Module 1 Final Project

MATH 285 – Advance Topics in Mathematics

San Jose State University, Fall 2021

Three–Body Problem

Introduction

The three–body problem describes the motion of three point mass particles under the influence of gravity.

The underlying fundamental laws that describe the motion of these particles is Newton's 2nd Law of Motion and Law of Gravitation.

Newton's 2nd Law of Motion can be written as $m\,\vec{a} = \sum \vec{F}_{\text{ext}}$ where m is the mass of the particle, $\vec{a}$ is the acceleration of the particle and $\sum \vec{F}_{\text{ext}}$ is the resultant of all external forces acting upon the particle.

Newton's Law of Gravitation describes the gravitational interaction between two particles which is needed for deriving the resultant, $\sum \vec{F}_{\text{ext}}$.

Behavior

So, I expect the behavior of the system to be periodic under certain initial conditions since we know astrophysical orbits exists.

If initial conditions are generated at random, then most likely solutions to the system will set off to infinity.

System of Nonlinear ODEs

According to Newton's 2nd Law and Law of Gravitation,

$$\begin{cases} m_1 \frac{d^2\vec{r}_1}{dt^2} &= G\,m_1\,m_2\,\frac{\vec{r}_2-\vec{r}_1}{\left\|\vec{r}_2-\vec{r}_1\right\|^3} + G\,m_1\,m_3\,\frac{\vec{r}_3-\vec{r}_1}{\left\|\vec{r}_3-\vec{r}_1\right\|^3} \\ m_2 \frac{d^2\vec{r}_2}{dt^2} &= G\,m_2\,m_1\,\frac{\vec{r}_1-\vec{r}_2}{\left\|\vec{r}_1-\vec{r}_2\right\|^3} + G\,m_2\,m_3\,\frac{\vec{r}_3-\vec{r}_2}{\left\|\vec{r}_3-\vec{r}_2\right\|^3} \\ m_3 \frac{d^2\vec{r}_3}{dt^2} &= G\,m_3\,m_1\,\frac{\vec{r}_1-\vec{r}_3}{\left\|\vec{r}_1-\vec{r}_3\right\|^3} + G\,m_3\,m_2\,\frac{\vec{r}_2-\vec{r}_3}{\left\|\vec{r}_2-\vec{r}_3\right\|^3} \end{cases} \Longrightarrow \begin{cases} \frac{d^2\vec{r}_1}{dt^2} &= G\,m_2\,\frac{\vec{r}_2-\vec{r}_1}{\left\|\vec{r}_2-\vec{r}_1\right\|^3} + G\,m_3\,\frac{\vec{r}_3-\vec{r}_1}{\left\|\vec{r}_3-\vec{r}_1\right\|^3} \\ \frac{d^2\vec{r}_2}{dt^2} &= G\,m_1\,\frac{\vec{r}_1-\vec{r}_2}{\left\|\vec{r}_1-\vec{r}_2\right\|^3} + G\,m_3\,\frac{\vec{r}_3-\vec{r}_2}{\left\|\vec{r}_3-\vec{r}_2\right\|^3} \\ \frac{d^2\vec{r}_3}{dt^2} &= G\,m_1\,\frac{\vec{r}_1-\vec{r}_3}{\left\|\vec{r}_1-\vec{r}_3\right\|^3} + G\,m_2\,\frac{\vec{r}_2-\vec{r}_3}{\left\|\vec{r}_2-\vec{r}_3\right\|^3} \end{cases}.$$

The system of vector equations can be rewritten as a system of nine scalar equations.

$$\begin{cases}
\frac{d^2 r_{1,1}}{dt^2} &= G\,m_2\,\frac{r_{2,1}-r_{1,1}}{\left((r_{2,1}-r_{1,1})^2+(r_{2,2}-r_{1,2})^2+(r_{2,3}-r_{1,3})^2\right)^{3/2}} + G\,m_3\,\frac{r_{3,1}-r_{1,1}}{\left((r_{3,1}-r_{1,1})^2+(r_{3,2}-r_{1,2})^2+(r_{3,3}-r_{1,3})^2\right)^{3/2}} \\
\frac{d^2 r_{1,2}}{dt^2} &= G\,m_2\,\frac{r_{2,2}-r_{1,2}}{\left((r_{2,1}-r_{1,1})^2+(r_{2,2}-r_{1,2})^2+(r_{2,3}-r_{1,3})^2\right)^{3/2}} + G\,m_3\,\frac{r_{3,2}-r_{1,2}}{\left((r_{3,1}-r_{1,1})^2+(r_{3,2}-r_{1,2})^2+(r_{3,3}-r_{1,3})^2\right)^{3/2}} \\
\frac{d^2 r_{1,3}}{dt^2} &= G\,m_2\,\frac{r_{2,3}-r_{1,3}}{\left((r_{2,1}-r_{1,1})^2+(r_{2,2}-r_{1,2})^2+(r_{2,3}-r_{1,3})^2\right)^{3/2}} + G\,m_3\,\frac{r_{3,3}-r_{1,3}}{\left((r_{3,1}-r_{1,1})^2+(r_{3,2}-r_{1,2})^2+(r_{3,3}-r_{1,3})^2\right)^{3/2}} \\
\frac{d^2 r_{2,1}}{dt^2} &= G\,m_1\,\frac{r_{1,1}-r_{2,1}}{\left((r_{1,1}-r_{2,1})^2+(r_{1,2}-r_{2,2})^2+(r_{1,3}-r_{2,3})^2\right)^{3/2}} + G\,m_3\,\frac{r_{3,1}-r_{2,1}}{\left((r_{3,1}-r_{2,1})^2+(r_{3,2}-r_{2,2})^2+(r_{3,3}-r_{2,3})^2\right)^{3/2}} \\
\frac{d^2 r_{2,2}}{dt^2} &= G\,m_1\,\frac{r_{1,2}-r_{2,2}}{\left((r_{1,1}-r_{2,1})^2+(r_{1,2}-r_{2,2})^2+(r_{1,3}-r_{2,3})^2\right)^{3/2}} + G\,m_3\,\frac{r_{3,2}-r_{2,2}}{\left((r_{3,1}-r_{2,1})^2+(r_{3,2}-r_{2,2})^2+(r_{3,3}-r_{2,3})^2\right)^{3/2}} \\
\frac{d^2 r_{2,3}}{dt^2} &= G\,m_1\,\frac{r_{1,3}-r_{2,3}}{\left((r_{1,1}-r_{2,1})^2+(r_{1,2}-r_{2,2})^2+(r_{1,3}-r_{2,3})^2\right)^{3/2}} + G\,m_3\,\frac{r_{3,3}-r_{2,3}}{\left((r_{3,1}-r_{2,1})^2+(r_{3,2}-r_{2,2})^2+(r_{3,3}-r_{2,3})^2\right)^{3/2}} \\
\frac{d^2 r_{3,1}}{dt^2} &= G\,m_1\,\frac{r_{1,1}-r_{3,1}}{\left((r_{1,1}-r_{3,1})^2+(r_{1,2}-r_{3,2})^2+(r_{1,3}-r_{3,3})^2\right)^{3/2}} + G\,m_2\,\frac{r_{2,1}-r_{3,1}}{\left((r_{2,1}-r_{3,1})^2+(r_{2,2}-r_{3,2})^2+(r_{2,3}-r_{3,3})^2\right)^{3/2}} \\
\frac{d^2 r_{3,2}}{dt^2} &= G\,m_1\,\frac{r_{1,2}-r_{3,2}}{\left((r_{1,1}-r_{3,1})^2+(r_{1,2}-r_{3,2})^2+(r_{1,3}-r_{3,3})^2\right)^{3/2}} + G\,m_2\,\frac{r_{2,2}-r_{3,2}}{\left((r_{2,1}-r_{3,1})^2+(r_{2,2}-r_{3,2})^2+(r_{2,3}-r_{3,3})^2\right)^{3/2}} \\
\frac{d^2 r_{3,3}}{dt^2} &= G\,m_1\,\frac{r_{1,3}-r_{3,3}}{\left((r_{1,1}-r_{3,1})^2+(r_{1,2}-r_{3,2})^2+(r_{1,3}-r_{3,3})^2\right)^{3/2}} + G\,m_2\,\frac{r_{2,3}-r_{3,3}}{\left((r_{2,1}-r_{3,1})^2+(r_{2,2}-r_{3,2})^2+(r_{2,3}-r_{3,3})^2\right)^{3/2}}
\end{cases}$$

