Orbital Integration of Earth and Jupiter

Karl Riedewald 123377661

1 Introduction

This report examines various methods of solving differential equations numerically by computer programme to simulate the Earth and Jupiter orbiting the Sun. Specifically, the Euler method, Runge-Kutta 2nd order (RK2) and Runge-Kutta 4th order (RK4) methods are used and compared for their accuracy and stability. The orbit of earth around the Sun was first simulated with each of the aforementioned methods. Jupiter was then added to the system and its gravitational affect on the orbit of the earth examined.

2 Orbit of Earth Around the Sun

2.1 Mathematical Analysis

From Newton's universal law of gravitation, the gravitational force F_E acting on the Earth as it orbits the Sun is given by

$$F_E = -\frac{G\mathrm{M}_E\mathrm{M}_\odot}{r^2}\hat{\mathbf{r}} \tag{1}$$

where G is the universal gravitational constant, $\mathrm{M}_\odot$ the mass of the sun, M_E the mass of the earth, r the distance between the Earth and the Sun and $\hat{\mathbf{r}}$ the unit vector pointing from the Earth to the Sun. The acceleration a_E that the earth experiences at any time during its orbit around the Sun is then given from Newton's second law of motion by

$$a_E = \frac{F_E}{\mathrm{M}_E} = -\frac{G\mathrm{M}_\odot}{r^2}\hat{\mathbf{r}} = -\frac{G\mathrm{M}_\odot\vec{\boldsymbol{r}}}{r^3} \tag{2}$$

where $\vec{\boldsymbol{r}}$ is the vector pointing from the Earth to the Sun.

It is now assumed that the mass of the Sun is sufficiently large relative to that of the Earth such that its motion can be ignored. Setting the Sun's position to the centre of the Cartesian coordinate system, the x and y components of the Earth's position as it orbits the Sun can now be written as

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -\frac{G\mathrm{M}_\odot x}{r^3} \qquad \frac{\mathrm{d}^2y}{\mathrm{d}t^2} = -\frac{G\mathrm{M}_\odot y}{r^3} \tag{3, 4}$$

Now in order to solve these differential equations numerically, they need to be split into two separate first order differential equations. By setting the velocities in the x and y directions to

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v_x \qquad \frac{\mathrm{d}y}{\mathrm{d}t} = v_y \tag{5, 6}$$

equations 3 and 4 can be rewritten in therms of the acceleration a_x and a_y as

$$a_x = \frac{\mathrm{d}v_x}{\mathrm{d}t} = -\frac{G\mathrm{M}_\odot x}{r^3} \qquad a_y = \frac{\mathrm{d}v_y}{\mathrm{d}t} = -\frac{G\mathrm{M}_\odot y}{r^3} \tag{7, 8}$$

This set of four differential equations are now all of first order and can thus be solved using the numerical integration methods.

2.2 Euler Method

2.2.1 Implementation

The Euler method is a simple technique which the next value y_{i+1} of first order differential equations over step h of the form

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y), \quad y(0) = y_0 \tag{9}$$

using the following equation

$$y_{i+1} = y_i + f(x_i, y_i)h \tag{10}$$

In order to solve the above differential equations with C++ in a nice manner using the Euler method, a state vector was defined as an array which stores the Earth's x, y position in the first two indexes and the Earth's velocity components v_x, v_y in the last two indexes. A function 'Earth Orbit' was then defined which takes in the current state vector and returns the derivative of the state vector, namely the velocity v_x, v_y and acceleration a_x, a_y. (This function corresponds to the $f(x_i, y_i)$ function of equation 10). The acceleration is calculated from equation 7 and 8. The new velocities can simply be passed on from the old state vector due to the nature of equations 5 and 6. Below is the code of this function.

```cpp
typedef std::array<double, 4> State;

State EarthOrbit(State s)
{
        //calculate the magnitude of r to find acceleration later
        double r = std::sqrt(s[0] * s[0] + s[1] * s[1]);

        //calcualte the acceleration for the x and y components
        double a_x = (-G * M_Sun * s[0] ) / (r * r * r);
        double a_y = (-G * M_Sun * s[1] ) / (r * r * r);

        //return the derivative state using the velocities of the input state vector
        return {s[2], s[3], a_x, a_y};
}
```

The function 'EulerStep' was then defined which takes in a function (the 'EarthOrbit' function in this case) and simply implements equation 10 as show here

```cpp
State EulerStep(State (*f)(State s), State& s, double dt)
{
        return s + f(s) * dt;
}
```

This function with a specified step size dt was then run in a loop over a certain time frame to numerically solve the required differential equations using the Euler method.

