Abstract

This project involves modeling the classic three-body problem using Python. Although the mathematical formulation of the three-body problem is relatively simple, finding an analytical solution to the problem is notoriously difficult due to the chaos of the dynamic system. The idea of this project is to use numerical methods to solve first-order differential equations and 3D plotting to simulate the motion trajectories of three celestial bodies under the influence of gravity.

Modeling the Three-Body Problem Using Python

Jun Wu

May 20, 2024

1 Introduction

The three-body problem has intrigued mathematicians, physicists, and astronomers for centuries. Despite its seemingly simple mathematical basis, finding an analytical solution has proven to be a difficult challenge, largely due to the chaotic nature of the dynamic systems involved. In this project, we leverage the power of Python and numerical methods to delve into the field of celestial mechanics.

2 Model

The mathematical statement of the three-body problem can be given in terms of the Newtonian equations of motion:

$$\ddot{r}_1 = -Gm_2\frac{r_1 - r_2}{|r_1 - r_2|^3} - Gm_3\frac{r_1 - r_3}{|r_1 - r_3|^3} \tag{1}$$

$$\ddot{r}_2 = -Gm_3\frac{r_2 - r_3}{|r_2 - r_3|^3} - Gm_1\frac{r_2 - r_1}{|r_2 - r_1|^3} \tag{2}$$

$$\ddot{r}_3 = -Gm_1\frac{r_3 - r_1}{|r_3 - r_1|^3} - Gm_2\frac{r_3 - r_2}{|r_3 - r_2|^3} \tag{3}$$

Where r is the vector positions and G is the gravitational constant:

$$\ddot{r}_i = (x_i, y_i, z_i) \tag{4}$$

Functions reduced to first-order differential form:

$$\frac{dv_i}{dt} = Gm_j\frac{r_j - r_i}{|r_j - r_i|^3} + Gm_k\frac{r_k - r_i}{|r_k - r_i|^3} \tag{5}$$

$$\frac{dr_i}{dt} = v_i \tag{6}$$

3 Method

3.1 Define Initial Conditions

```
#Earth–Moon–Sun Partmeters
G = 6.67*10**(-11)
m1 = 1.989*10**30 #sun
m2 = 5.972*10**24 #earth
m3 = 7.348*10**22 #moon

# Initial positions for E–M–S
r1 = [0,0,0]
r2 = [149.6*10**9,0,0]
r3 = [1.49984*10**11,0,0]

# Initial velocity for E–M–S
v1 = [0,0,0]
v2 = [0,30000,0]
v3 = [0,33000,0]
```

Here, the initial condition is configured to a well-known stable solution to the three-body problem, G denotes the gravitational constant, and m1, m2, and m3 denote masses of the Sun, Earth, and Moon (in kg). For simplicity, the position of the Sun is set to be the origin of the 3D space, with the initial velocity set to zero. We will explore various initial conditions in this project to understand how the initial conditions affect the stability of the three-body problem.

3.2 Define Function for Simulation

```
def ThreeBody(state, t, param):
    G = param[0]
    m1 = param[1]
    m2 = param[2]
    m3 = param[3]
    r1 = state[:3]
    r2 = state[3:6]
    r3 = state[6:9]
    v1 = state[9:12]
    v2 = state[12:15]
    v3 = state[15:18]

    #Distance bewteen two stars
    r12 = np.linalg.norm(r2-r1)
    r13 = np.linalg.norm(r3-r1)
    r23 = np.linalg.norm(r3-r2)

    #First-order differential equations
    dr1 = v1
    dr2 = v2
    dr3 = v3
    dv1 = G * m2 * (r2 - r1) / r12 ** 3 + G * m3 * (r3 - r1) / r13 ** 3
    dv2 = G * m1 * (r1 - r2) / r12 ** 3 + G * m3 * (r3 - r2) / r23 ** 3
    dv3 = G * m1 * (r1 - r3) / r13 ** 3 + G * m2 * (r2 - r3) / r23 ** 3

    #Combine arrays into one
    dr = np.concatenate((dr1,dr2,dr3))
    dv = np.concatenate((dv1,dv2,dv3))
    derivs = np.concatenate((dr,dv))

    return derivs
```

The ThreeBody function takes in the state vector, time, and parameters as input and returns the derivatives of the state vector with respect to time. It will be passed into a numerical solver along with initial conditions and parameters to obtain the trajectories of the three bodies over time.

3.3 Apply Runge-Kutta Integrator (4th Order)

```
def rk4(x,t,tau,derivsRK,param):
    """
        Input arguments:
         x = current value of dependent variable
         t = independent variable (usually time)
         tau = step size (usually timestep)
         derivsRK = right hand side of the ODE; derivsRK is the
                    name of the function which returns dx/dt
                    Calling format derivsRK (x,t,param).
         param = extra parameters passed to derivsRK
        Output arguments:
         xout = new value of x after a step of size tau
    """
    half_tau = 0.5*tau
    F1 = derivsRK(x,t,param)
    t_half = t + half_tau
    xtemp = x + half_tau*F1
    F2 = derivsRK(xtemp,t_half,param)
    xtemp = x + half_tau*F2
    F3 = derivsRK(xtemp,t_half,param)
    t_full = t + tau
    xtemp = x + tau*F3
    F4 = derivsRK(xtemp,t_full,param)
    xout = x + tau/6.*(F1 + F4 + 2.*(F2+F3))
    return xout

state = rk4(state,t,tau,ThreeBody,param)
```

This part of the code was imported from Lab 6. It will be used to integrate equations (5) and (6), which will take the inputs of initial values and the ThreeBody function, then output the new positions and velocity after the chosen step size. All positions will be stored in an array which can be used to plot the 3d trajectories of the three stars.