To convert this system of 2nd order differential equations to 1st order differential equations, we make the following substitutions.

$$\begin{cases}
u_1 &= r_{1,1} \\
u_2 &= \frac{dr_{1,1}}{dt} \\
u_3 &= r_{1,2} \\
u_4 &= \frac{dr_{1,2}}{dt} \\
u_5 &= r_{1,3} \\
u_6 &= \frac{dr_{1,3}}{dt} \\
u_7 &= r_{2,1} \\
u_8 &= \frac{dr_{2,1}}{dt} \\
u_9 &= r_{2,2} \\
u_{10} &= \frac{dr_{2,2}}{dt} \\
u_{11} &= r_{2,3} \\
u_{12} &= \frac{dr_{2,3}}{dt} \\
u_{13} &= r_{3,1} \\
u_{14} &= \frac{dr_{3,1}}{dt} \\
u_{15} &= r_{3,2} \\
u_{16} &= \frac{dr_{3,2}}{dt} \\
u_{17} &= r_{3,3} \\
u_{18} &= \frac{dr_{3,3}}{dt}
\end{cases}$$

We end up with a system of 18 first order differential equations.

$$\begin{cases}
\frac{du_1}{dt} = u_2 \\
\frac{du_2}{dt} = G m_2 \frac{u_7-u_1}{\left((u_7-u_1)^2+(u_9-u_3)^2+(u_{11}-u_5)^2\right)^{3/2}} + G m_3 \frac{u_{13}-u_1}{\left((u_{13}-u_1)^2+(u_{15}-u_3)^2+(u_{17}-u_5)^2\right)^{3/2}} \\
\frac{du_3}{dt} = u_4 \\
\frac{du_4}{dt} = G m_2 \frac{u_9-u_3}{\left((u_7-u_1)^2+(u_9-u_3)^2+(u_{11}-u_5)^2\right)^{3/2}} + G m_3 \frac{u_{15}-u_3}{\left((u_{13}-u_1)^2+(u_{15}-u_3)^2+(u_{17}-u_5)^2\right)^{3/2}} \\
\frac{du_5}{dt} = u_6 \\
\frac{du_6}{dt} = G m_2 \frac{u_{11}-u_5}{\left((u_7-u_1)^2+(u_9-u_3)^2+(u_{11}-u_5)^2\right)^{3/2}} + G m_3 \frac{u_{17}-u_5}{\left((u_{13}-u_1)^2+(u_{15}-u_3)^2+(u_{17}-u_5)^2\right)^{3/2}} \\
\frac{du_7}{dt} = u_8 \\
\frac{du_8}{dt} = G m_1 \frac{u_1-u_7}{\left((u_1-u_7)^2+(u_3-u_9)^2+(u_5-u_{11})^2\right)^{3/2}} + G m_3 \frac{u_{13}-u_7}{\left((u_{13}-u_7)^2+(u_{15}-u_9)^2+(u_{17}-u_{11})^2\right)^{3/2}} \\
\frac{du_9}{dt} = u_{10} \\
\frac{du_{10}}{dt} = G m_1 \frac{u_3-u_9}{\left((u_1-u_7)^2+(u_3-u_9)^2+(u_5-u_{11})^2\right)^{3/2}} + G m_3 \frac{u_{15}-u_9}{\left((u_{13}-u_7)^2+(u_{15}-u_9)^2+(u_{17}-u_{11})^2\right)^{3/2}} \\
\frac{du_{11}}{dt} = u_{12} \\
\frac{du_{12}}{dt} = G m_1 \frac{u_5-u_{11}}{\left((u_1-u_7)^2+(u_3-u_9)^2+(u_5-u_{11})^2\right)^{3/2}} + G m_3 \frac{u_{17}-u_{11}}{\left((u_{13}-u_7)^2+(u_{15}-u_9)^2+(u_{17}-u_{11})^2\right)^{3/2}} \\
\frac{du_{13}}{dt} = u_{14} \\
\frac{du_{14}}{dt} = G m_1 \frac{u_1-u_{13}}{\left((u_1-u_{13})^2+(u_3-u_{15})^2+(u_5-u_{17})^2\right)^{3/2}} + G m_2 \frac{u_7-u_{13}}{\left((u_7-u_{13})^2+(u_9-u_{15})^2+(u_{11}-u_{17})^2\right)^{3/2}} \\
\frac{du_{15}}{dt} = u_{16} \\
\frac{du_{16}}{dt} = G m_1 \frac{u_3-u_{15}}{\left((u_1-u_{13})^2+(u_3-u_{15})^2+(u_5-u_{17})^2\right)^{3/2}} + G m_2 \frac{u_9-u_{15}}{\left((u_7-u_{13})^2+(u_9-u_{15})^2+(u_{11}-u_{17})^2\right)^{3/2}} \\
\frac{du_{17}}{dt} = u_{18} \\
\frac{du_{18}}{dt} = G m_1 \frac{u_5-u_{17}}{\left((u_1-u_{13})^2+(u_3-u_{15})^2+(u_5-u_{17})^2\right)^{3/2}} + G m_2 \frac{u_{11}-u_{17}}{\left((u_7-u_{13})^2+(u_9-u_{15})^2+(u_{11}-u_{17})^2\right)^{3/2}}
\end{cases}$$

Related Concepts

Total Momentum

$$\vec{P} = \sum_k m_k \frac{d\vec{r}_k}{dt}$$

Total Angular Momentum

$$\vec{L} = \sum_k \vec{r}_k \times m_k \vec{p}_k$$

Total Energy

$$E = \frac{1}{2} \sum_k m_k \left\| \frac{d\vec{r}_k}{dt} \right\|^2 - \sum_{j>k} \frac{G m_j m_k}{\left\| \vec{r}_j - \vec{r}_k \right\|}$$

Schemes

For generating solutions simply, we set $m_1 = m_2 = m_3 = G = 1$.

Ovals with Flourishes

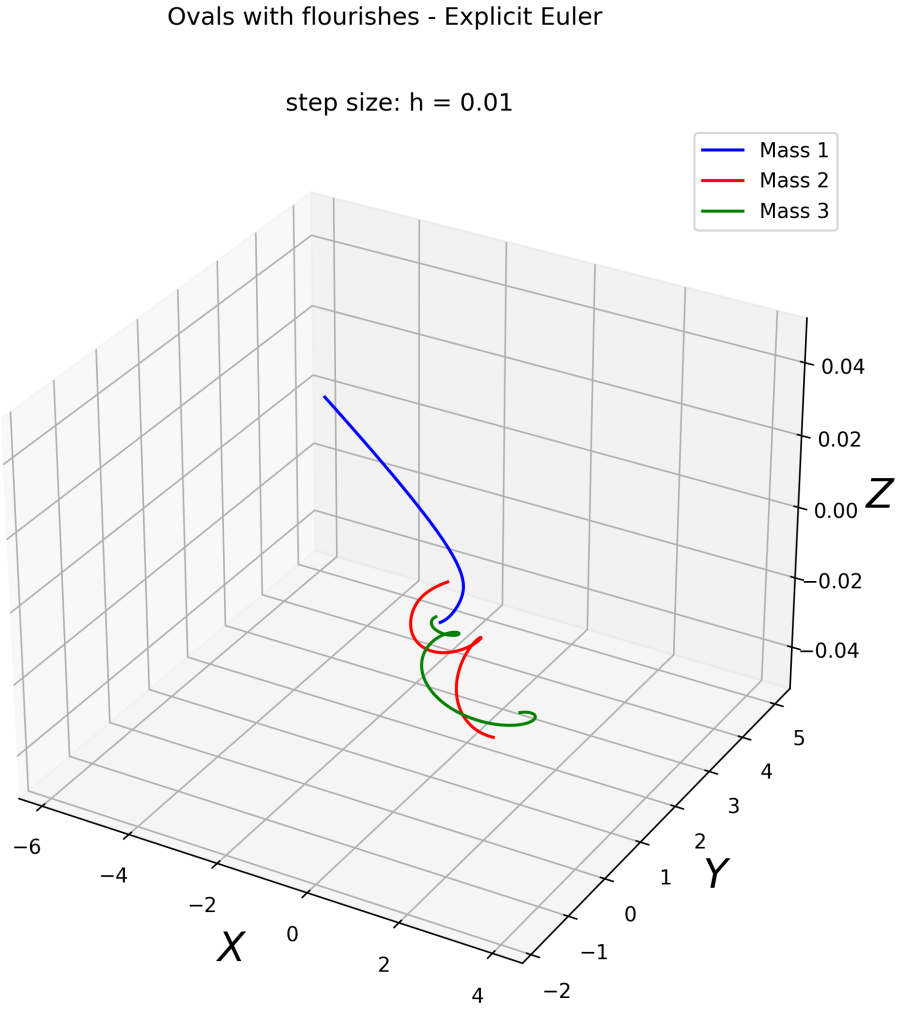
Initial Positions:

