


Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestones 5

Ampliación de Matemáticas 1

6 de diciembre de 2022

Autor:

- Javier Zatón Miguel

Índice

1. Objetivo del software

Este programa tiene como objetivo determinar los movimientos individuales de un grupo de cuerpos que interactúan mutuamente según las leyes de la gravitación universal de Newton, en un caso simplificado donde todos los cuerpos tienen la misma masa.

2. Desarrollo de código

2.1. Módulos

Para poder integrar el problema de N cuerpos, se ha creado una función denominada KeplerNbody, esta será llamada por el módulo de integración de Cauchy, con la diferencia que en este caso, el input U, en vez de ser un vector será una matriz, que tendrá la siguiente forma:

$$U = \begin{pmatrix} \begin{pmatrix} \\ r \\ \end{pmatrix} \\ \\ \begin{pmatrix} \\ v \\ \end{pmatrix} \end{pmatrix} \tag{1}$$

siendo las matrices r y matrices v:

$$r = \begin{pmatrix} rx1 & ry1 & rz1 \\ rx2 & ry2 & rz2 \\ \vdots & \vdots & \vdots \\ rxNb & ryNb & rzNb \end{pmatrix} \tag{2}$$

$$v = \begin{pmatrix} vx1 & vy1 & vz1 \\ vx2 & vy2 & vz2 \\ \vdots & \vdots & \vdots \\ vxNb & vyNb & vzNb \end{pmatrix} \tag{3}$$

Nb representa el número de cuerpos que se va a estudiar, la función es la siguiente:

```
def KeplerNbody(U, t):

    Nb = (size(U)/3)/2
    Nb = int(Nb)

    F = zeros([2*Nb,3])
    for i in range(0,Nb):
        F[i,:] = U[Nb+i,:]

    a0 = zeros([Nb,3])

    for i in range(Nb):
      for j in range(Nb):
        if j != i:
            d = U[j,:] - U[i,:]
            a0[i,:] =  a0[i,:] + d[:]/norm(d)**3

    for i in range(0,Nb):
        F[Nb+i,:] = a0[i,:]

    return  F
```

La función está pensada para que al insertar la matriz U del programa principal, calcule el número de cuerpos que hay que analizar, Nb. Después se crea una matriz F de las mismas dimensiones que U, basándose en la función de Kepler creada en el Milestones 2, esta matriz F va a ser el dato que devuelva la función, que va a estar compuesta de dos tipos de datos, la velocidad que aporta la matriz U y las aceleraciones que tendrán que ser calculadas a continuación.

$$F = \begin{pmatrix} \begin{pmatrix} \\ v \\ \end{pmatrix} \\ \\ \begin{pmatrix} \\ a \\ \end{pmatrix} \end{pmatrix} \tag{4}$$

$$a = \begin{pmatrix} ax1 & ay1 & az1 \\ ax2 & ay2 & az2 \\ \vdots & \vdots & \vdots \\ axNb & ayNb & azNb \end{pmatrix} \tag{5}$$

Aplicando la ley de gravitación universal, se despeja la aceleración de un cuerpo como:

$$\mathbf{a}_i = G \sum_{j \neq i} m_j \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|^3}$$

Para calcular la aceleración de un cuerpo i , se necesita calcular las respectivas distancias del resto de cuerpos (excepto de sí mismo), en la función se ha creado un bucle, que calcula las aceleraciones de las componentes x,y,z por separado, dispuestas de la misma manera que las sub-matrices R y V, después de obtener la matriz de aceleraciones como indica la ecuación 5, se colocan en la mitad inferior de la matriz F para hacer el return.

En el programa se ha supuesto una constante G = 1 y que todas las masas de cada cuerpo son unidad, en versiones de prueba solo se ha conseguido con un vector interno de la función, que se tendría que cambiar en el módulo, por lo que se ha descartado en un inicio.

