Modeling Dynamics of Euler's Three Body Problem

Jasper Swallen

Department of Physics and Astronomy,

University of Southern California, Los Angeles, California 90089, USA

(Dated: December 16, 2022)

Abstract

Euler's Three Body Problem is a simplification of the general 3-Body Problem, with two fixed large masses and one free mass. While it is possible to solve Euler's Three Body Problem analytically, the solutions are not simple to evaluate in cartesian coordinates. Instead, various numerical methods were used. Euler's method, the Runge-Kutta method, and Forest & Neri's method were used. With these methods, plots were generated in Python based on various initial conditions. Forest & Neri's method was found to be the most precise, followed by Runge-Kutta and then Euler's Method. Euler's Method had the fastest runtime, followed by Forest & Neri and then the Runge-Kutta methods.

CONTENTS

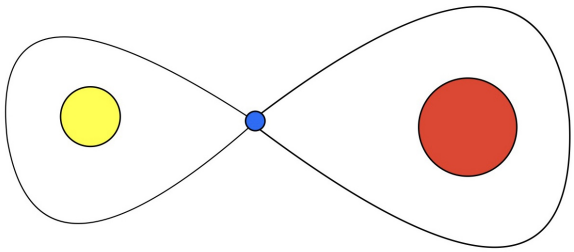

FIG. 1: A model solar system with simplified orbits

I. BACKGROUND AND APPLICATIONS

Euler's Three Body Problem is a simplification of the 3-body problem (which is itself a special case of the N-body problem). The general N-body problem is to describe the movement of N masses which exert an attractive gravitational force on each other. In Euler's simplification, two large bodies are fixed at points in space (and are assumed to not act upon each other), while one smaller body is free to move through space. This smaller body's movement is dependent only on its relative position to the two larger masses and their masses. It can be assumed that the smaller mass m is much smaller than the two larger masses m_2 and m_3. This can therefore be represented by a one-body problem. Given the level of simplification, it would not seem that this system is a close representation of real-world problems. However, this is not the case.

There are two main applications of this problem. In our solar system, this is a very close representation of the movement of comets which interact with Jupiter and the sun [1, 2]. The other application is in an electric field. Although electric fields are not described by exactly the same forces, their potential is still inverse to the distance. Thus, an analogue would be an electron moving through the electric field of two nuclei.

To solve N-body problems, the gravitational forces must be known. Newton's Law of Gravitation states that, for any two masses m_1 and m_2 separated by a radius $r = |\boldsymbol{r_1} - \boldsymbol{r_2}|$ (where $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are the positions of m_1 and m_2, respectively), the gravitational force of m_2 on m_1 is

$$\boldsymbol{F_{12}} = -\frac{Gm_1m_2}{|\boldsymbol{r_1} - \boldsymbol{r_2}|^3} * (\boldsymbol{r_1} - \boldsymbol{r_2}) \tag{1}$$

and the potential energy is

$$U(\boldsymbol{r_1}, \boldsymbol{r_2}) = -\frac{Gm_1m_2}{|\boldsymbol{r_1} - \boldsymbol{r_2}|} \tag{2}$$

Since gravitation forces can be superimposed, the force of every body on a given body in an N-body system can be calculated as

$$\boldsymbol{F_i} = \sum_{j=1, j\neq i}^{N} -\frac{Gm_i m_j}{|\boldsymbol{r_i} - \boldsymbol{r_j}|} * (\boldsymbol{r_i} - \boldsymbol{r_j}) \tag{3}$$

Additionally, this is a closed system, so the total energy of the system is conserved.

II. GENERAL EQUATIONS OF MOTION

In 3-dimensional space, there are three degrees of freedom because only one mass is moving. If the three masses initially lie in a plane together, they will remain in this plane because there will be no forces normal to the plane. This results in two degrees of freedom, so for simplicity, the initial conditions will be chosen for such a case ($z \equiv 0$).

