


Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

MÁSTER UNIVERSITARIO EN SISTEMAS ESPACIALES

MILESTONE 5

Ampliación de Matemáticas I

1 de diciembre de 2022

Autor:
Alberto García Rincón

Índice

1. Introducción

En este trabajo se ha integrado el problema de los N cuerpos con métodos de resolución numérica.

Se ha realizado una demostración de la resolución del movimiento de los cuerpos a través de varias simulaciones con distintos números de cuerpos implicados.

2. Código de Python

2.1. Diagrama de bloques

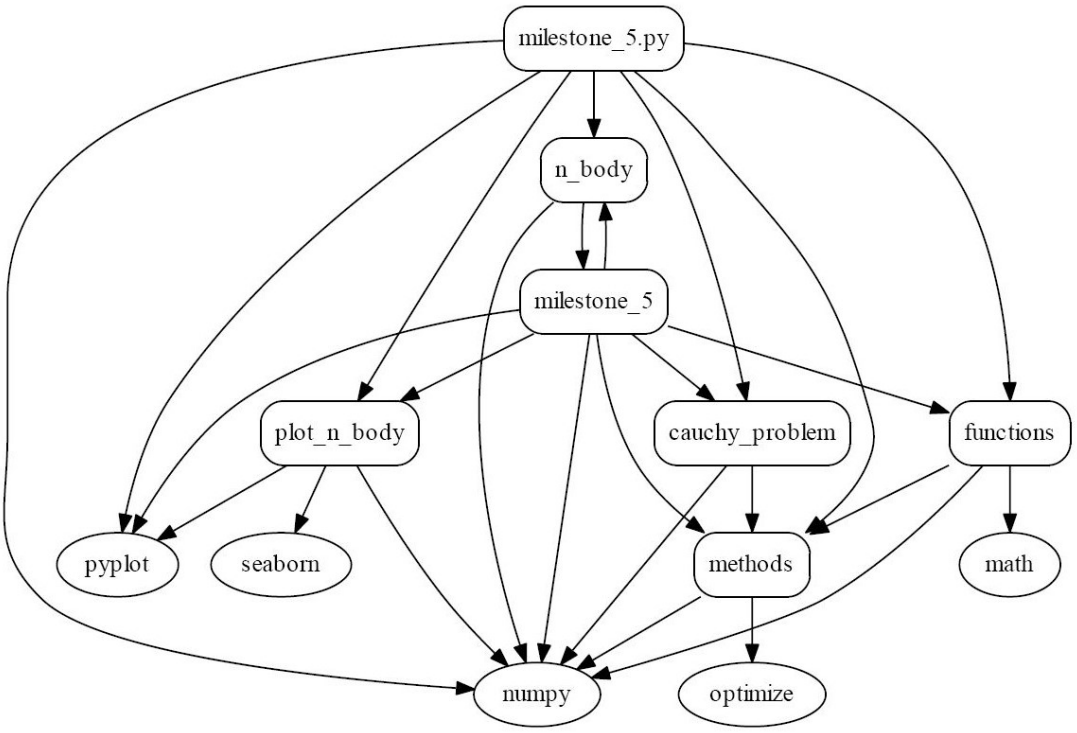

Figura 1: Diagrama módulos Milestone 5.

Como puede observarse en la Figura 1 para hacer funcionar este programa se debe ejecutar el programa principal denominado $milestone\_5.py$. Este programa se encarga de llamar al resto de funciones definidas en los respectivos archivos.

Se van a comentar las nuevas funciones implementadas para la resolución de este problema.

2.2. milestone_5.py

En el archivo principal se ha definido un código para iniciar las condiciones iniciales. Posteriormente se selecciona el método de resolución, para las simulaciones de ha elegido el método

de Runge-Kutta de orden 4, y se ejecuta el programa de cálculo numérico. Y por último, se ejecutan las funciones que se encargan de dibujar los resultados.

```
15  def main():
16
17      #Initialize values
18      Dt = 0.001   #[s]
19      tf = 5
20      N = int(tf/Dt)+1    #Nº steps
21      t = np.linspace(0, tf, N)
22
23      mets = [met.explicit_euler, met.runge_kutta4, met.inverse_euler, met.crank_nicolson, met.leap_frog]
24
25      #U0 = fn.init_state_nbody_rand(Nb, Nc) #create random init values
26      U0 = fn.init_state_nbody(Nb, Nc)
27      U = np.zeros([len(U0), N])
28
29      #Calculus
30      i = 1 #Runge-Kutta
31      U = cauchy_problem(U0, t, nb.n_body, mets[i])
32      plo_n.plot_3D(U, t, Dt, mets[i], Nb, Nc)
33      plo_n.plot_vel(U, t, Dt, mets[i], Nb, Nc)
```

Figura 2: Código principal Milestone 5.