2.2. Programa principal

En el programa principal , primero se definen las condiciones iniciales del problema a resolver, el número de cuerpos que se va a estudiar, y estos vectores fila se acoplan a ambas matrices de condiciones iniciales r0 y v0.

```
10  Nb = 5
11  #Body1
12  r01 = array([0.4,0.1,0])
13  v01 = array([0,0.4,1])
14  #Body2
15  r02 = array([0.2,1,0])
16  v02 = array([0,0.9,1])
17  #Body3
18  r03 = array([0.5,0.5,1])
19  v03 = array([1,-1,0.6])
20  #Body4
21  r04 = array([0,0,0])
22  v04 = array([-0.3,0,-0.3])
23  #Body5
24  r05 = array([0.2,-0.1,0.9])
25  v05 = array([-0.5,1,-0.1])
26
27  r0 = zeros([Nb,3])
28  v0 = zeros([Nb,3])
29
30  r0[0,:] = r01
31  r0[1,:] = r02
32  r0[2,:] = r03
33  r0[3,:] = r04
34  r0[4,:] = r05
35
36  v0[0,:] = v01
37  v0[1,:] = v02
38  v0[2,:] = v03
39  v0[3,:] = v04
40  v0[4,:] = v05
```

Luego, el programa principal de forma automática ensambla las dos matrices de condiciones iniciales en la matriz U0, teniendo en cuenta el parámetro Nb.

```
43  U0 = zeros((2*Nb,3))
44
45  for i in range(0,Nb):
46      U0[i,:] = r0[i,:]
47      U0[Nb+i,:] = v0[i,:]
48
```

Antes de aplicar el problema de Cauchy a la función de N cuerpos, se define la precisión y el número de iteraciones, con estos dos datos se calcula el tiempo de integración y se compone el vector lint, que se necesita para Cauchy.

```
50
51    dt = 0.00001
52    Nt = 200000
53    ti = 0
54    tf = dt*(Nt-1)+ti
55    lint = linspace(ti,tf,Nt+1)
56
57    r = Cauchy(KeplerNbody,U0,lint,RK4)
58
```

Aplicando Cauchy se obtiene una matriz de 4 dimensiones r(:,:,:).

El primer término determina el cuerpo, por ejemplo r(1,:,:), devuelve la posición y velocidad del segundo cuerpo (porque python empieza a contar en 0).

El segundo término son las coordenadas cartesianas, que siempre van a ser 3, (x,y,z).

El tercer término corresponde al paso de integración, por lo que las condiciones iniciales corresponden al valor 0, hasta Nt.

```
60
61    ax = plt.figure().add_subplot(projection='3d')
62    for i in range(0,Nb):
63    |    ax.plot(r[i,0,:],r[i,1,:],r[i,2,:])
64
65    ax.set_xlim([-0.5, 1])
66    ax.set_ylim([-0.2, 1.2])
67
68    plt.show()
69
70
```

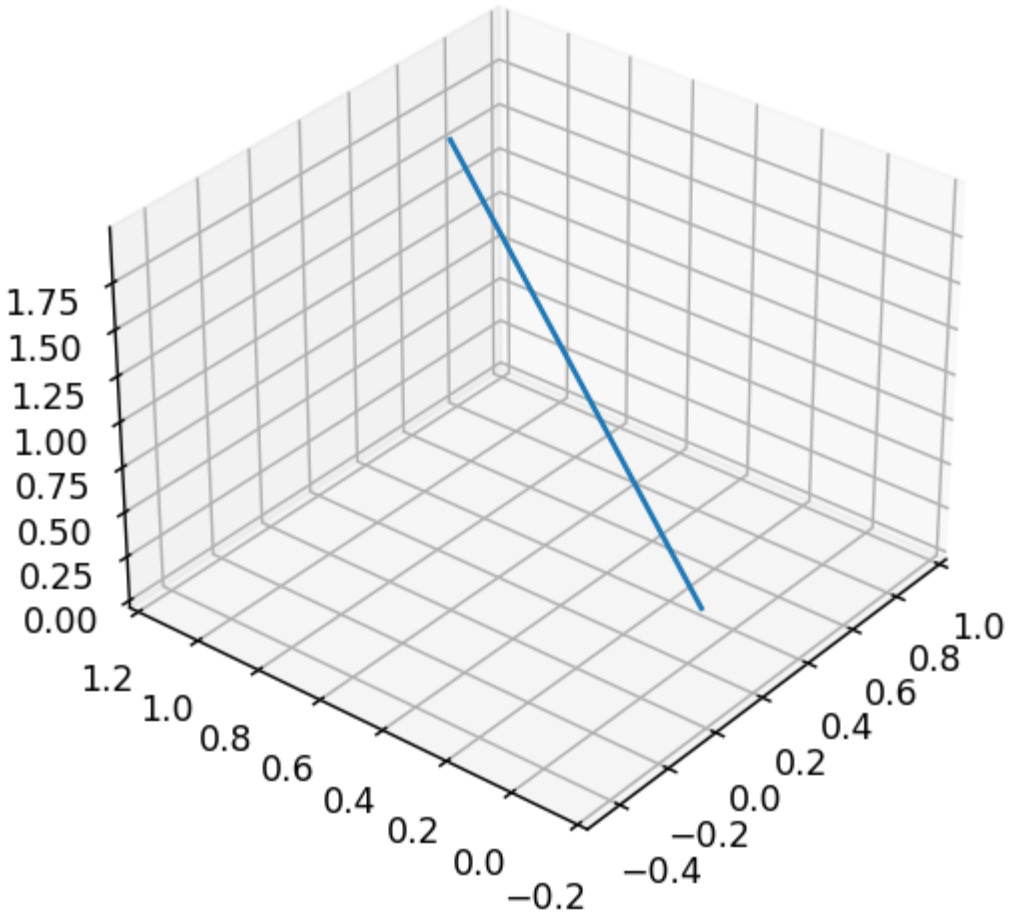
Finalmente, para representar las trayectorias, se ha añadido un bucle, que traza la trayectoria colocando cada punto tridimensional en una proyección 3D, esto se realiza para los Nt puntos calculados y cada cuerpo.