2.2.2 Results

The Euler integration method was run for the Earth around the Sun for a simulated time of 5 years for three different step sizes of a day, a half day and a quarter of a day as shown in figure 1. The initial x position was set to 1 Astronomical Unit (AU) which is defined as the mean distance between the Sun and the Earth while the initial y position was set to 0. The initial velocity of the Earth was set to the Keplerian velocity in the positive y direction given by

$$v_{y0} = \sqrt{\frac{GM_E}{r}} \tag{11}$$

The initial velocity in the x direction v_{x_0} was set to zero.

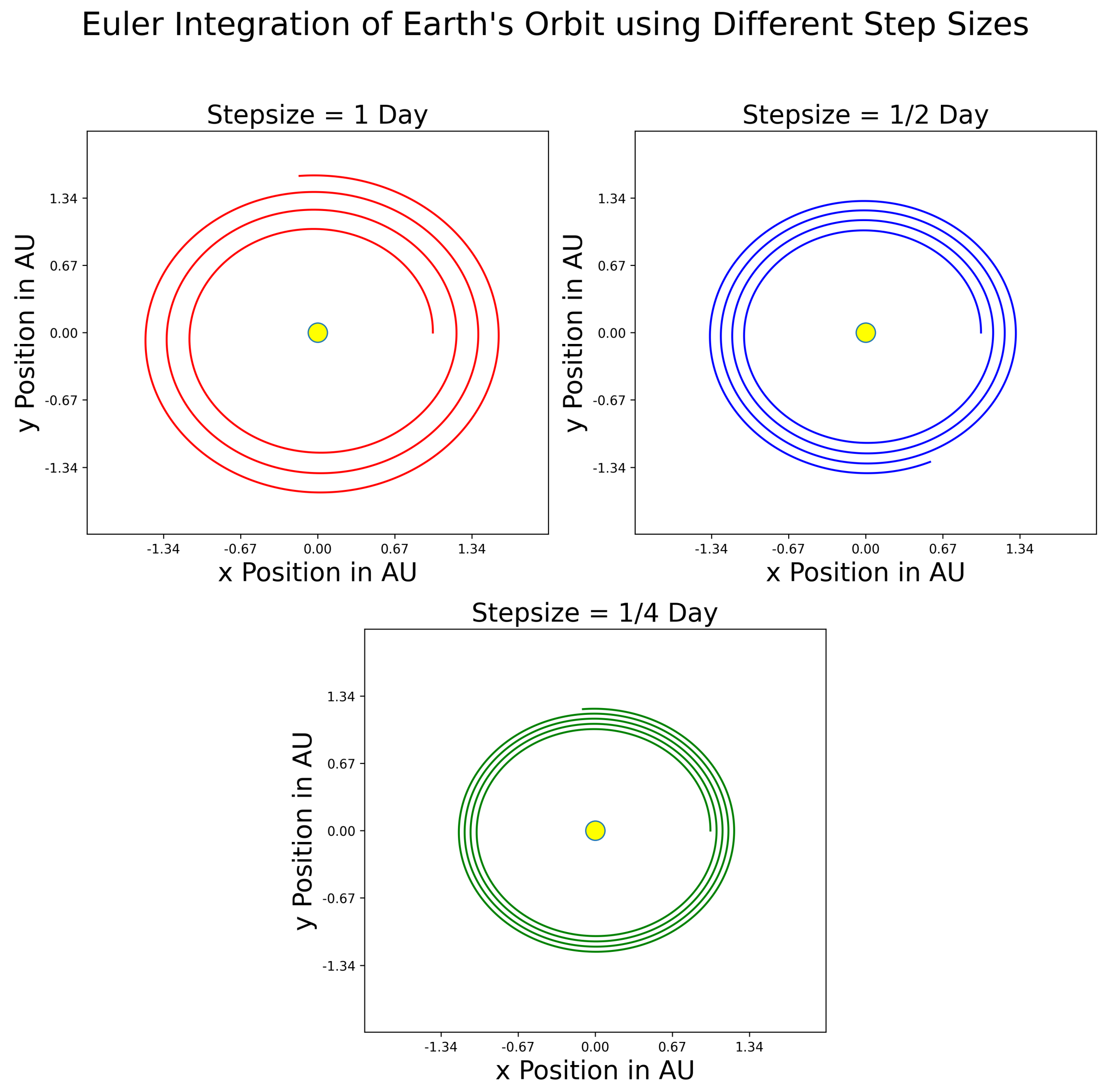

Figure 1: The Euler integration method performed for 3 different step sizes over a simulated time of 5 years. The Earth's initial position was set to 1 AU to the right of Sun which is represented by the yellow circle in each plot. The initial velocity was set the the Keplerian velocity

As can be seen from the plots of figure 1, the Euler method is not very stable. The radius of Earth's orbit increases dramatically over the 5 year simulation time. While decreasing the step size does decrease the rate at which the radius of the orbit increases, the orbit is still unstable even for a step size as low as a quarter of a day.

In reality, in the case where the earth is the only planet orbiting the Sun and assuming the Sun is fixed in place at the origin, the magnitude of the angular momentum of the earth and the total energy of the system would be conserved.

The total energy of the system is just the total energy of the Earth as the position of the Sun is assumed to be fixed. The total energy of the earth is then given by the its total kinetic and potential energy. The potential U_E and kinetic K_E energy of the earth are given by

$$U_E = -\frac{G\mathrm{M}_\odot \mathrm{M}_E}{r} \qquad\qquad K_E = \frac{1}{2}\mathrm{M}_E v^2 \tag{12}$$

Thus the total energy E of the earth at a certain point can be calculated as

$$E = \frac{1}{2}\mathrm{M}_E v^2 - \frac{G\mathrm{M}_\odot \mathrm{M}_E}{r} \tag{13}$$

The angular momentum L of the Earth is given by

$$\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p} \tag{14}$$

Where p is the instantaneous linear momentum of the Earth given by

$$\boldsymbol{p} = \boldsymbol{v}\mathrm{M}_{\boldsymbol{E}}$$

where v is the velocity of the Earth. Calculations of equations 13 and 14 were performed and saved after each Euler step with the plotted results shown in figure 2