2.3. functions.py

En esta función se definen las condiciones iniciales de posición y velocidad de los distintos cuerpos implicados en la simulación.

```python
#Initiate n_body problem conditions manually
def init_state_nbody(Nb, Nc):
    U0 =  np.zeros(2*Nc*Nb)
    U1  = np.reshape(U0, (Nb, Nc, 2))

    # body 1
    r1 = [1, 0, 0]
    v1 = [0, 0.5, 0]

    # body 2
    r2= [-1, 0, 0]
    v2 = [0, -0.5, 0]

    # body 3
    r3 = [0, 1, 0]
    v3 = [-0.5, 0, 0]

    # body 4
    r4 = [0, -1, 0]
    v4 = [0.5, 0, 0]

    r = np.concatenate(([r1], [r2], [r3], [r4]), axis=0)
    v = np.concatenate(([v1], [v2], [v3], [v4]), axis=0)

    for i in range(Nb):
        U1[i, :, 0] = r[i,:]
        U1[i, :, 1] = v[i,:]

    return U0
```

Figura 3: Código función definir condiciones iniciales.

2.4. plot_n_body.py

En el siguiente código se muestra la función encargada de dibujar en 3 dimensiones las trayectorias calculadas previamente.

Se ha definido un bucle que recorre todos los cuerpos del problema y dibuja en cada interacción la trayectoria asociada al cuerpo correspondiente. Destacar que se dibuja un punto que destaca sobre el resto en el punto final, para indicar la posición final del cuerpo una vez terminada la simulación.

```python
#3D plot position
def plot_3D(U, t, Dt, m, Nb, Nc): #(Values, time, Delta t, method, Nbodies, Ncoord)
    sns.set()
    plt.figure(figsize=(10,9))
    ax = plt.axes(projection='3d')
    col = ['b', 'r', 'g', 'y']

    for i in range(Nb):
        a = 2*Nc*i
        ax.plot3D(U[a,:], U[a+2,:], U[a+4,:], label='Body ' + str(i+1), color=col[i])
        ax.plot(U[a,len(t)-1], U[a+2,len(t)-1], U[a+4,len(t)-1], color=col[i], marker='o', markersize=8)

    ax.set_title('Positions in 3D, ' + m.__name__ + ' method')
    ax.set_xlabel('X [m]')
    ax.set_ylabel('Y [m]')
    ax.set_zlabel('Z [m]')
    ax.legend()
```

Figura 4: Código función dibujar trayectorias cuerpos en 3D.

Con el siguiente código se permite dibujar el módulo de velocidad a lo largo del tiempo de simulación. Dentro de esta misma función y por simplificar el código, se ha calculado dicho módulo a partir del vector de estados del sistema que evoluciona a través del tiempo.

```python
def plot_vel(U, t, Dt, m, Nb, Nc):
    sns.set()
    plt.figure()

    for i in range(Nb):
        a = 2*Nc*i
        v = [U[a+1,:], U[a+3,:], U[a+5,:]]
        uu = np.empty(len(t))

        for k in range(len(t)):
            uu[k] = np.linalg.norm([v[0][k], v[1][k], v[2][k]])

        plt.plot(t, uu, label='Body ' + str(i+1))

    plt.title('Velocity modules, ' + m.__name__ + ' method')
    plt.xlabel('Time [s]')
    plt.ylabel('Velocity [m/s]')
    plt.legend()
```

Figura 5: Código función dibujar módulos velocidades de los cuerpos.

2.5. n_body.py

En el siguiente código se muestra la función que define el problema de los N cuerpos.

```python
def n_body(U, t):
    Us  = np.reshape(U, (Nb, Nc, 2))
    F =  np.zeros(len(U))
    dUs = np.reshape(F, (Nb, Nc, 2))

    r = np.reshape(Us[:, :, 0], (Nb, Nc))   #pos
    v = np.reshape(Us[:, :, 1], (Nb, Nc))   #vel

    drdt = np.reshape(dUs[:, :, 0], (Nb, Nc))   #vel
    dvdt = np.reshape(dUs[:, :, 1], (Nb, Nc))   #accel

    dvdt[:,:] = 0  #WARNING

    for i in range(Nb):
        drdt[i,:] = v[i,:]
        for j in range(Nb):
            if j != i:
                d = r[j,:] - r[i,:]
                dvdt[i,:] = dvdt[i,:] +  d[:] / np.linalg.norm(d)**3

    return F
```

Figura 6: Código función problema de los n-cuerpos.