Since the total energy of the system is preserved, and the potential of the system can be described as above, the following equations of energy hold

$$\begin{aligned} T &= \frac{1}{2}mv^2 = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) \\ U_i &= -\frac{Gmm_i}{|\boldsymbol{r_i} - \boldsymbol{r}|} \\ E &= \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) - \frac{Gmm_1}{|\boldsymbol{r_1} - \boldsymbol{r}|} - \frac{Gmm_2}{|\boldsymbol{r_2} - \boldsymbol{r}|} \end{aligned} \tag{4}$$

While solving a Lagrangian or Hamiltonian with these equations in cartesian coordinates is possible, simplifications arise when the system is converted to an elliptical coordinate system (ξ, η) (see Fig. 2). If the masses are positioned in this coordinate system at the foci $2f$ apart, which can be done by rotating and shifting, this is translatable to $x = f\cosh\xi\cos\eta$ and $y = f\sinh\xi\sin\eta$.

Therefore, there are two conserved quantities in the system: Whittaker's constant (see [3, p. 100]) and total energy. Whittaker's constant can be represented as

$$\begin{aligned} w &= \boldsymbol{L_1} \cdot \boldsymbol{L_2} + 2mf(-\mu_1\cos\theta_1 + \mu_2\cos\theta_2) \\ &= m^2 r_1^2 r_2^2 \dot{\theta_1}^2 \dot{\theta_2}^2 + 2mf(-\mu_1\cos\theta_1 + \mu_2\cos\theta_2) \\ &= -2mf^2E - \alpha \end{aligned} \tag{5}$$

where r_i is the distance $|\boldsymbol{r_i} - \boldsymbol{r}|$ between m_i and m; $\boldsymbol{L_i} = mr_i^2\dot{\theta}_i\hat{z}$ is the angular momentum of a mass; θ_i is the angle between $\boldsymbol{r_i}$ and the x-axis; $\mu_i = Gmm_i$; and α is the "separation constant"

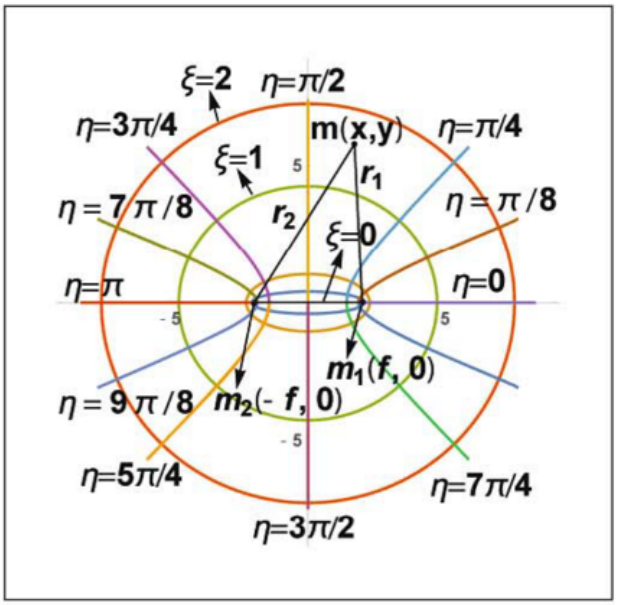

FIG. 2: the elliptical coordinate system (ξ, η). Relative to polar coordinates, ξ can be thought of as correspondent to radial distance r, and η the angle θ. Adopted from [3]

.

derived from the separation of variables of E in elliptical coordinates [3, p. 101].

Since there are also just two degrees of freedom in this case, the system is exactly analytically solvable by integration in the Liouville sense. See Ó'Mathúna [4, pages 49-105 and 113-142] for a detailed solution.

III. NUMERICAL SOLUTION

A. Finding Equations

While analytical solutions are useful for determining the exact motion of systems, numerical solutions are often much simpler. Such is the case here. To do this, without loss of generality, shift the system so that m_1 is at the origin and m_2 is a distance d along the y-axis from the origin. Now represent m_2 as a factor of $m_a \equiv m_1$ such that $m_2 = \alpha m_a$, as shown in Fig. 3.

Up until this point, the masses and distances have been unitless and arbitrary. If these masses are instead measured in terms of solar units, further simplifications arise. If m_a is set to a "solar mass", we see that $Gm_a = 4\pi^2$. This is because the angular velocity of a mass around a primary (for example, the Earth around the Sun) is $2\pi * AU/yr$, where AU is the average distance between the primary and the mass and yr is the period (on Earth, one year). That is, the problem is nondimensionalized by setting the unit of mass to m_1 and the unit of time such that the period of