Para la parte de resultados, se ha decidido fijar un límite en los ejes x e y ya que va a ayudar a visualizar y comparar los resultados posteriores.

3. Discusión de Resultados

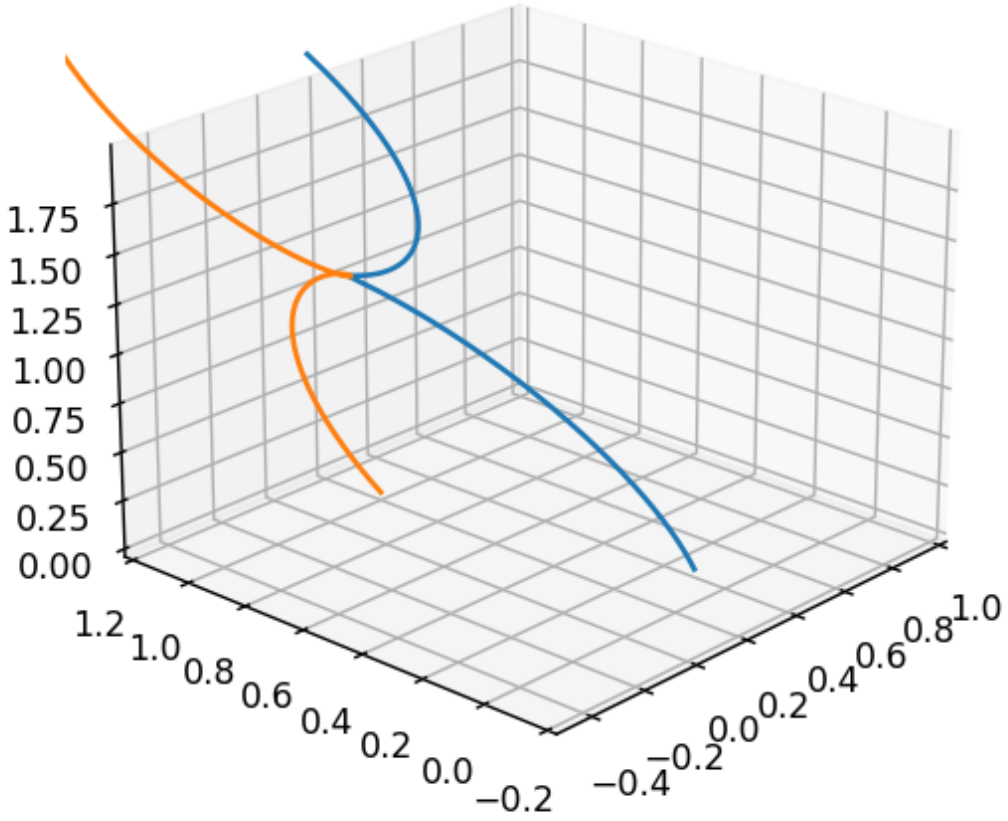
3.1. Caso de un cuerpo

Si se ejecuta el programa con el primer cuerpo de las condiciones iniciales propuestas, la trayectoria define una linea recta, ya que no existen atracciones gravitatorias externas, caso MRU.