3. Resultados y análisis

Se han realizado un total de 3 simulaciones con distinto número de cuerpos implicados. El análisis de los resultados se muestra a continuación.

Para todas las simulaciones se ha utilizado el método de resolución numérico de Runge-Kutta de orden 4, por ser uno de los más precisos dentro de los métodos de integración que se han desarrollado en los trabajos anteriores.

Se ha elegido un $\Delta t = 0{,}001$ [s] para todas las simulaciones y un tiempo de 5 segundos. Considerando que este tiempo es suficiente para obtener unos resultados decentes y visualmente aptos para analizar los resultados.

3.1. 2 cuerpos

En la Figura 7 se pueden observar las trayectorias de los 2 cuerpos implicados en la simulación en un escenario en 3 dimensiones.

Se han pintado los 3 ejes cartesianos y el punto que destaca sobre el resto en cada una de las trayectorias indica la posición del cuerpo en el instante final de la simulación.

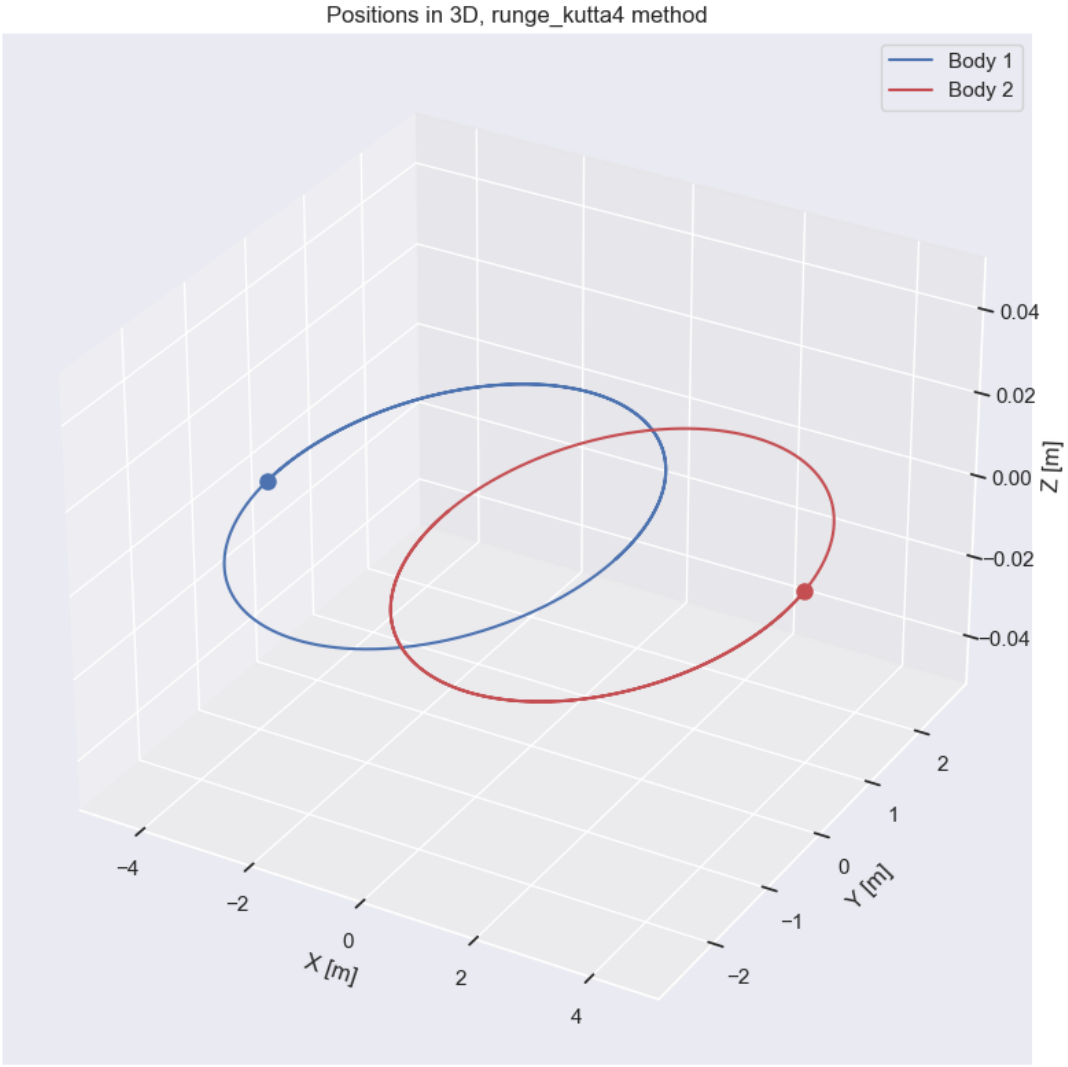

Figura 7: Trayectorias 2 cuerpos.