$\vec{r}_1 = \langle 0.716248295713, 0.384288553041, 0\rangle$, $\vec{r}_2 = \langle 0.086172594591, 1.342795868577, 0\rangle$, $\vec{r}_3 = \langle 0.538777980808, 0.481049882656, 0\rangle$

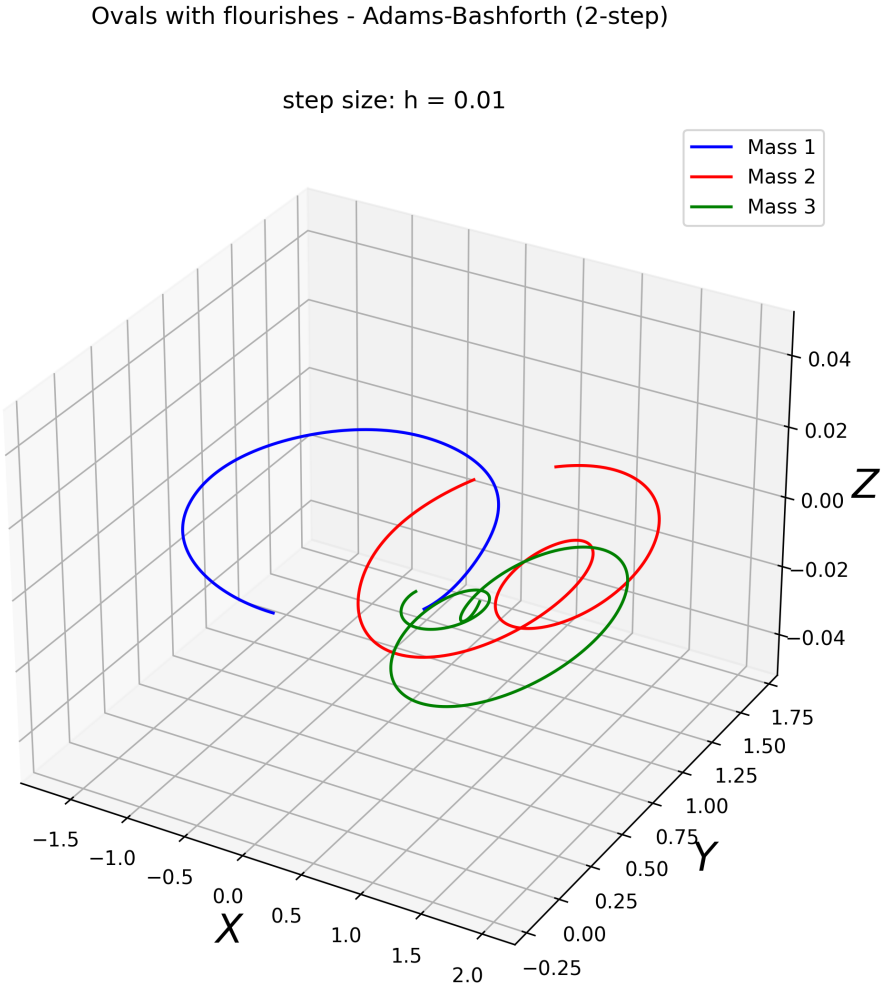
Initial Velocities:

$\vec{v}_1 = \langle 1.245268230896, 2.444311951777, 0\rangle$, $\vec{v}_2 = \langle -0.675224323690, -0.962879613630, 0\rangle$, $\vec{v}_3 = \langle -0.570043907206, -1.481432338147, 0\rangle$

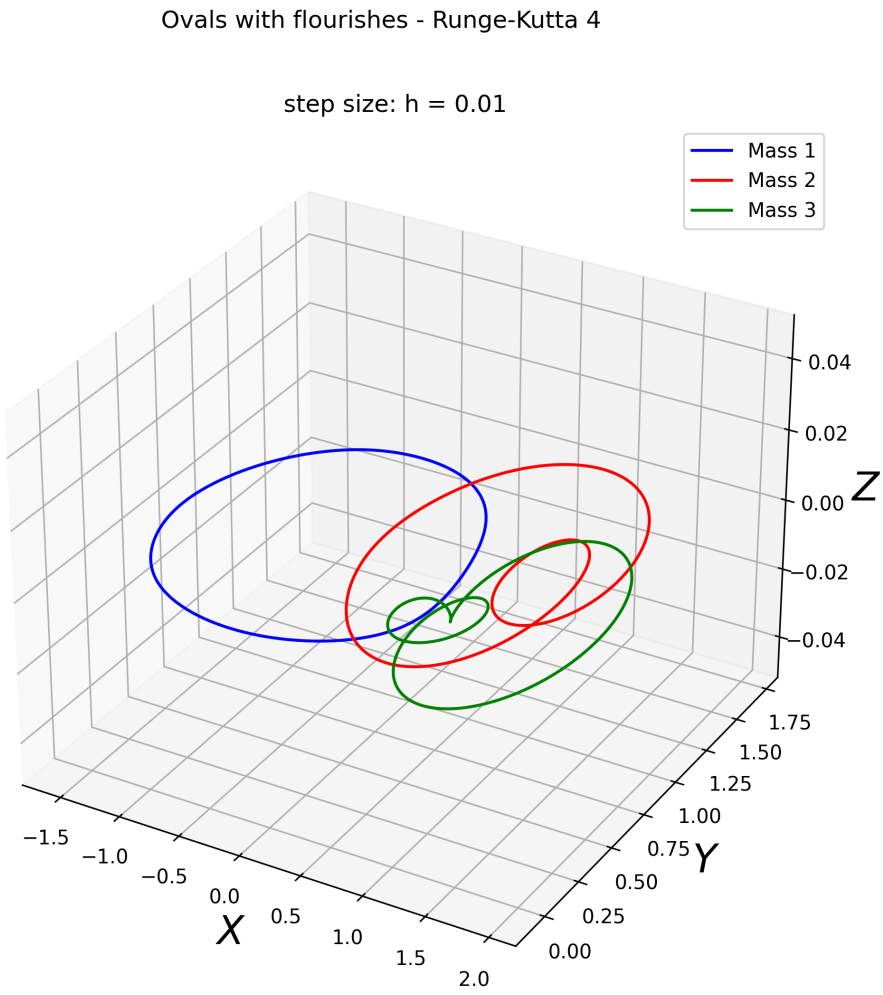
Explicit Euler



Adams–Bashforth (2–step)



Runge–Kutta 4



Remarks

Ovals with Flourishes

The Runge–Kutta 4 scheme worked best since it produced closed orbits for a step size of $h = 0.01$.