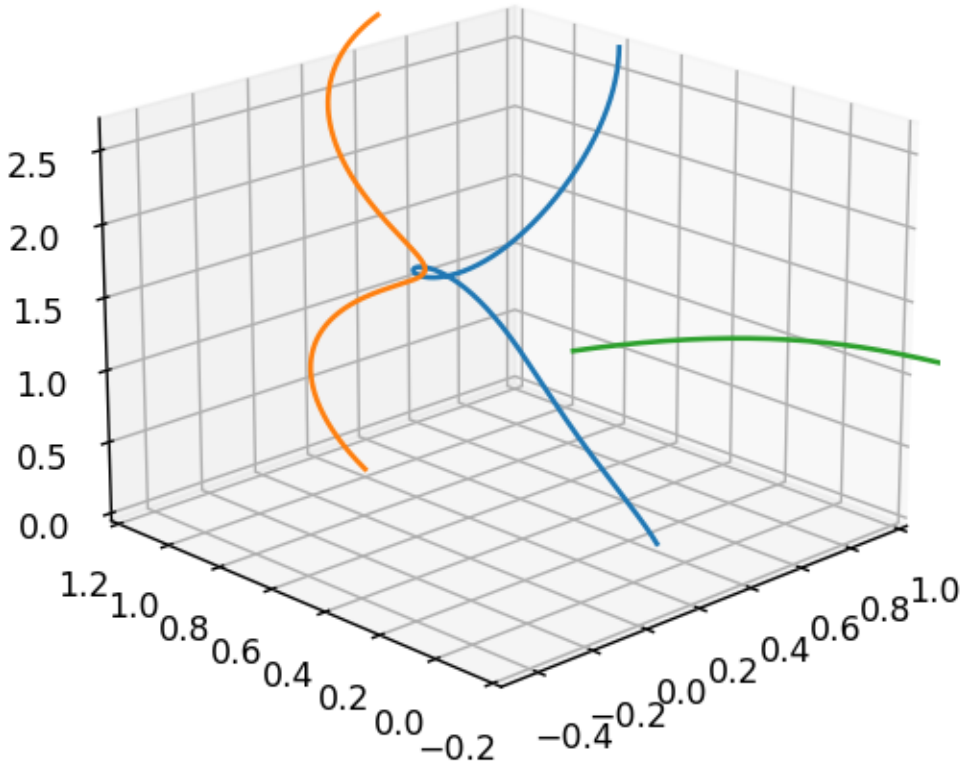
3.2. Caso de dos cuerpos

Si se añade el segundo cuerpo, ambos se atraen mutuamente y forman las siguientes trayectorias, si se aumenta el tiempo de integración, estos volverán a acercar.