En la Figura 8 se muestra el módulo de velocidades de cada uno de los cuerpos partícipes a lo largo del tiempo. Para la obtención de este módulo se ha utilizado una función definida en el código de la Figura 5.

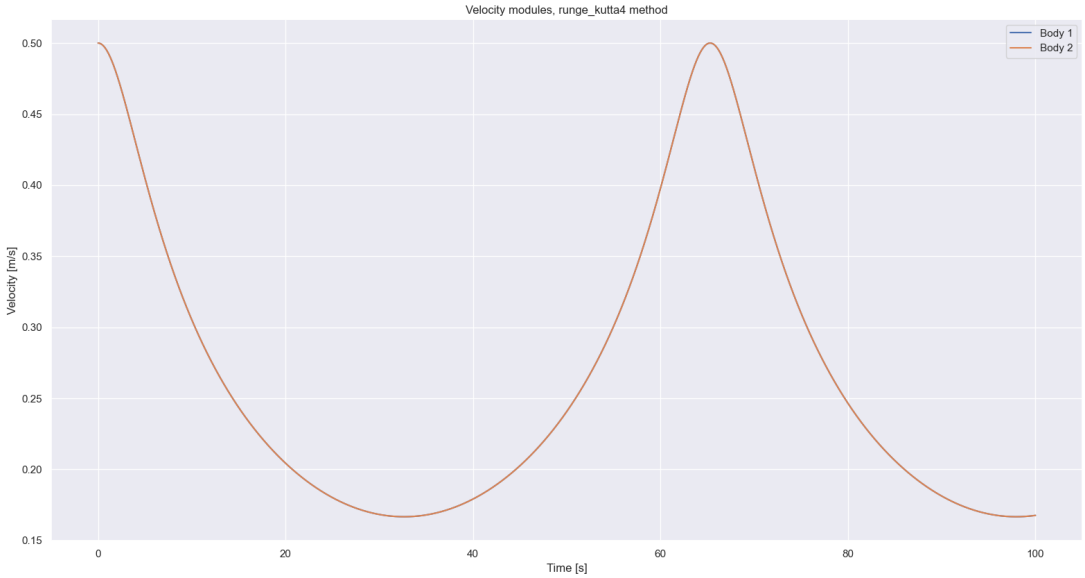

Figura 8: Perfil de velocidades 2 cuerpos.

3.2. 3 cuerpos

En la Figura 9 se muestran las trayectorias que siguen los 3 cuerpos definidos para esta simulación.

Como se puede apreciar, se observan como se afectan entre ellos los cuerpos y como modifican su trayectoria en función de la presencia de otro cuerpo cercano a él.