For the Adams–Bashforth scheme, the solutions were almost as good as the RK4 solutions. The orbits are not closed which most likely implies that the step size needed to be smaller.

For the Explicit–Euler scheme, the solutions are unstable and go off towards infinity.

Here, Runge–Kutta 4 is considered the more complicated scheme. Thus it is worth the effort to imple-

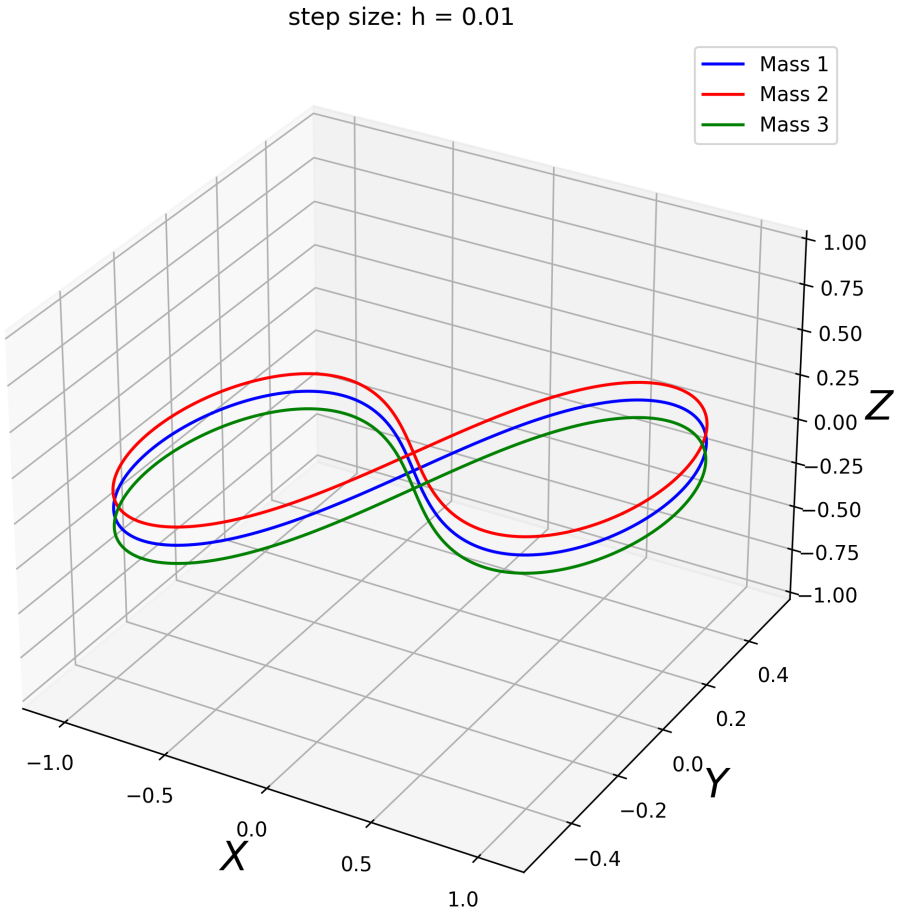
ment a Runge–Kutta 4 method for the three–body problem since it produced great results for a relatively small amount of iterations ($n = 100$).

Other Orbits (Solutions)