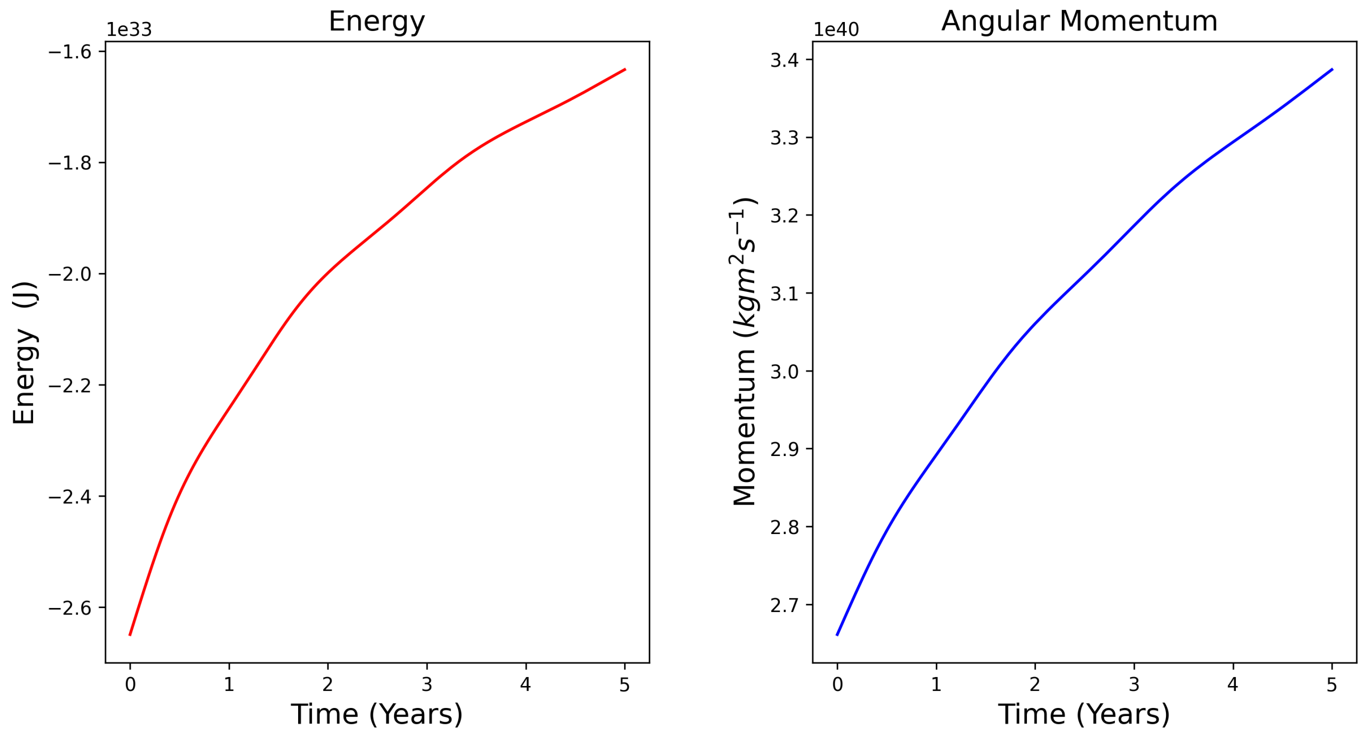

Figure 2: Plots of the total energy and angular momentum of the system with time for the Euler method over a simulated time of 5 years with a step size of a day.

Both the total energy and angular momentum of the system are seen to be rising quite rapidly which again highlights the instability and inaccuracy of the Euler method.

2.3 Runge-Kutta Methods

The Runge-Kutta methods are a set of more sophisticated numerical technique compared with the Euler method. The Runge-Kutta 2nd order (RK2) and 4th order (RK4) methods will be discussed here.

2.3.1 RK2 (Ralston's method)

The Ralston RK2 method is given by the following equation to solve differential equations again of the form given by equation 9

$$y_{i+1} = y_i + \left(\frac{1}{3}k_1 + \frac{2}{3}k_2\right)h \tag{15}$$

where k_1 and k_2 are given by

$$k_1 = f(x_i, y_i) \qquad k_2 = f\left(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1 h\right)$$

To implement this using state vectors, the RK2 step function becomes

```
State RK2StepRalston(State (*f)(State s), State& s, double dt)
{
        State k1 = f(s);
        State k2 = f(s + 0.25 * k1 * dt);

        //perform rk2 step
        return s + ((1.0/3.0) * k1 + (2.0/3.0) * k2) * dt;
}
```

with the 'EarthOrbit' function from before passed in.

2.3.2 RK4 Method

Similarly the RK4 method is given by

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \tag{16}$$

where

$$\begin{aligned}
k_1 &= f(x_i, y_i) \\
k_2 &= f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \\
k_3 &= f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h\right) \\
k_4 &= f(x_i + h, y_i + k_3 h)
\end{aligned}$$

The Rk4 step function is then written as

```
State RK4Step(State (*f)(State s), State& s, double dt)
{
        State k1 = f(s);
        State k2 = f(s + 0.5 * k1 * dt);
        State k3 = f(s + 0.5 * k2 * dt);
        State k4 = f(s + k3 * dt);

        //perform rk4 step
        return s + (1.0 / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4) * dt;
}
```