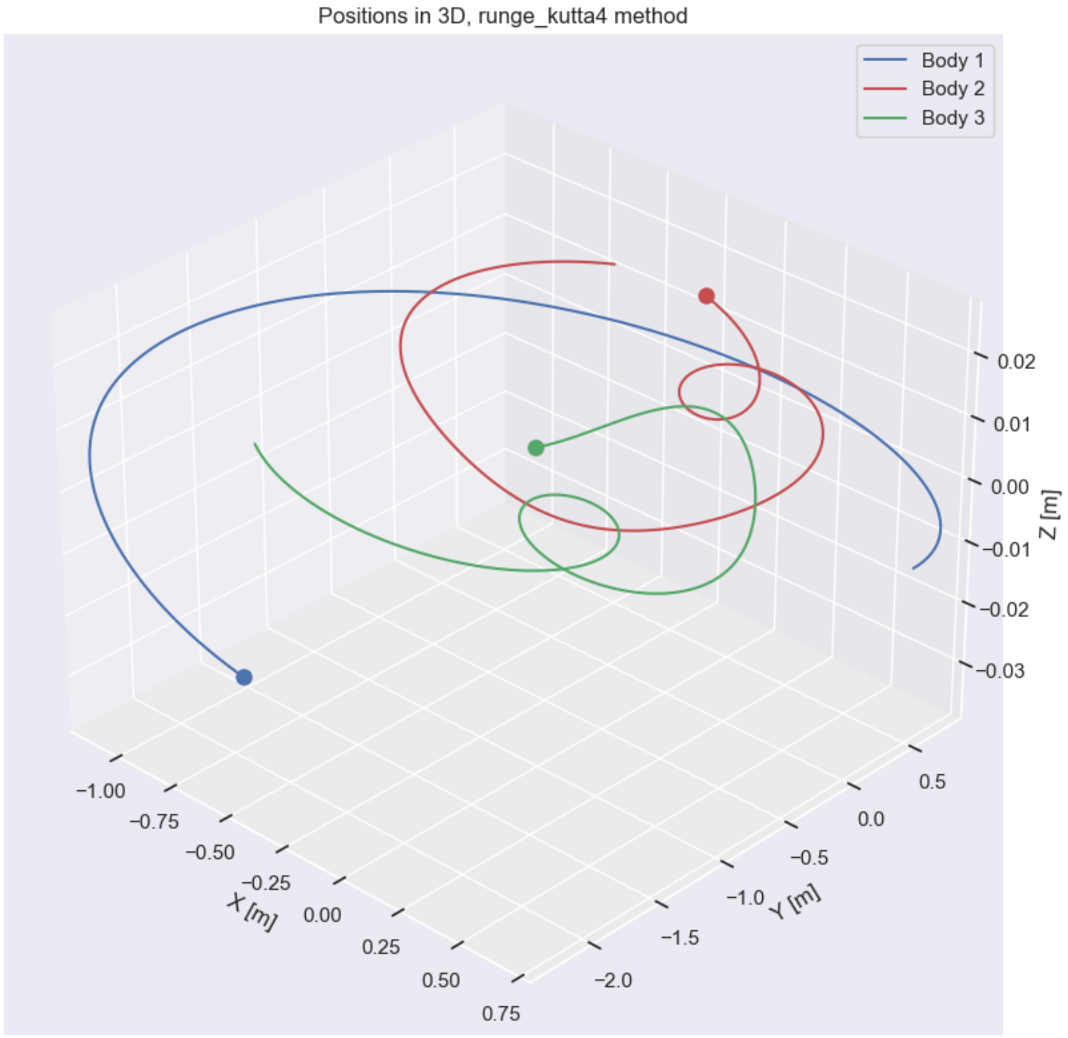

Figura 9: Trayectorias 3 cuerpos.

En la Figura 10 se observa la variación de la velocidad absoluta de cada cuerpo en función del tiempo. Se puede apreciar más en detalle como los cambios de trayectoria afectan a la velocidad de cada cuerpo y como esta se modifica.

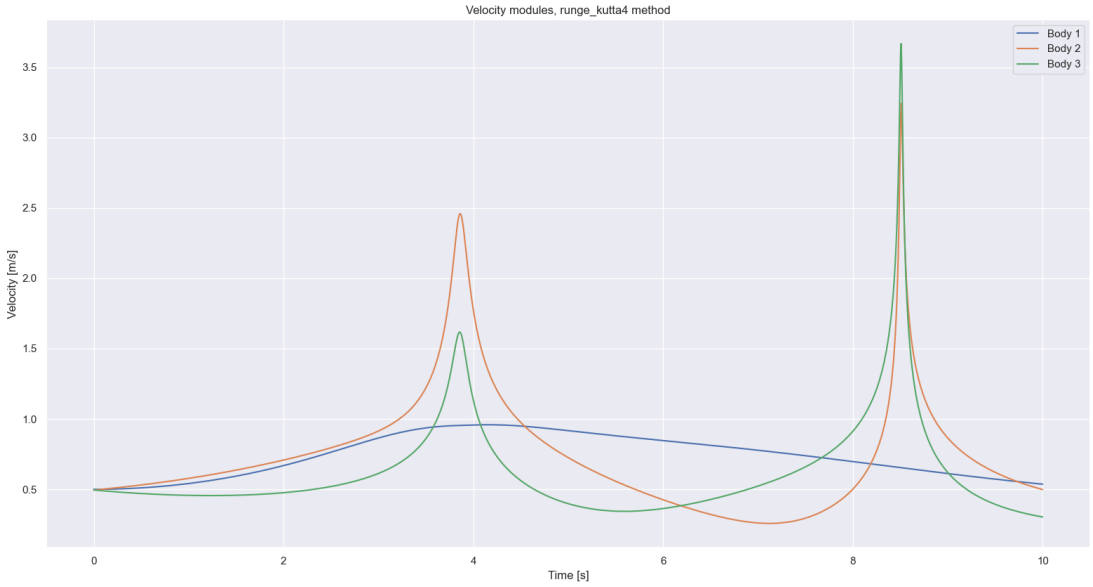

Figura 10: Perfil de velocidades 3 cuerpos.