Figure 8

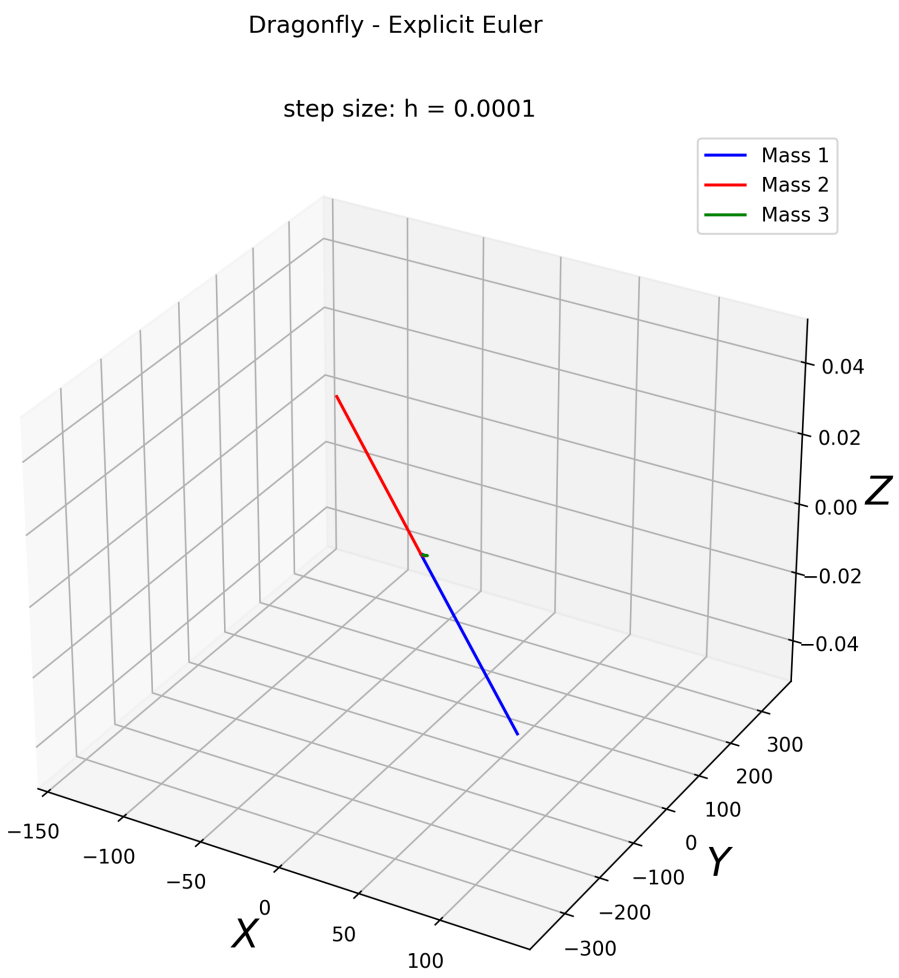
Runge–Kutta 4

Figure 8 - Runge-Kutta 4

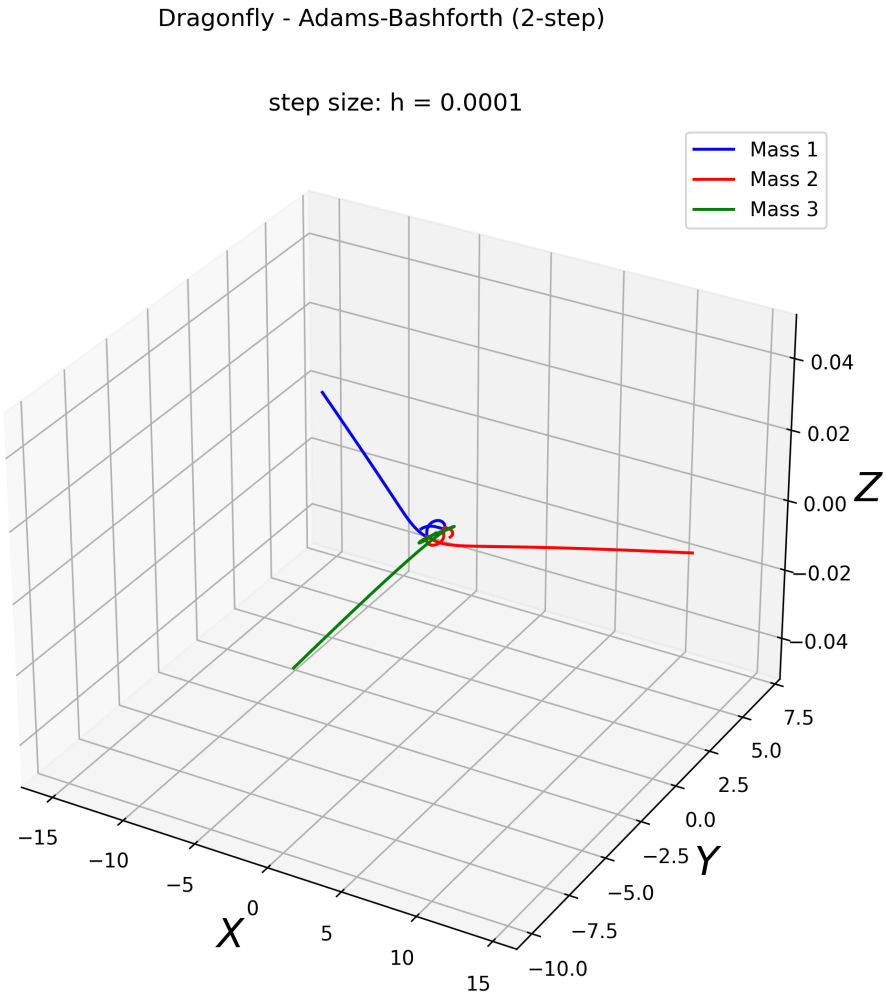


Dragonfly

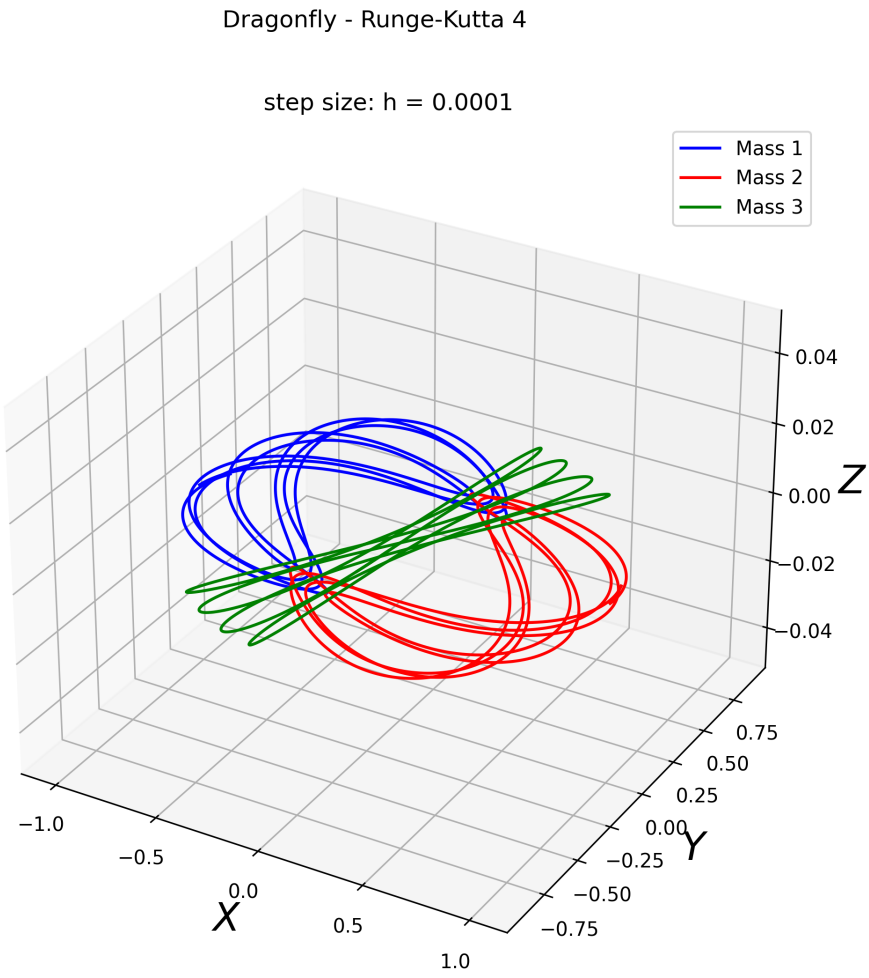
Explicit Euler



Adams–Bashforth (2–step)