2.3.3 Results

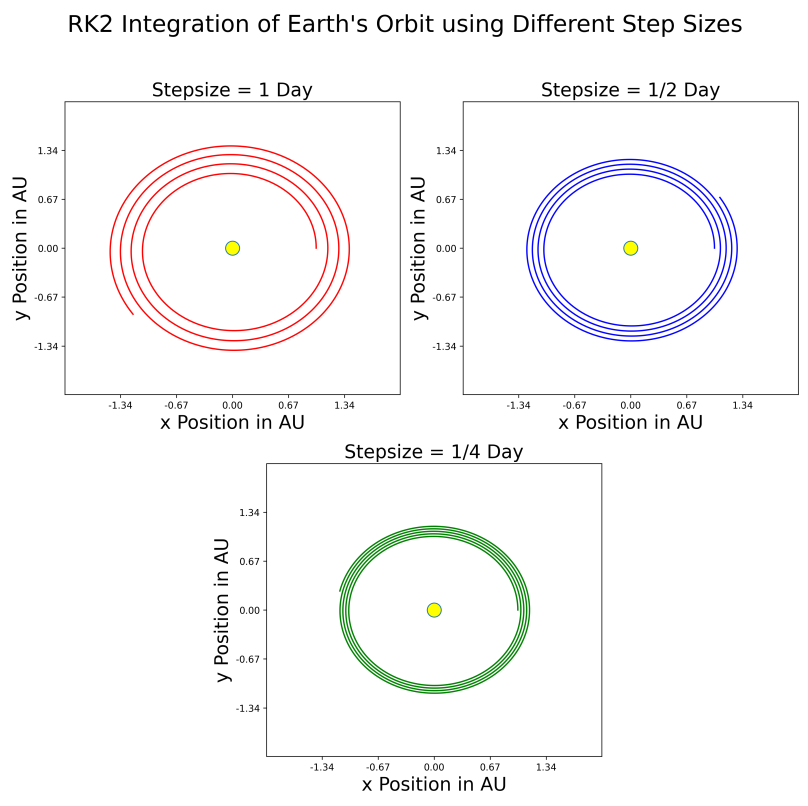

Figure 3

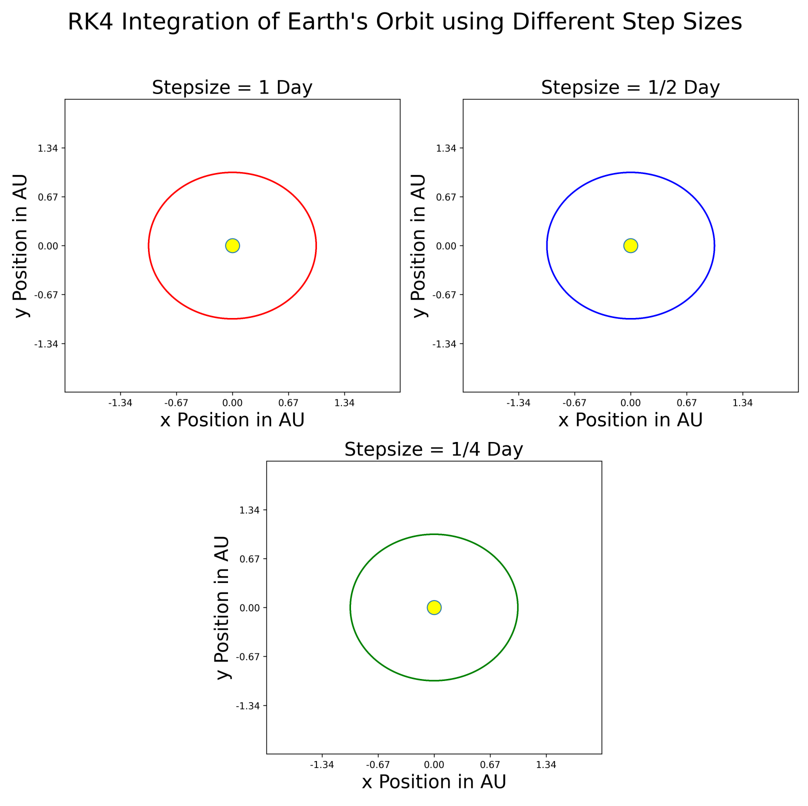

Figure 4