3.3. 4 cuerpos

Se han realizado dos simulaciones con 4 cuerpos implicados en la simulación con condiciones iniciales distintas de velocidad y de posición.

En la primera simulación (Figuras 11 y 12) con 4 cuerpos se puede observar que el movimiento de los cuerpos es mucho más aleatorio, pero aún así se puede apreciar como se afectan entre sí los cuerpos y como se provoca un cambio de trayectoria y velocidad cada vez que esto ocurre.

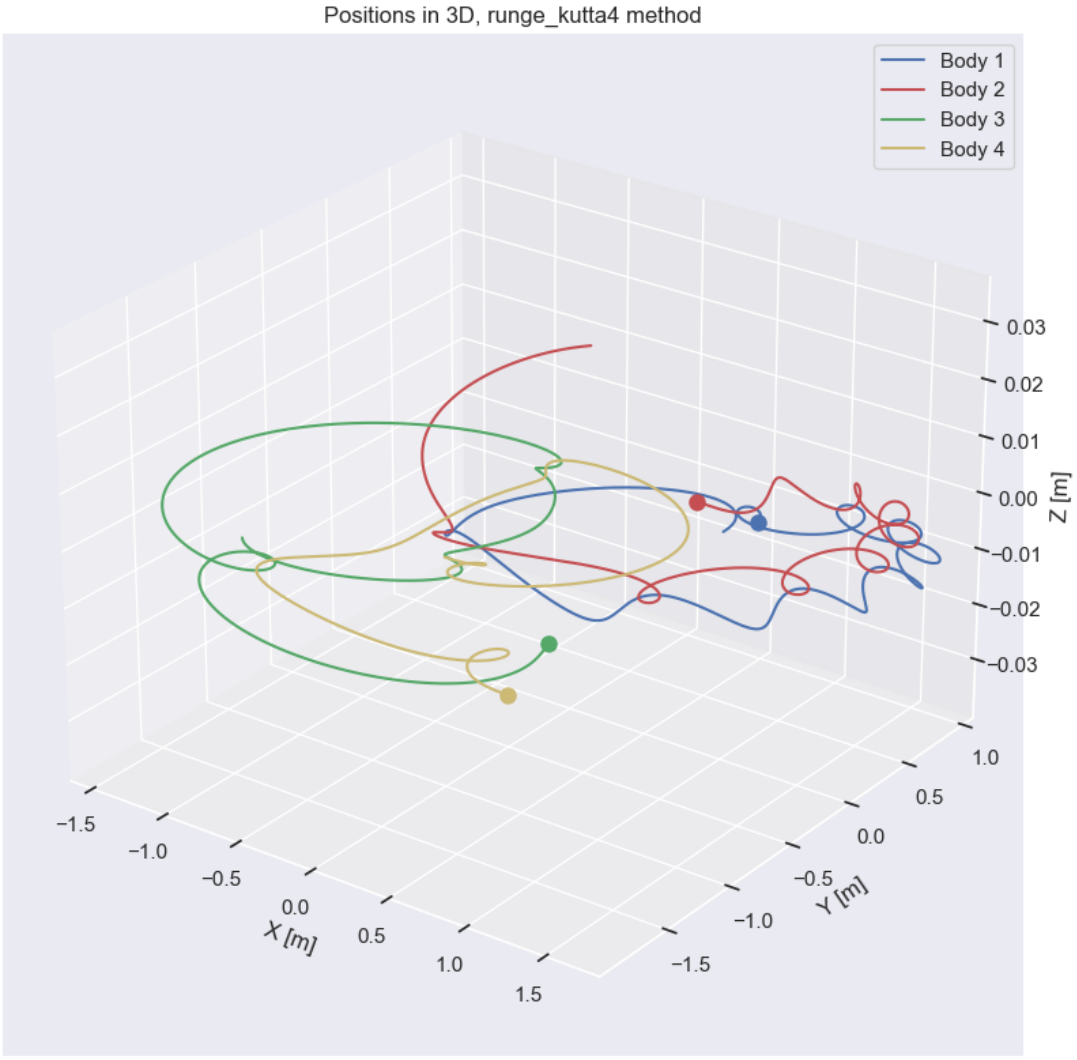

Figura 11: Trayectorias 4 cuerpos.