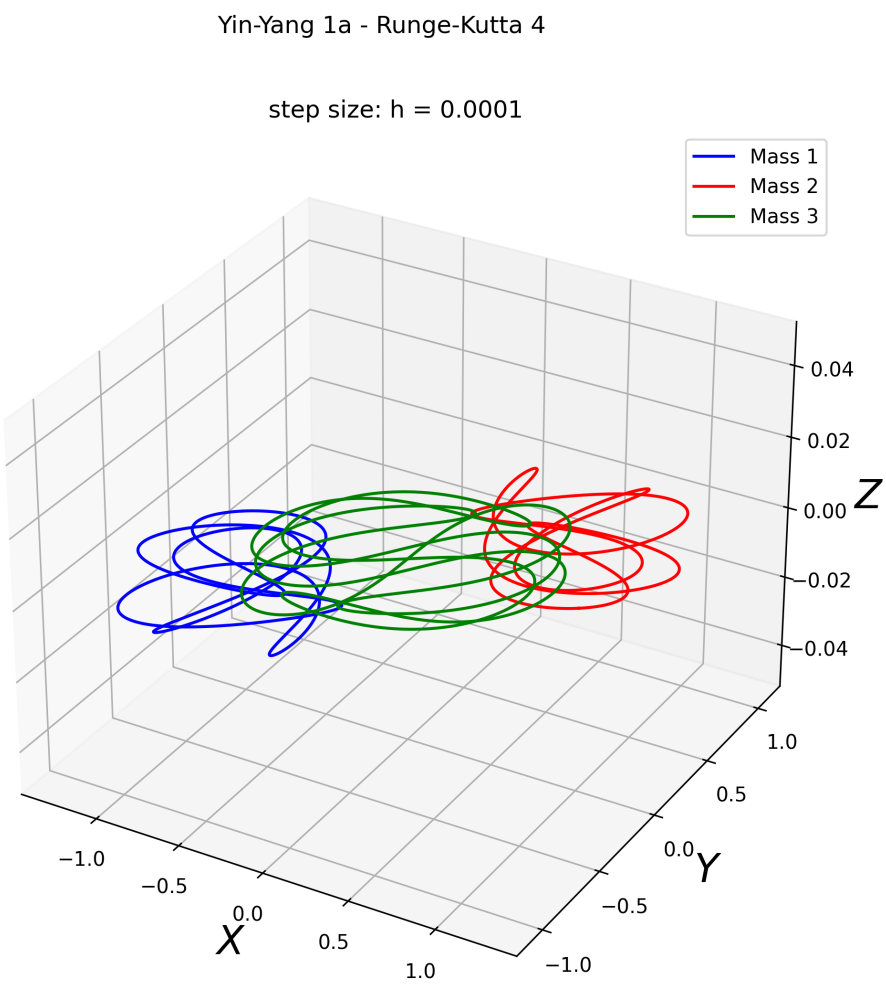


Runge–Kutta 4



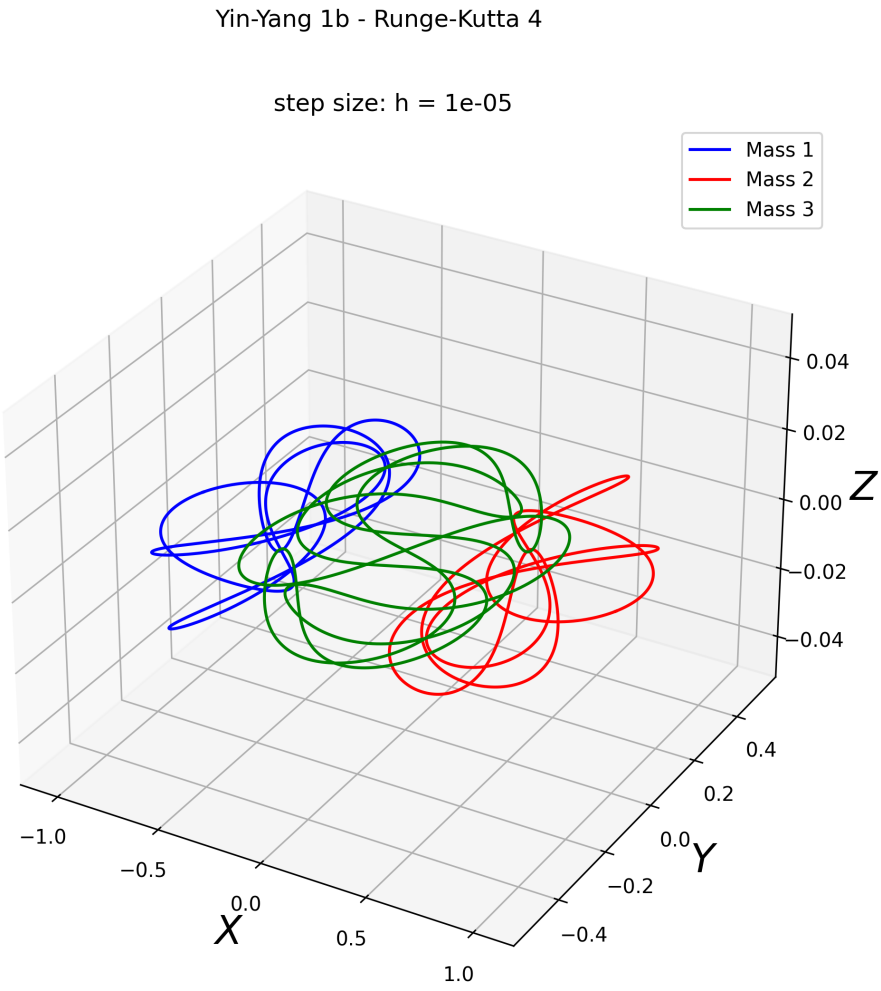
Yin–Yang 1A

Runge–Kutta 4



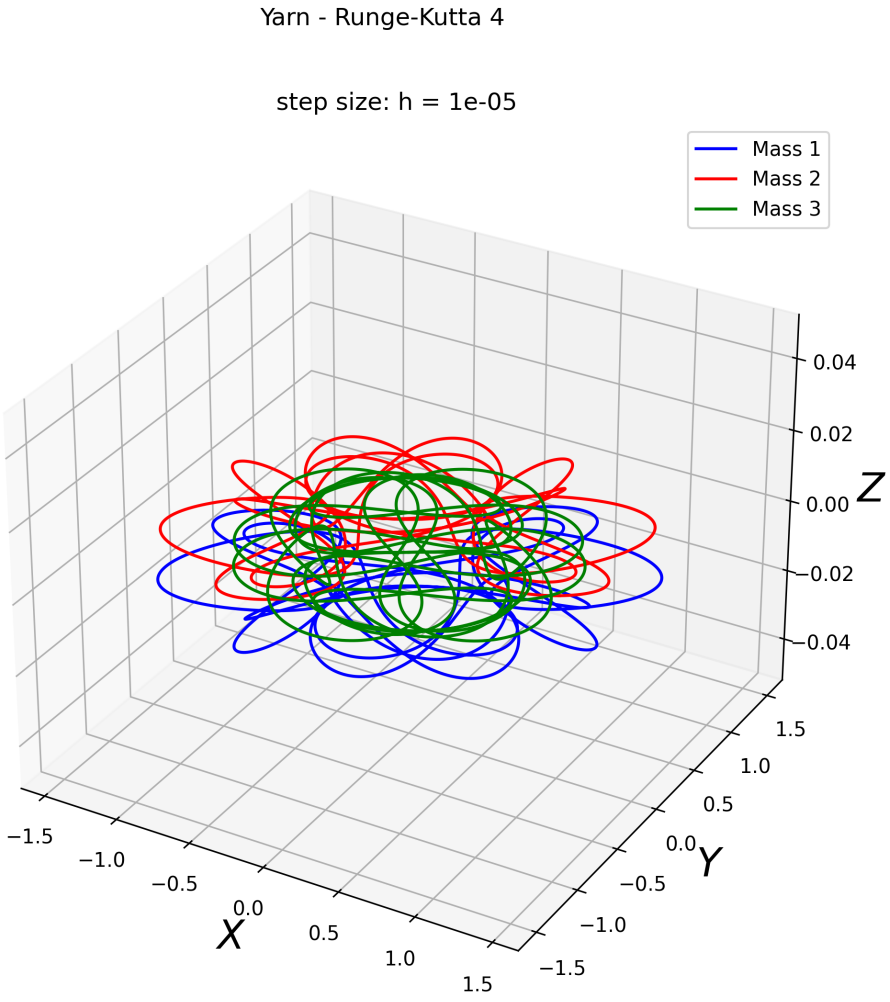
Yin–Yang 1B

Runge–Kutta 4



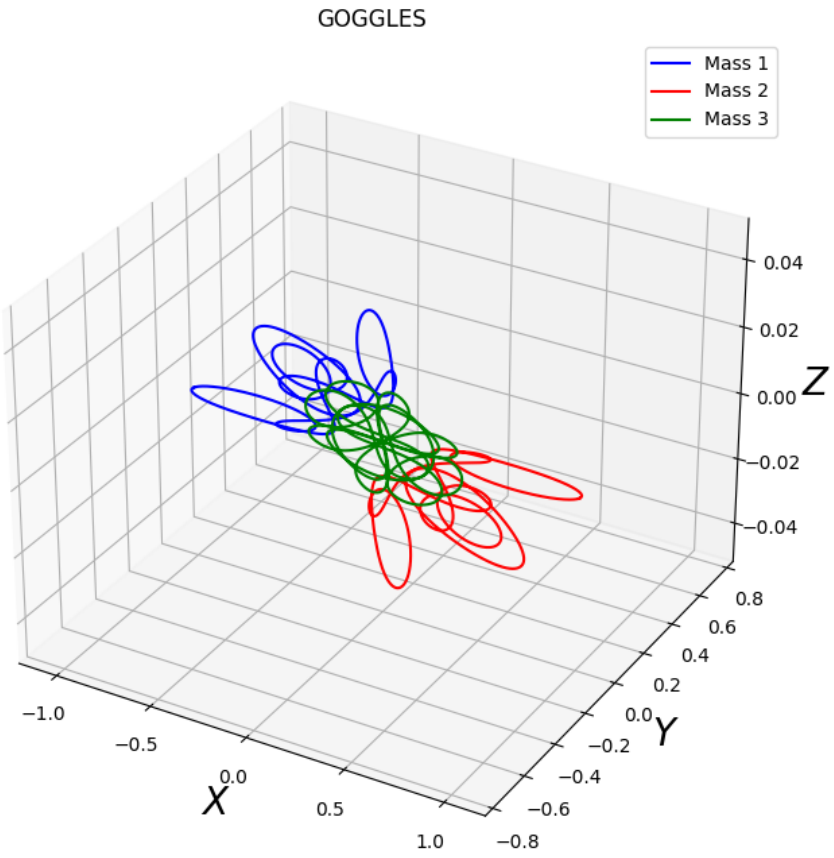
Yarn

Runge–Kutta 4



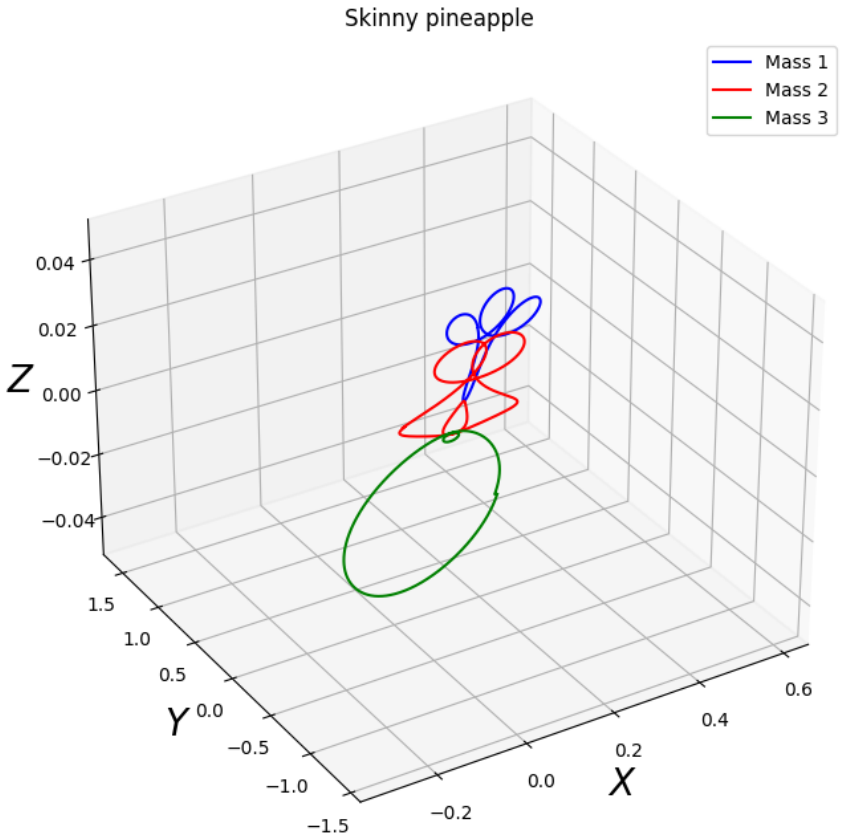
Goggles

Runge–Kutta 4



Skinny Pineapple

Runge–Kutta 4



Code

```
import numpy as np
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as ani

GRAVITATIONAL_CONSTANT = 6.67408e-11

def explicit_euler(F, Y0, h, n):
    Y = []
```

```
    Y.append(Y0)
    for k in range(n - 1):
        Y.append(Y[k] + h * F(*Y[k]))
    return Y

def adams_bashforth(F, Y0, h, n):
    Y = []
    Y.append(Y0)
    Y.append(Y0 + h * F(*Y0))
    for k in range(n - 2):
        Y.append(Y[k + 1] + h * ((3 / 2) * F(*Y[k + 1]) - (1 / 2) * F(*Y[k])))
    return Y

def runge_kutta(F, Y0, h, n):
    Y = []
    Y.append(Y0)
    for k in range(n - 1):
        mk = F(*Y[k]) # (Forward) Euler
        nk = F(*(Y[k] + mk * h / 2)) # Midpoint slope
        pk = F(*(Y[k] + nk * h / 2)) # Better midpoint slope
        qk = F(*(Y[k] + pk * h)) # Endpoint slope
        Y.append(Y[k] + (h / 6) * (mk + 2 * nk + 2 * pk + qk))
    return Y

def main():
    # t-interval [t_min,t_max] with step size h
    t_min, t_max, h = 0, 8.094721, 0.01

    name = "Ovals with flourishes"

    # Initial positions
    r_1_x, r_1_y, r_1_z = 0.716248295713, 0.384288553041, 0
    r_2_x, r_2_y, r_2_z = 0.086172594591, 1.342795868577, 0
    r_3_x, r_3_y, r_3_z = 0.538777980808, 0.481049882656, 0

    # Initial velocities
    v_1_x, v_1_y, v_1_z = 1.245268230896, 2.444311951777, 0
    v_2_x, v_2_y, v_2_z = -0.675224323690, -0.962879613630, 0
    v_3_x, v_3_y, v_3_z = -0.570043907206, -1.481432338147, 0

    ########################################

    ## t-interval [t_min,t_max] with step size h
    #t_min, t_max, h = 0, 6.324449, 0.01

    #name = "Figure 8"
    #p1, p2 = 0.347111, 0.532728

    ## Initial positions
    #r_1_x, r_1_y, r_1_z = -1, 0, 0
    #r_2_x, r_2_y, r_2_z = 1, 0, 0
    #r_3_x, r_3_y, r_3_z = 0, 0, 0

    ## Initial velocities
    #v_1_x, v_1_y, v_1_z = p1, p2, 0
    #v_2_x, v_2_y, v_2_z = p1, p2, 0
    #v_3_x, v_3_y, v_3_z = -2 * p1, -2 * p2, 0

    ########################################
```

```
## t-interval [t_min,t_max] with step size h
#t_min, t_max, h = 0, 21.270975, 0.0001

#name = "Dragonfly"
#p1, p2 = 0.080584, 0.588836

## Initial positions
#r_1_x, r_1_y, r_1_z = -1, 0, 0
#r_2_x, r_2_y, r_2_z = 1, 0, 0
#r_3_x, r_3_y, r_3_z = 0, 0, 0

## Initial velocities
#v_1_x, v_1_y, v_1_z = p1, p2, 0
#v_2_x, v_2_y, v_2_z = p1, p2, 0
#v_3_x, v_3_y, v_3_z = -2 * p1, -2 * p2, 0

########################################

## t-interval [t_min,t_max] with step size h
#t_min, t_max, h = 0, 10.962563, 0.00001

#name = "Yin-Yang 1b"
#p1, p2 = 0.282699, 0.327209

## Initial positions
#r_1_x, r_1_y, r_1_z = -1, 0, 0
#r_2_x, r_2_y, r_2_z = 1, 0, 0
#r_3_x, r_3_y, r_3_z = 0, 0, 0

## Initial velocities
#v_1_x, v_1_y, v_1_z = p1, p2, 0
#v_2_x, v_2_y, v_2_z = p1, p2, 0
#v_3_x, v_3_y, v_3_z = -2 * p1, -2 * p2, 0

########################################

## t-interval [t_min,t_max] with step size h
#t_min, t_max, h = 0, 17.328370, 0.0001

#name = "Yin-Yang 1a"
#p1, p2 = 0.513938, 0.304736

## Initial positions
#r_1_x, r_1_y, r_1_z = -1, 0, 0
#r_2_x, r_2_y, r_2_z = 1, 0, 0
#r_3_x, r_3_y, r_3_z = 0, 0, 0

## Initial velocities
#v_1_x, v_1_y, v_1_z = p1, p2, 0
#v_2_x, v_2_y, v_2_z = p1, p2, 0
#v_3_x, v_3_y, v_3_z = -2 * p1, -2 * p2, 0

########################################

## t-interval [t_min,t_max] with step size h
#t_min, t_max, h = 0, 55.501762, 0.00001