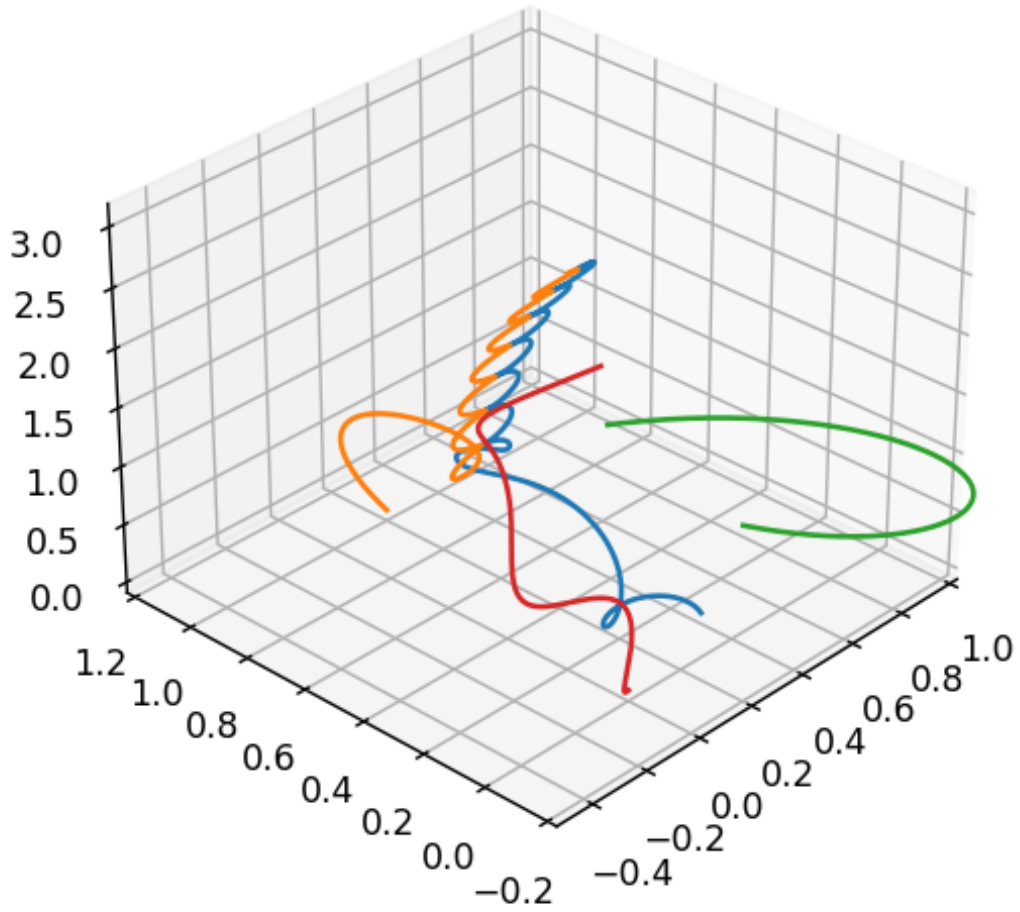
3.3. Caso de tres cuerpos

En este caso, al añadir un tercer cuerpo, la trayectoria anterior se endereza en el eje z desplazándose ambos cuerpos, los cuales siguen acercándose mucho en un punto, el tercer cuerpo, tiene una velocidad inicial lo suficientemente elevada como para alejarse de los dos primeros cuerpos.