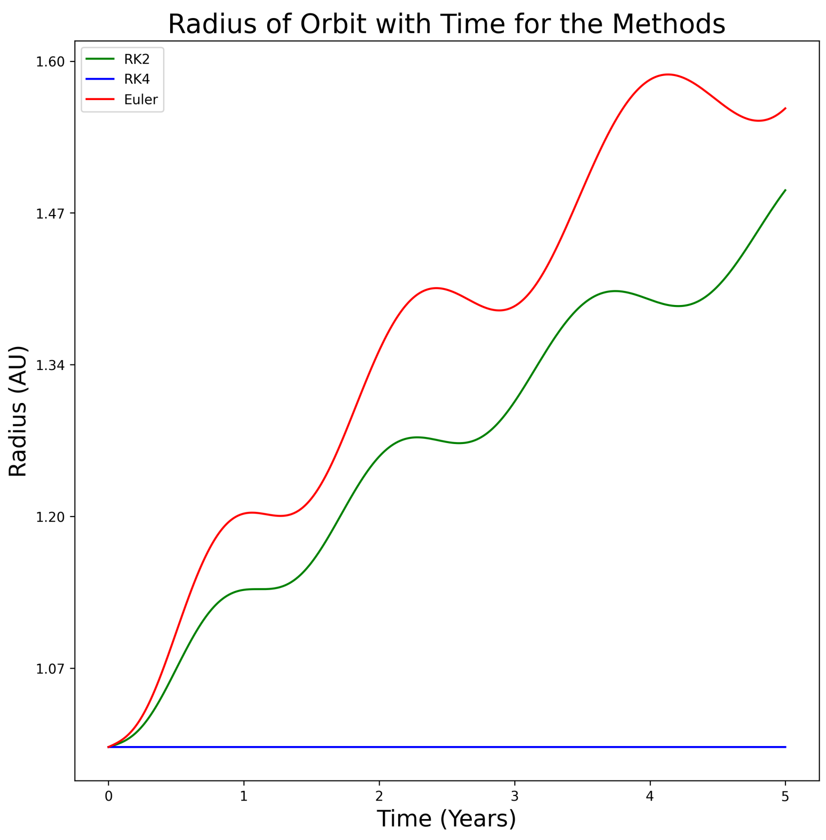

Figure 5

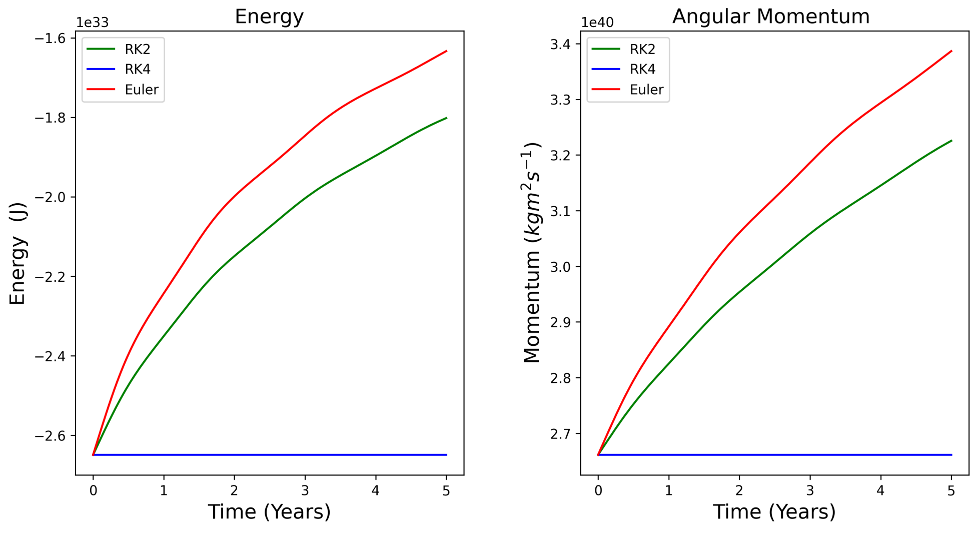

Figure 6

3 Orbit of Earth and Jupiter