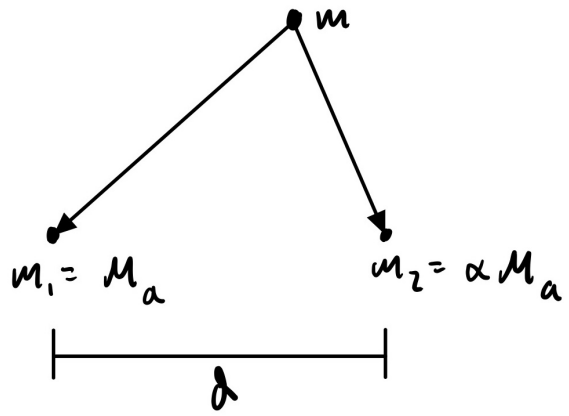

FIG. 3: the shifted system with m_1 at the origin

the orbits of the primaries is 2π [1, 5].

Since the origin is now fixed, the equations of motion now use relative positioning to the origin (and m_1). From this and Newton's Law of Gravitation,

$$\begin{aligned} F_x &= -\frac{Gm_a m}{r_1^3}x - \frac{G\alpha m_a m}{r_2^3}(x-d) \\ F_y &= -\frac{Gm_a m}{r_1^3}y - \frac{G\alpha m_a m}{r_2^3}y \end{aligned} \tag{6}$$

with $r_i = |\boldsymbol{r_i} - \boldsymbol{r}|$ the distance between m and m_i. In cartesian coordinates, this results in

$$\begin{aligned} \boldsymbol{r_1} &= x\hat{x} + y\hat{y} \\ \boldsymbol{r_2} &= (x-d)\hat{x} + y\hat{y} \end{aligned} \tag{7}$$

so

$$\begin{aligned} r_1 &= \sqrt{x^2+y^2} \\ r_2 &= \sqrt{(x-d)^2+y^2} \end{aligned} \tag{8}$$

Plugging in $Gm_a = 4\pi^2$ and dividing 6 by m the following equations of acceleration emerge, dependent only on x and y positions

$$\begin{aligned} \ddot{x}(x,y) &= -4\pi^2\left(\frac{x}{(x^2+y^2)^{\frac{3}{2}}} + \frac{\alpha(x-d)}{((x-d)^2+y^2)^{\frac{3}{2}}}\right) \\ \ddot{y}(x,y) &= -4\pi^2 y\left(\frac{1}{(x^2+y^2)^{\frac{3}{2}}} + \frac{\alpha}{((x-d)^2+y^2)^{\frac{3}{2}}}\right) \end{aligned} \tag{9}$$

This is a system of nonlinear differential equations. As demonstrated above, analytical solutions are possible, but messy.

B. Plottable Solutions

Instead of analytically solving this system, numerical integrators can be used. Here, Euler's Method, the Runge-Kutta 4$^{\text{th}}$ Order Method, and the Forest & Neri method will be demonstrated. There are many other methods that allow for approximate solutions of systems of coupled differential equations, such as the leapfrog method.

However, these particular methods have been chosen for several reasons. Euler's Method was chosen for its simplicity, speed, and ease of use with arrays (it accesses each element one-by-one). The Runge-Kutta 4$^{\text{th}}$ Order Method was chosen because it is commonly used in general iterative differential equations solutions. The Forest & Neri method was chosen because it was specifically written for use with force equations and computer modeling.

1. Euler's Method

Euler's method is a simple and efficient formula to iteratively solve differential equations. Using Euler's approximation

$$x(t+\delta t) = x(t) + \delta t\dot{x}(t) + O(\delta t^2) \tag{10}$$

and neglecting higher order terms $O(\delta t^2)$, the solution is quickly plottable in cartesian coordinates. The error (by which I mean local truncation error, or the potential error from each step) is on the order of δt [5, p. 298]. Adapting 10 for x, y, $\dot{x}$, and $\dot{y}$, we get the following system of equations.

$$\begin{aligned} x_{n+1} &= x_n + \dot{x}_n\delta t \\ y_{n+1} &= y_n + \dot{y}_n\delta t \\ \dot{x}_{n+1} &= \dot{x}_n + \delta t\ddot{x}(x_{n+1}, y_{n+1}) \\ \dot{y}_{n+1} &= \dot{y}_n + \delta t\ddot{y}(x_{n+1}, y_{n+1}) \end{aligned} \tag{11}$$

When they are evaluated in order, this is an iterable, programmable, and plottable solution.