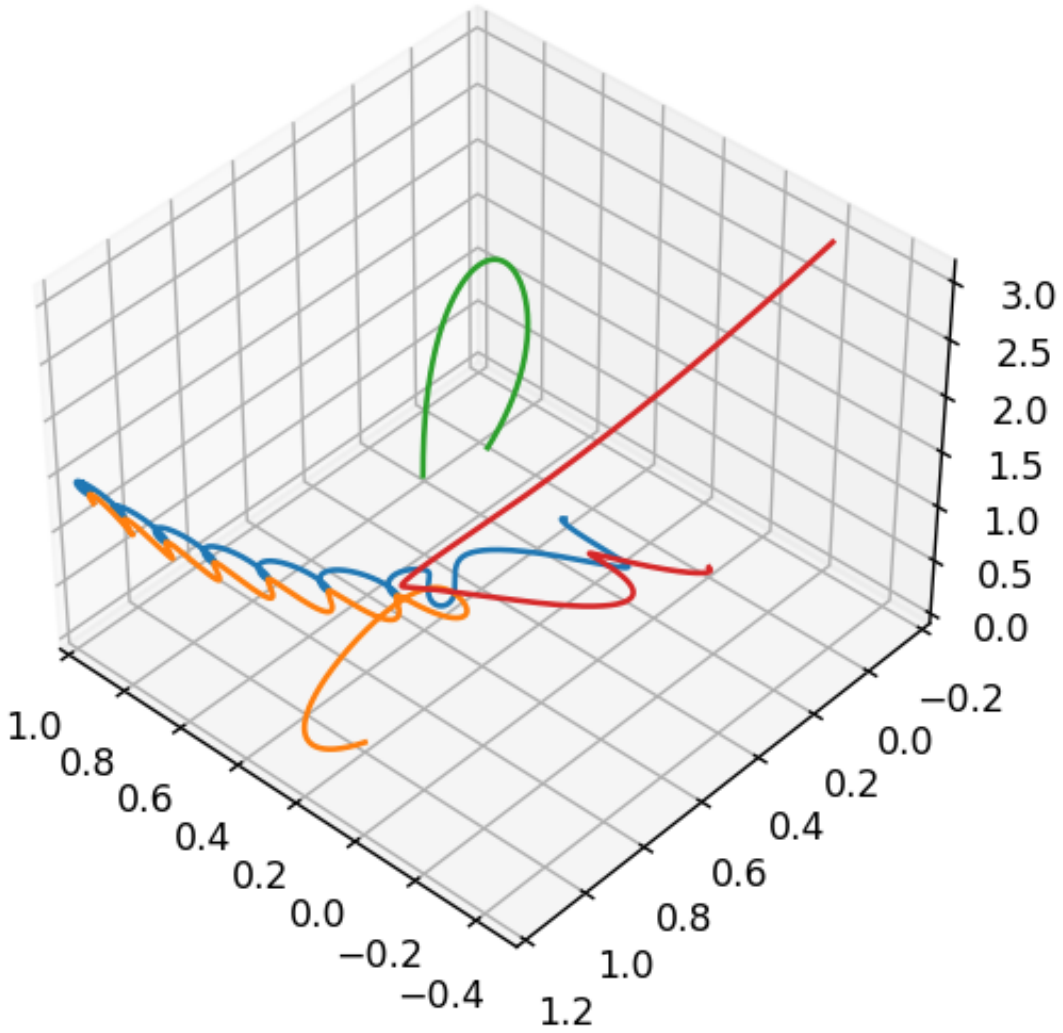


3.4. Caso de cuatro cuerpos

Al añadir el cuarto cuerpo (color rojo), complica más aun las trayectorias, en este caso, el cuerpo tres (color verde) no tiene suficiente velocidad para escapar del campo gravitacional del conjunto y se aprecia como regresa.

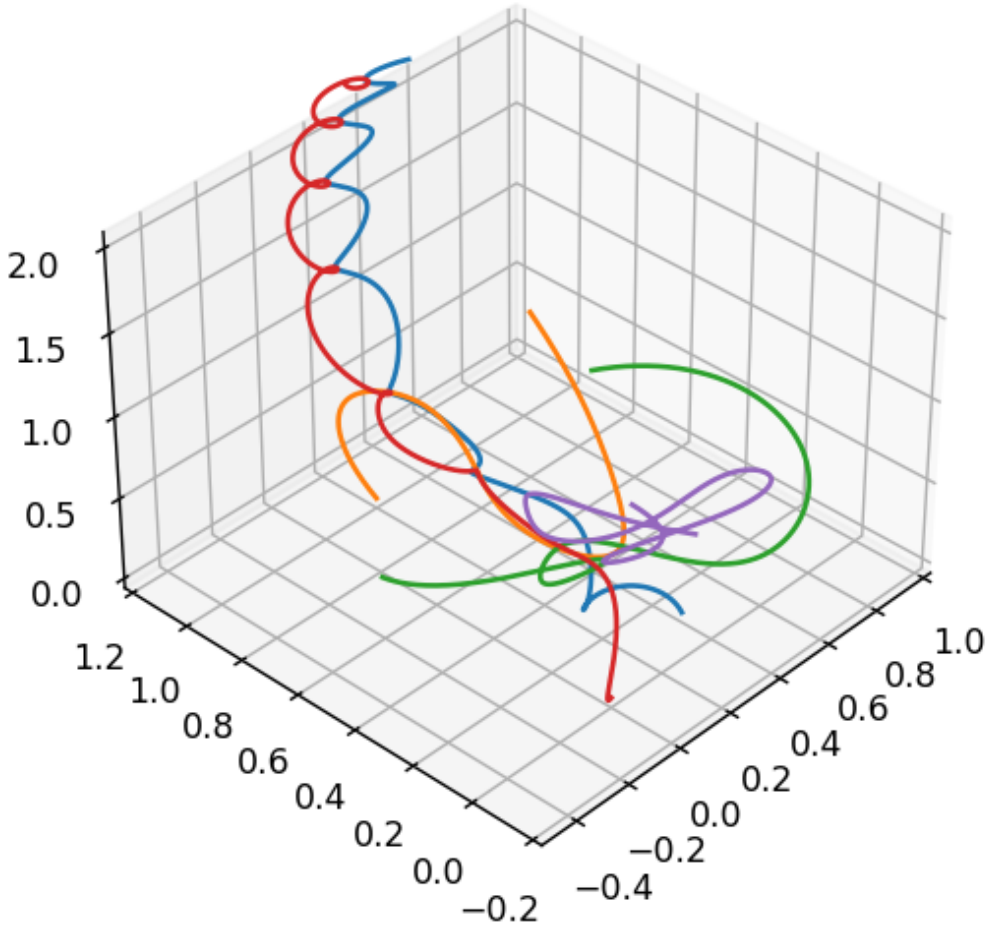


Si se observa desde otro ángulo, se puede ver que el cuarto cuerpo (color rojo) al inicio del problema se ve atraído por los cuerpos azul y naranja hasta que es .[ex]pulsadoz se aleja de ambos.