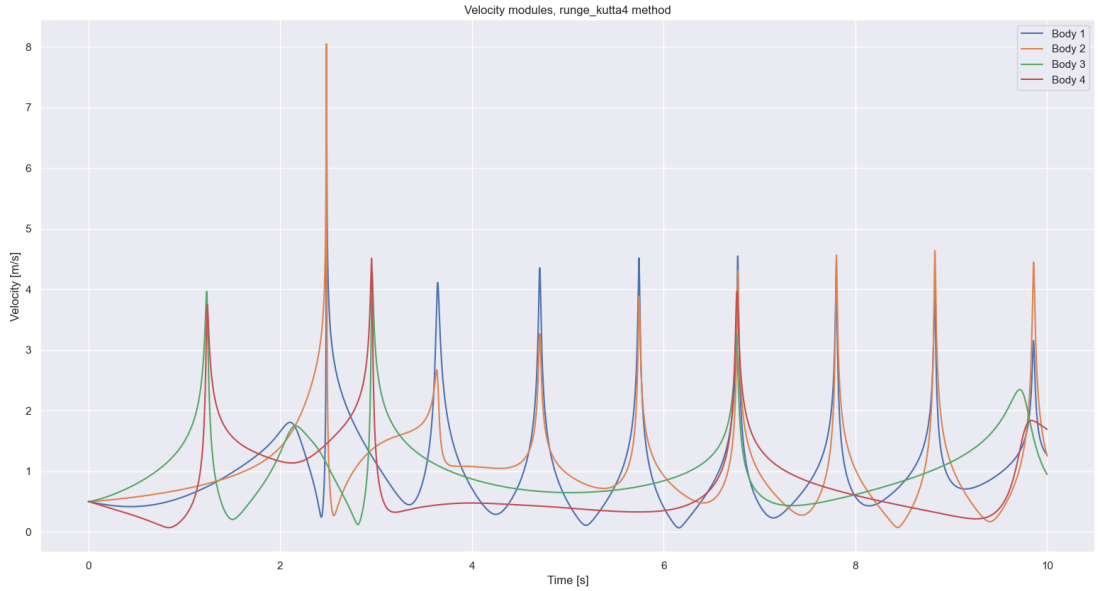

Figura 12: Perfil de velocidades 4 cuerpos.

En esta segunda simulación (Figuras 13 y 14) se han introducido unas condiciones iniciales propensas para que a lo largo de la simulación se dibujen 4 órbitas que interactúan entre sí. El resultado se muestra en la figura siguiente, se muestra las trayectorias en 3D y la evolución de la velocidad de cada cuerpo a lo largo del tiempo de la simulación.