2. Runge-Kutta Method

Another way to incrementally solve integrable solutions is the Runge-Kutta method (specifically, the Runge-Kutta 4^{th} Order Method) [6]. With the same logic of iteration as the Euler Method above, Runge-Kutta allows for more precision by using intermediate steps to calculate the $n+1$ position. Runge-Kutta is much more precise than Euler's method, with an error of $O(\delta t^5)$. The general Runge-Kutta 4^{th} order method is [6, p. 21]:

$$\begin{aligned} x_{n+1} &= x_n + \frac{1}{6}\delta t(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= f(x_n) \\ k_2 &= f(x_n + \delta t\frac{k_1}{2}) \\ k_3 &= f(x_n + \delta t\frac{k_2}{2}) \\ k_4 &= f(x_n + \delta t k_3) \end{aligned} \tag{12}$$

Since both position and velocity can be calculated in this manner, in the context of this 3-Body problem, the equations become (evaluated in this order)

$$\begin{aligned} x_1 &= \delta t\dot{x}_n & \dot{x}_1 &= \delta t\ddot{x}(x_n, y_n) \\ x_2 &= \delta t(\dot{x}_n + \frac{\dot{x}_1}{2}) & \dot{x}_2 &= \delta t\ddot{x}(x_n + \frac{x_1}{2}, y_n + \frac{y_1}{2}) \\ &... & &... \\ x_{n+1} &= x_n + \frac{x_1 + 2x_2 + 2x_3 + x_4}{6} & \dot{x}_{n+1} &= \dot{x}_n + \frac{\dot{x}_1 + 2\dot{x}_2 + 2\dot{x}_3 + \dot{x}_4}{6} \end{aligned} \qquad \text{(for } x \text{ and } y\text{)} \tag{13}$$

3. Forest & Neri Method

The Forest and Neri method is an alternative 4^{th} order integrator to the Runge-Kutta method that requires fewer calculations and is more precise [7, pp. 60-61]. While the Runge-Kutta method is applicable to any numerical solution of a system of equations, Forest and Neri's method is specifically adapted to computer solutions to modeling force equations. The equations are as follows:

$$\begin{aligned} &\beta = 2^{\frac{1}{3}}, c_4 = c_1 = \frac{1}{2(2-\beta)}, c_3 = c_2 = \frac{1-\beta}{2(2-\beta)} \\ &d_3 = d_1 = \frac{1}{2-\beta}, d_2 = \frac{-\beta}{2-\beta}, d_4 = 0 \end{aligned} \tag{14}$$

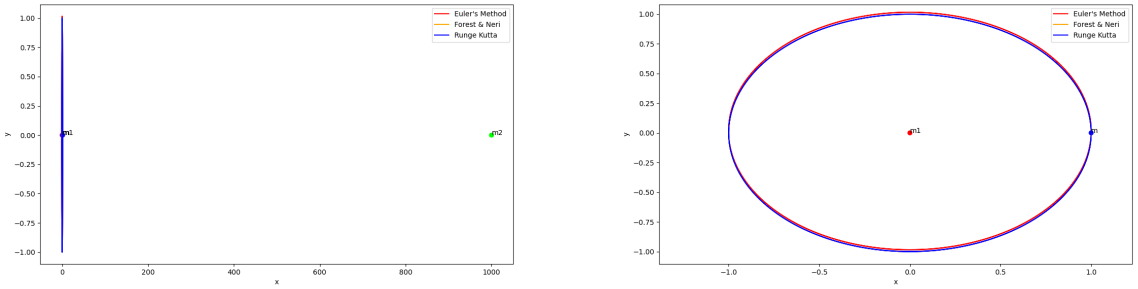

FIG. 4: With a large d, the system behaves as though there is only one star. **(Left)** The full system overview. **(Right)** The same system, zoomed in to the moving mass.

$$\begin{aligned} x_1 &= x_n + \delta t c_1 \dot{x}_n & \dot{x}_1 &= \dot{x}_n + \delta t d_1 \ddot{x}(x_1, y_1) \\ x_2 &= x_1 + \delta t c_2 \dot{x}_1 & \dot{x}_2 &= \dot{x}_1 + \delta t d_2 \ddot{x}(x_2, y_2) \\ &\cdots \quad \text{(for } x \text{ and } y\text{)} & &\cdots \\ x_{n+1} &= x_4 & \dot{x}_{n+1} &= \dot{x}_4 \end{aligned} \tag{15}$$

As with Runge-Kutta, when calculated in order, this is an iterative solution. The error is on the order of $O(\delta t^5)$.