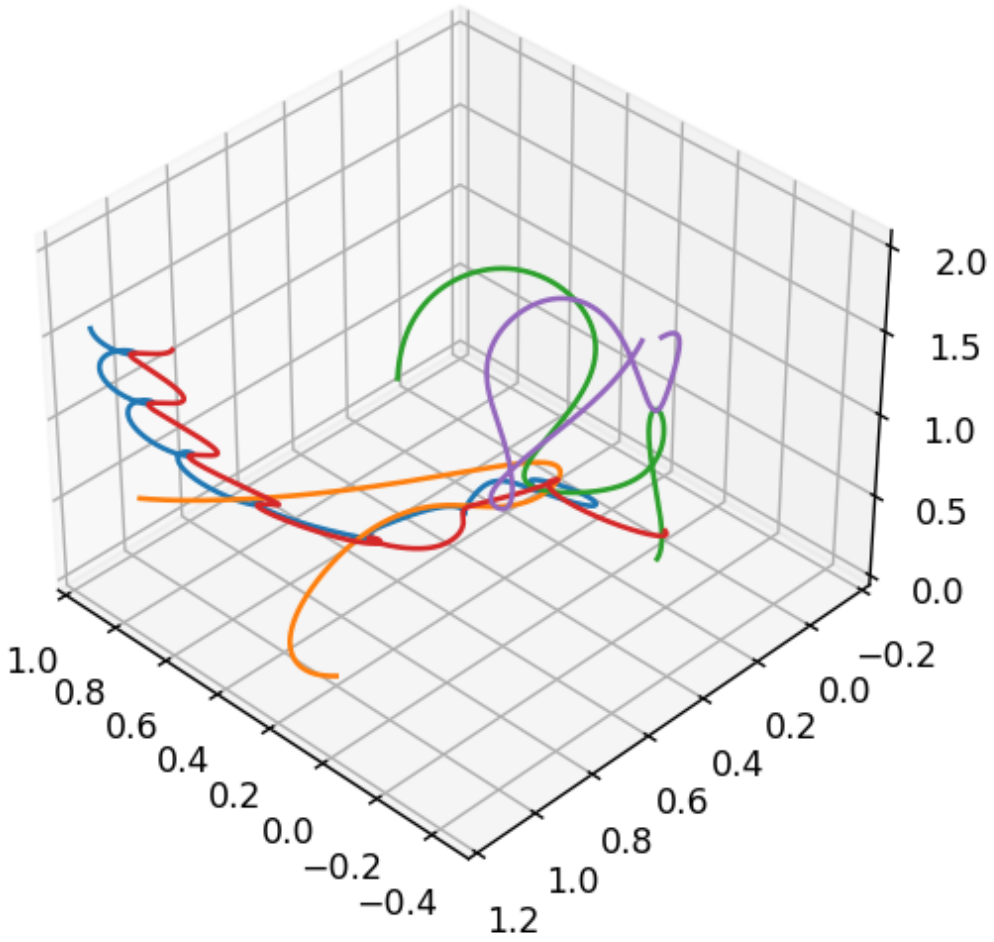

3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

3.5. Caso de cinco cuerpos

Al añadir un quinto cuerpo al sistema (color lila) se puede apreciar que en este caso, el cuerpo naranja no sigue al azul, si no que el cuerpo rojo, que en el caso de cuatro cuerpos, se alejó, esta vez ha a alejarse junto al cuerpo azul del resto de cuerpos.



En la segunda proyección se aprecia mejor las trayectorias del cuerpo verde, morado y naranja, las cuales son muy impredecibles y es difícil saber como van a continuar en el tiempo.


2.0
1.5
1.0
0.5
0.0
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2