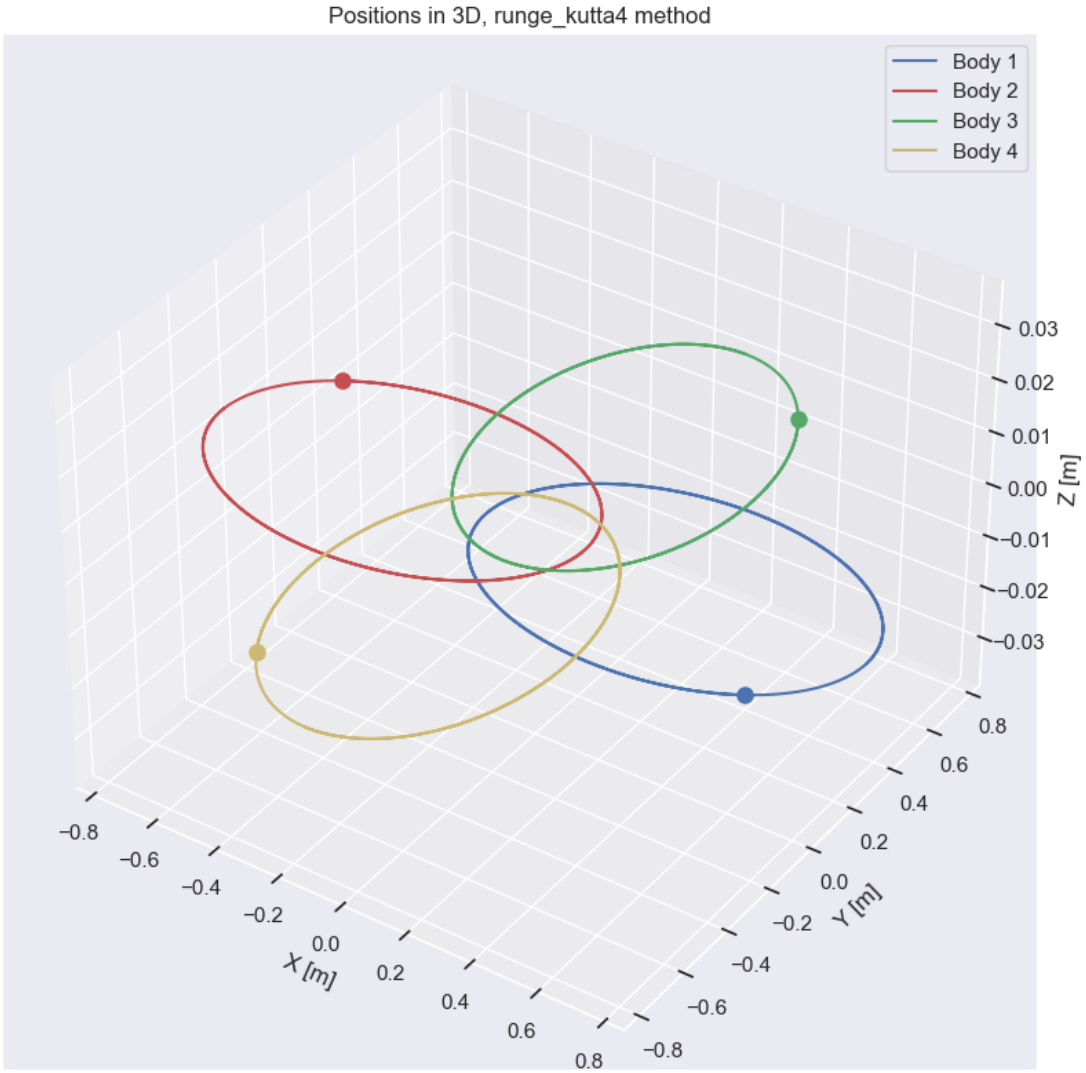

Figura 13: Trayectorias 4 cuerpos.

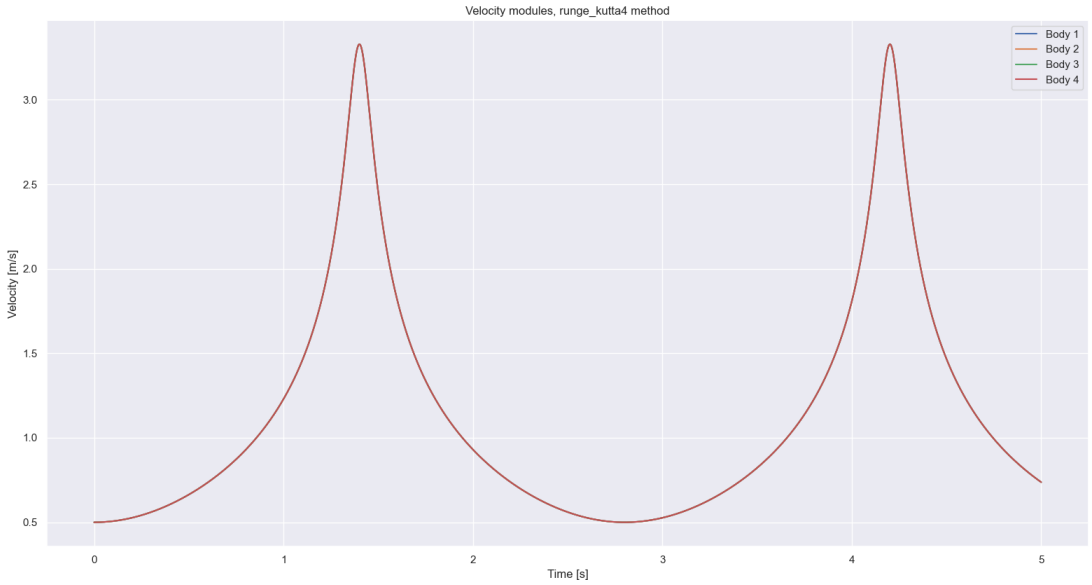

Figura 14: Perfil de velocidades 4 cuerpos.

4. Conclusiones

El iniciar los cuerpos con unas condiciones iniciales de velocidad en cada uno de los ejes ha supuesto tener que iterar con varias simulaciones hasta conseguir una simulación con unos resultados aceptables visualmente y que no convergieran todos los cuerpos al mismo punto para luego divergir.