IV. INTERPRETING SOLUTIONS

There are several cases in the limit where this solution demonstrates its clarity. The first case is when d is very large (and x is not) 4. In this case, the terms dependent on r_2 disappear. This is equivalent to a system with only one star with mass m_a. On the other hand, if d is very small, $r_1 \cong r_2$. This is also equivalent to a system with one star, but this time the star has a mass $(1+\alpha)m_a$.

From Fig. 5, we can see the difference between methods and iteration size. At large iteration size (higher error), the difference between the methods is far more pronounced. Indeed, the curves are noticeably not smooth. As the errors compound, the final positions between the methods are very different. Decreasing the iteration size by a factor of 2 results in much closer results. In fact, it is hard at first glance to differentiate between the Runge-Kutta and Forest & Neri methods – only after zooming in is the difference evident. The closeness between the methods increases even further when δt is decrease by another factor of 5. Here, even Euler's method is very similar to the 4^{th} order methods, although they are still closer to each other.

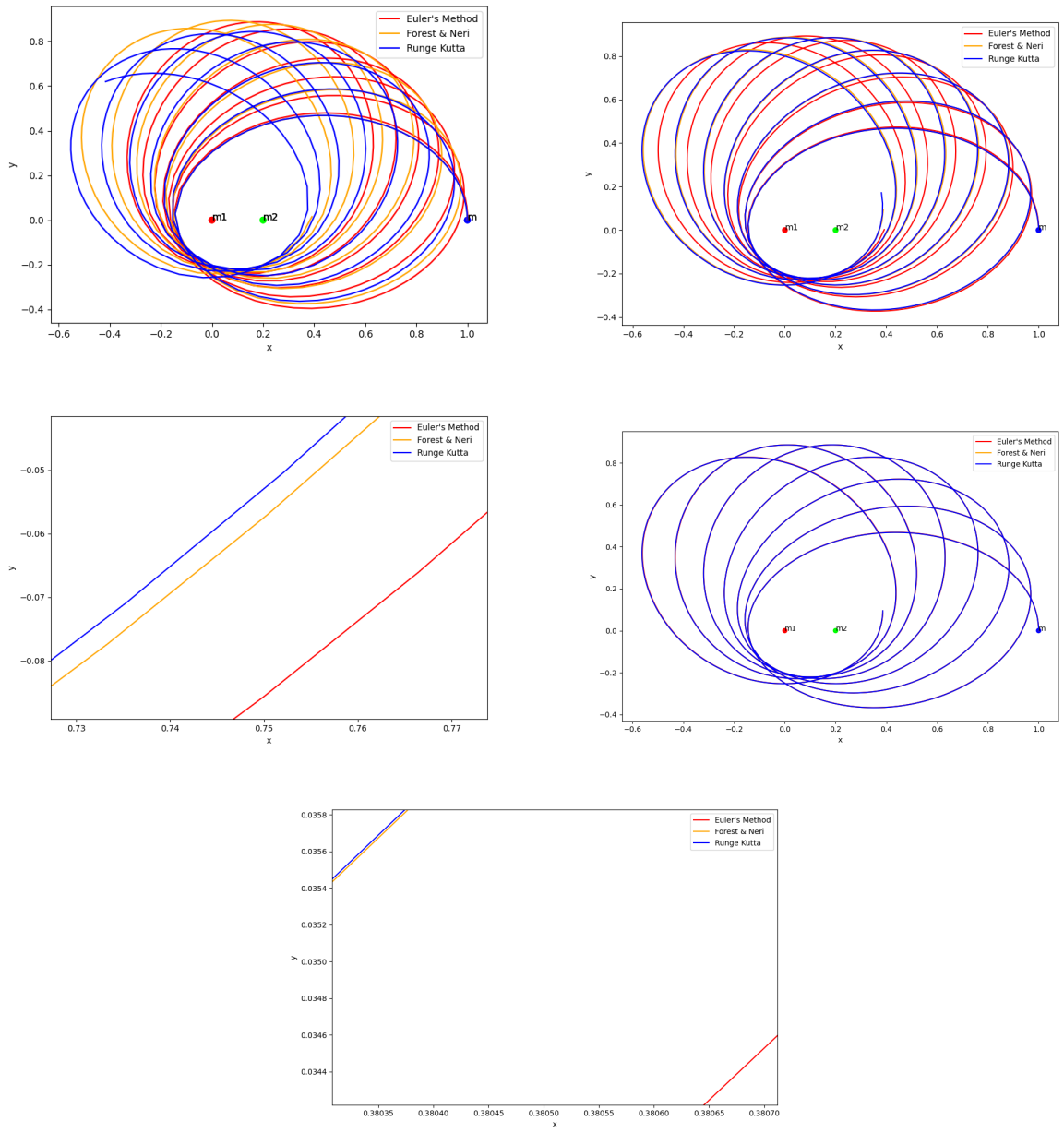

FIG. 5: All graphs are with the same initial conditions, but different iteration size. **(Top Left)** Graph at $\delta t = 0.005$. All three methods are noticeably separated. **(Top Right)** Graph at $\delta t = 0.0025$ **(Middle Left)** Zoomed in graph at $\delta t = 0.0025$ to show difference between Forest & Neri III B 3 and Runge-Kutta III B 2 methods. Euler's method III B 1 is still noticeably different. **(Middle Right)** Graph at $\delta t = 0.0005$ **(Bottom)** Zoomed in graph at $\delta t = 0.0005$. All three methods are very close at this δt.

Appendix A: Code

The code for this project is fully available at my public GitHub repository, containing both a Jupyter Notebook and command-line Python script. The requirements to run these are Python 3,

`matplotlib`, and `numpy` (as well as `ipywidgets` for the Jupyter Notebook). Then, run `$ python3 main.py` and input the initial positions and masses of all three objects, as well as the initial velocity of the moving object. Alternatively, use the `main.ipynb` Jupyter Notebook to input values from a browser or IDE. The Jupyter Notebook presentation of graphs is slightly different, as each method is in its own graph to show the evolution of the function with respect to time.

