories of all planets in the system

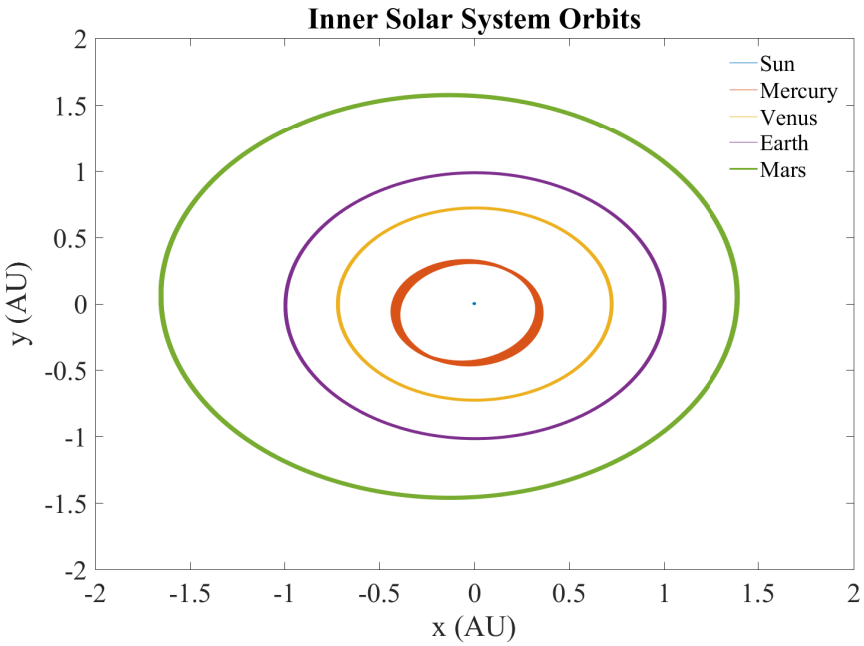

Figure 6: Trajectories of planets close to the Sun

Also the total kinetic and potential energy for the full solar system are calculated and shown in Figure 7. Both the kinetic and potential energy show oscillation. This is because the trajectories are all elliptical so the gravitational force is periodically oscillating as the planets complete their orbits. The period of the oscillation is about 4.2 years and corresponds to the orbital period of Jupiter. Jupiter has the largest mass of all the planets so its contribution to the oscillation of kinetic and potential energies dominates all contributions from other planets. Notice that each maximum in the kinetic energy oscillation corresponds to a minimum in the potential energy which is expected since energy is transferred back and forth between kinetic and potential. The flat line in the middle shows that the total energy for the whole system is conserved which must be true as there are no external forces acting on this system.

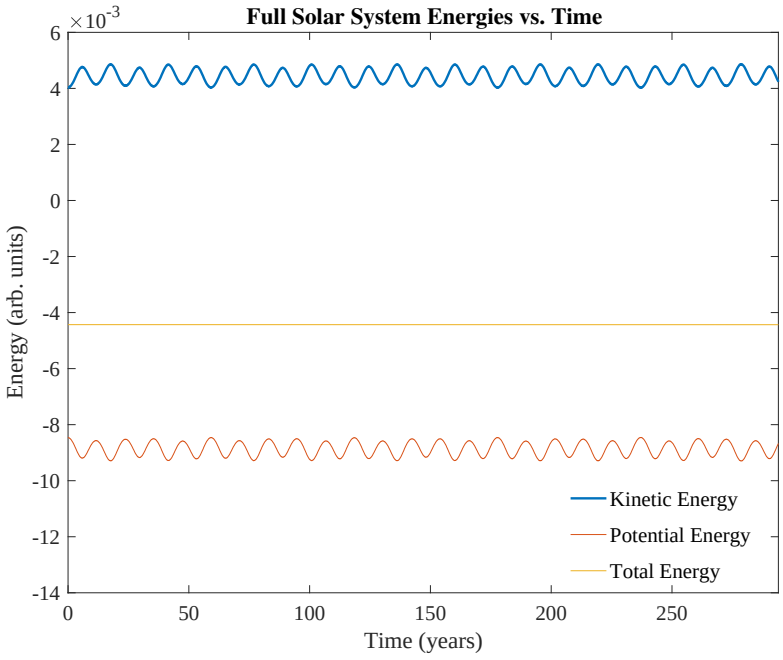

Figure 7: Kinetic and potential energy for the full solar system

3.5 Perihelion precession of Mercury

We run simulations of Mercury's orbit over one century with and without relativistic correction to gravitational force and study the precession of the orbit. Since this relativistic correction is very small (order of 10^{-7} compared with the classical force) it is necessary to use a sufficient time resolution in the simulation. If the time step is not small enough, an apparent large precession will be found due to numerical errors causing instability of the orbit. Using a time step of 10^{-8} yr, we calculate that the perihelion changes by 43.854 arcsec in one century. This is consistent with the literature value, when all classical effects of change in the orbit due to the other planets are subtracted [3].

Time Step (year)	Non-Relativistic Precession (arcsec)	Relativistic Precession (arcsec)
2×10^{-7}	4.3160	44.807
1×10^{-7}	4.0372	44.495
1×10^{-8}	4.3612	43.854

Table 1: Precession angle (arc seconds) of Mercury over one century with and without including relativistic correction to gravitational force for various time steps.

We note that the calculated non-relativistic precession is not monotonously increasing or decreasing with decreasing time step and therefore it is most-likely only an apparent precession due to numerical errors. In order to ensure that the computed relativistic precession is not due to numerical errors, we also simulate Mercury's orbit with and without relativistic correction over 1000 years using 1 billion mesh points. The total precession for the relativistic case was 426.47 arcsec or an average of 42.6 arcsec per century. The total precession over one millenium for the non-relativistic case was 4.7538 arcsec which is similar to the precession result for one century. Since this apparent precession does not increase with time, we believe it to be due to numerical errors.

4 Conclusion

Two object-oriented codes based on Velocity Verlet method and Euler's method are developed to solve the ordinary differential equations describing planetary motion. Velocity Verlet method is found to be more stable than the Euler method while still maintaining the speed of calculation. Analysis of conservation of energy and angular momentum from both methods also show that the Verlet method is more accurate. Verlet method is then used to calculate the escape velocity for the solar system and found to be very close to the analytical results. The Velocity Verlet code is then adapted to calculate the Sun-Earth-Jupiter system. Mass of Jupiter can have a significant effect on the motion of the Earth. With large enough mass for Jupiter, the trajectory of the Earth becomes unstable. Trajectories for all planets in the solar system are calculated by considering the gravitational force between any two planets and the energy for the whole solar systems are found to be conserved. Distinct behaviors of Pluto are also found. A relativistic correction to gravitational force is added in order to explain the precession of Mercury. [1]

Supplementary Material

All programs and benchmarks calculations can be found in the GIT repository: `https://github.com/tgolubev/PHY905MSU/tree/master/Project3`

References

[1] Morten Hjorth-Jensen, Diagonalization and eigenvalue problems. https://compphysics.github.io/ComputationalPhysicsMSU/doc/pub/ode/pdf/ode-beamer.pdf, 2017.

[2] NASA. Initial condion for all planets in solar system from nasa. https://ssd.jpl.nasa.gov/horizons.cgi#top.

[3] Richard Fitzpatrick., Perihelion Precession of Mercury, https://farside.ph.utexas.edu/teaching/336k/Newton/node116.html.