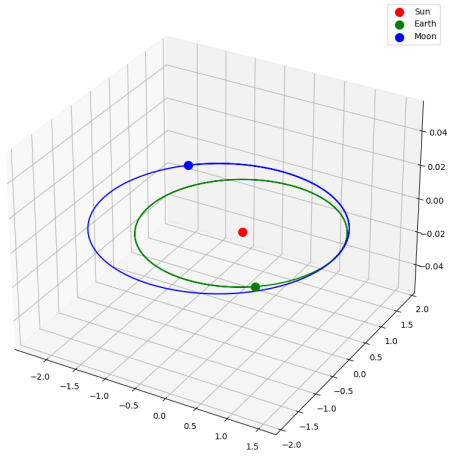

Figure 1: Orbits of Earth, Moon, and Sun

4 Results

In this section, I examine the results obtained in simulation scenarios with different initial conditions, both stable and chaotic results. As shown in Figure 1, the visualization depicts three celestial bodies in a closed orbit configuration, with the central red dot representing the Sun, the green dot representing the Earth, and the blue dot representing the Moon. While the observed results are largely in line with expectations, it is worth noting that there are slight biases as the simulations rely exclusively on classical mechanics.

Figure 2 shows a scenario characterized by chaotic behavior, with the gravitational constant set to 0.2 and the stellar mass normalized to 1 to reduce the consumption of computing power by large numbers. The image reflects the clear influence of gravity on the stars' trajectories, especially when the two stars are close together.

Subsequently, Figure 3 retains most of the parameter settings of Figure 2, but adjusts the initial velocity of the celestial body to try to stabilize the trajectories. This modification introduces brief but predictable movements into the trajectories. Notably, two stars exhibit a spiral trajectory, while a third star passes between them.

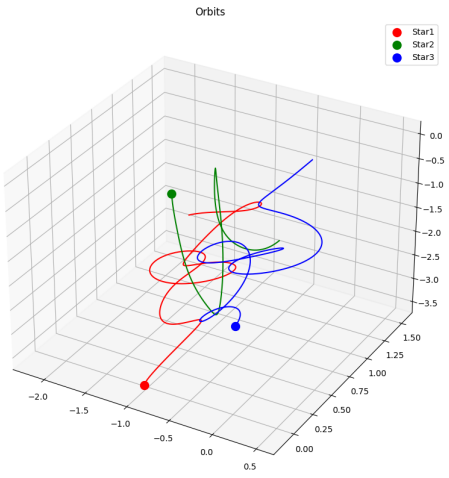

Figure 2: Chaotic Orbits

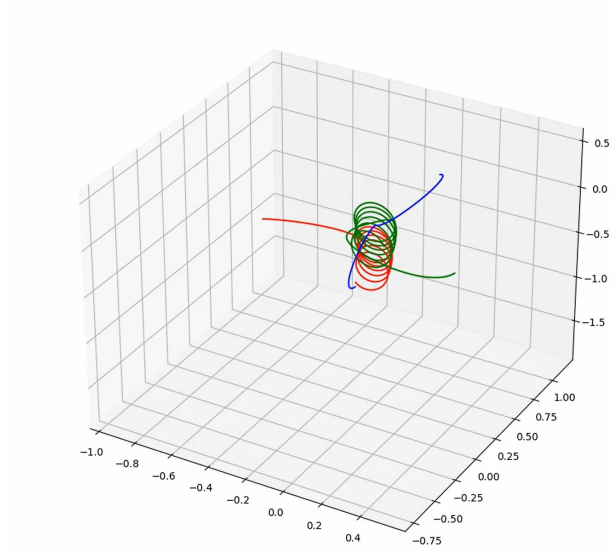

Figure 3: Stable for a short period

5 Conclusions

The three-body problem remains a difficult mathematical and physical problem to solve, and until a general closed-form solution is found, we can only rely on numerical approaches. Unfortunately, this model based on classical mechanics is very sensitive to initial conditions, and slight changes will lead to complete changes in the motion trajectory, which makes it difficult to find a long-term stable solution. Yet, amid the complexities and uncertainties, there remains an enduring sense of fascination and wonder. The pursuit of solutions to the three-body problem serves not only as an intellectual endeavor but also as a testament to humanity's insatiable curiosity and relentless pursuit of understanding. As we continue to grapple with these challenges, we are reminded of the boundless potential of human ingenuity and the inexhaustible mysteries that await our exploration.

References

[1] Wikipedia contributors. (2024, May 13). Three-body problem. Wikipedia. https://en.wikipedia.org/wiki/Three-body$_problem$

[2] Heggie, D. (2006). Gravitational N-Body Problem (Classical). In Elsevier eBooks (pp. 575–582).

[3] KAO, Z. C. (2011). CLASSICAL MECHANICS: THE THREE-BODY PROBLEM. In CLASSICAL MECHANICS: THE THREE-BODY PROBLEM. https://www.math.uchicago.edu/ may/VIGRE/VIGRE2011/REUPapers/KaoZ.pdf