The following is `eulerMethod()`, which calculates and plots the graph of Euler's Three Body Problem with Euler's numerical approximation method. This code is found in `main.py` and `main.ipynb`.

```
def eulerMethod(moving_mass_x, moving_mass_y, moving_mass_x_prime,
                                        moving_mass_y_prime, its, delta):
    # declare arrays that will be appended to in loop
    x = [moving_mass_x]
    y = [moving_mass_y]
    x_prime = [moving_mass_x_prime]
    y_prime = [moving_mass_y_prime]

    for n in range(its):
        # calculate next x, y, x', and y'
        x_n_plus_one = x[n] + delta * x_prime[n]
        x.append(x_n_plus_one)
        y_n_plus_one = y[n] + delta * y_prime[n]
        y.append(y_n_plus_one)

        x_prime_n_plus_one = x_prime[n] + delta * \
            calculateXAccel(x[n + 1], y[n + 1])
        x_prime.append(x_prime_n_plus_one)

        y_prime_n_plus_one = y_prime[n] + delta * \
            calculateYAccel(x[n + 1], y[n + 1])
        y_prime.append(y_prime_n_plus_one)

    plt.plot(x, y, label="Euler's Method", color="red")
```

The Runge-Kutta and Forest & Neri methods work in a similar manner, although they require many more calculations per iteration.

Note: the Jupyter Notebook uses an additional recipe, colorline [8], to show evolution with

time.

[1] J. Worthington, A study of the planar circular restricted three body problem and the vanishing twist (2012).

[2] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, Equadiff 99 (World Scientific, 2000) Chap. Dynamical Systems, the Three-Body Problem and Space Mission Design, pp. 1167–1181.

[3] G. S. Krishnaswami and H. Senapati, *An Introduction to the Classical Three-Body Problem: From Periodic Solutions to Instabilities and Chaos*, Resonance **24**, 87 (2019).

[4] D. Ó'Mathúna, *Integrable Systems in Celestial Mechanics*, 1st ed. (Birkhäuser Boston, 2008).

[5] W. J. Wild, *Euler's Three Body Problem*, American Journal of Physics **48**, 297 (1980).

[6] Z. E. Musielak and B. Quarles, The three-body problem, Reports on Progress in Physics **77**, 065901 (2014).

[7] H. Kinoshita, H. Yoshida, and H. Nakai, Symplectic integrators and their application to dynamical astronomy, Celestial Mechanics and Dynamical Astronomy **50**, 59 (1990).

[8] D. P. Sanders, colorline (2013).