#name = "Yarn"
#p1, p2 = 0.559064, 0.349192
```

```
    ## Initial positions
    #r_1_x, r_1_y, r_1_z = -1, 0, 0
    #r_2_x, r_2_y, r_2_z = 1, 0, 0
    #r_3_x, r_3_y, r_3_z = 0, 0, 0

    ## Initial velocities
    #v_1_x, v_1_y, v_1_z = p1, p2, 0
    #v_2_x, v_2_y, v_2_z = p1, p2, 0
    #v_3_x, v_3_y, v_3_z = -2 * p1, -2 * p2, 0

    ########################################

    ## t-interval [t_min,t_max] with step size h
    #t_min, t_max, h = 0, 10.466818, 0.0001

    #name = "GOGGLES"
    #p1, p2 = 0.083300, 0.127889

    ## Initial positions
    #r_1_x, r_1_y, r_1_z = -1, 0, 0
    #r_2_x, r_2_y, r_2_z = 1, 0, 0
    #r_3_x, r_3_y, r_3_z = 0, 0, 0

    ## Initial velocities
    #v_1_x, v_1_y, v_1_z = p1, p2, 0
    #v_2_x, v_2_y, v_2_z = p1, p2, 0
    #v_3_x, v_3_y, v_3_z = -2 * p1, -2 * p2, 0

    ########################################

    ## t-interval [t_min,t_max] with step size h
    #t_min, t_max, h = 0, 5.095054, 0.000001

    #name = "Skinny pineapple"

    ## Initial positions
    #r_1_x, r_1_y, r_1_z = 0.419698802831, 1.190466261252, 0
    #r_2_x, r_2_y, r_2_z = 0.076399621771, 0.296331688995, 0
    #r_3_x, r_3_y, r_3_z = 0.100310663856, -0.729358656127, 0

    ## Initial velocities
    #v_1_x, v_1_y, v_1_z = 0.102294566003, 0.687248445943, 0
    #v_2_x, v_2_y, v_2_z = 0.148950262064, 0.240179781043, 0
    #v_3_x, v_3_y, v_3_z = -0.251244828060, -0.927428226977, 0

    ########################################

    # number of iterations
    n = int((t_max - t_min) / h) + 1

    # Constants
    G, m_1, m_2, m_3 = 1, 1, 1, 1

    # list of t values
    T = np.linspace(t_min, t_max, n)

    # Initial conditions
    U0 =
np.array([r_1_x,v_1_x,r_1_y,v_1_y,r_1_z,v_1_z,r_2_x,v_2_x,r_2_y,v_2_y,r_2_z,v_2_z,r_3_x,v_3_x,r_3_
y,v_3_y,r_3_z,v_3_z])
```

```
    ########## Velocity Equations ##########

    v_1 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_2 * (u_7 - u_1) / ((u_7 - u_1) ** 2 + (u_9 - u_3) ** 2 + (u_11 - u_5) ** 2) **
(3/2) + \
             m_3 * (u_13 - u_1) / ((u_13 - u_1) ** 2 + (u_15 - u_3) ** 2 + (u_17 - u_5) ** 2) **
(3/2))
    v_2 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_2 * (u_9 - u_3) / ((u_7 - u_1) ** 2 + (u_9 - u_3) ** 2 + (u_11 - u_5) ** 2) **
(3/2) + \
             m_3 * (u_15 - u_3) / ((u_13 - u_1) ** 2 + (u_15 - u_3) ** 2 + (u_17 - u_5) ** 2) **
(3/2))
    v_3 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_2 * (u_11 - u_5) / ((u_7 - u_1) ** 2 + (u_9 - u_3) ** 2 + (u_11 - u_5) ** 2) **
(3/2) + \
             m_3 * (u_17 - u_5) / ((u_13 - u_1) ** 2 + (u_15 - u_3) ** 2 + (u_17 - u_5) ** 2) **
(3/2))
    v_4 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_1 * (u_1 - u_7) / ((u_1 - u_7) ** 2 + (u_3 - u_9) ** 2 + (u_5 - u_11) ** 2) **
(3/2) + \
             m_3 * (u_13 - u_7) / ((u_13 - u_7) ** 2 + (u_15 - u_9) ** 2 + (u_17 - u_11) ** 2) **
(3/2))
    v_5 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_1 * (u_3 - u_9) / ((u_1 - u_7) ** 2 + (u_3 - u_9) ** 2 + (u_5 - u_11) ** 2) **
(3/2) + \
             m_3 * (u_15 - u_9) / ((u_13 - u_7) ** 2 + (u_15 - u_9) ** 2 + (u_17 - u_11) ** 2) **
(3/2))
    v_6 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_1 * (u_5 - u_11) / ((u_1 - u_7) ** 2 + (u_3 - u_9) ** 2 + (u_5 - u_11) ** 2) **
(3/2) + \
             m_3 * (u_17 - u_11) / ((u_13 - u_7) ** 2 + (u_15 - u_9) ** 2 + (u_17 - u_11) ** 2)
** (3/2))
    v_7 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_1 * (u_1 - u_13) / ((u_1 - u_13) ** 2 + (u_3 - u_15) ** 2 + (u_5 - u_17) ** 2) **
(3/2) + \
             m_2 * (u_7 - u_13) / ((u_7 - u_13) ** 2 + (u_9 - u_15) ** 2 + (u_11 - u_17) ** 2) **
(3/2))
    v_8 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_1 * (u_3 - u_15) / ((u_1 - u_13) ** 2 + (u_3 - u_15) ** 2 + (u_5 - u_17) ** 2) **
(3/2) + \
             m_2 * (u_9 - u_15) / ((u_7 - u_13) ** 2 + (u_9 - u_15) ** 2 + (u_11 - u_17) ** 2) **
(3/2))
    v_9 = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        G * (m_1 * (u_5 - u_17) / ((u_1 - u_13) ** 2 + (u_3 - u_15) ** 2 + (u_5 - u_17) ** 2) **
(3/2) + \
             m_2 * (u_11 - u_17) / ((u_7 - u_13) ** 2 + (u_9 - u_15) ** 2 + (u_11 - u_17) ** 2)
** (3/2))

    ########## Vector Function ##########
```

```
    F = lambda u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13, u_14, u_15,
u_16, u_17, u_18: \
        np.array([u_2, v_1(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_4, v_2(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_6, v_3(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_8, v_4(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_10, v_5(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_12, v_6(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_14, v_7(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_16, v_8(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18), \
                  u_18, v_9(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_10, u_11, u_12, u_13,
u_14, u_15, u_16, u_17, u_18)])

    ########## Scheme ##########

    #U = explicit_euler(F, U0, h, n)
    #name += ' - Explicit Euler'

    #U = adams_bashforth(F, U0, h, n)
    #name += ' - Adams-Bashforth (2-step)'

    U = runge_kutta(F, U0, h, n)
    name += ' - Runge-Kutta 4'

    R1, R2, R3 = [], [], []
    for u in U:
        R1.append([u[index] for index in [0, 2, 4]])
        R2.append([u[index] for index in [6, 8, 10]])
        R3.append([u[index] for index in [12, 14, 16]])

    global X1, X2, X3, Y1, Y2, Y3, Z1, Z2, Z3

    X1 = [p[0] for p in R1]
    Y1 = [p[1] for p in R1]
    Z1 = [p[2] for p in R1]

    X2 = [p[0] for p in R2]
    Y2 = [p[1] for p in R2]
    Z2 = [p[2] for p in R2]

    X3 = [p[0] for p in R3]
    Y3 = [p[1] for p in R3]
    Z3 = [p[2] for p in R3]

    ########## Plot ##########

    global ax

    fig_1 = plt.figure(num = 1, figsize=(8,8))
    ax = plt.axes(projection ='3d')
    plt.suptitle(name)
    plt.title('step size: h = {}'.format(h))
```

```
    ax.plot3D(X1, Y1, Z1, 'blue', label = 'Mass 1')
    ax.plot3D(X2, Y2, Z2, 'red', label = 'Mass 2')
    ax.plot3D(X3, Y3, Z3, 'green', label = 'Mass 3')
    ax.set_xlabel('$X$', fontsize=20)
    ax.set_ylabel('$Y$', fontsize=20)
    ax.set_zlabel('$Z$', fontsize=20)
    #ax.set_zlim([-1, 1])
    ax.legend()
    plt.savefig('{}.png'.format(name), dpi = 300)

    ########## Animate ##########

    global frames

    frames, fps = 60, 10

    fig_2 = plt.figure(num = 2, figsize=(8,8))
    ax = plt.axes(projection ='3d')
    plt.suptitle(name)
    plt.title('step size: h = {}'.format(h))
    animator = ani.FuncAnimation(fig_2, buildmebarchart, frames = frames)
    animator.save('{}.gif'.format(name), fps = fps)

    ########## Show ##########

    plt.close(fig_2)
    plt.show()

def buildmebarchart(i=int):
    k = i * (math.floor(len(X1) / frames))
    ax.plot3D(X1[:k], Y1[:k], Z1[:k], 'blue', label = 'Mass 1')
    ax.plot3D(X2[:k], Y2[:k], Z2[:k], 'red', label = 'Mass 2')
    ax.plot3D(X3[:k], Y3[:k], Z3[:k], 'green', label = 'Mass 3')
    ax.set_xlabel('$X$', fontsize=20)
    ax.set_ylabel('$Y$', fontsize=20)
    ax.set_zlabel('$Z$', fontsize=20)
    handles, labels = plt.gca().get_legend_handles_labels()
    by_label = dict(zip(labels, handles))
    plt.legend(by_label.values(), by_label.keys())

if __name__ == '__main__':
    main()
```