Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestone 5

Ampliación de Matemáticas I

Javier Garrido Castillo

8 de diciembre de 2022

Índice

1. Introducción

En este hito 5, se procederá a estudiar el problema de los N cuerpos donde se analizará la evolución temporal de N masas puntuales que ejercen fuerzas gravitatorias entre sí. Para ello se definirá el problema estudiado, la presentación de los resultados obtenidos y por último, se mostrará el código implementado para resolverlo.

2. Problema

Para resolver problema de los N cuerpos, se deben calcular las fuerzas gravitatorias debidas a las masas de los distintos cuerpos de la siguiente forma:

$$m_i \frac{d\vec{v_i}}{dt} = \sum_{i=1, i\neq j}^{N} m_i m_j \frac{(\vec{r_j} - \vec{r_i})}{\|\vec{r_j} - \vec{r_i}\|^3} \tag{1}$$

Para su resolución, hemos supuesto que todos los cuerpos poseen la misma masa y se debe tener en cuenta que $\vec{v_i} = \frac{d\vec{r_i}}{dt}$. De esta forma, se obtiene una ecuación diferencial que puede resolverse a partir del problema de *Cauchy* con un esquema numérico de los empleados en hitos anteriores. Como solución, se obtendrá un vector U donde se almacenarán tanto las velocidades como posiciones de los N cuerpos para cada paso del tiempo.

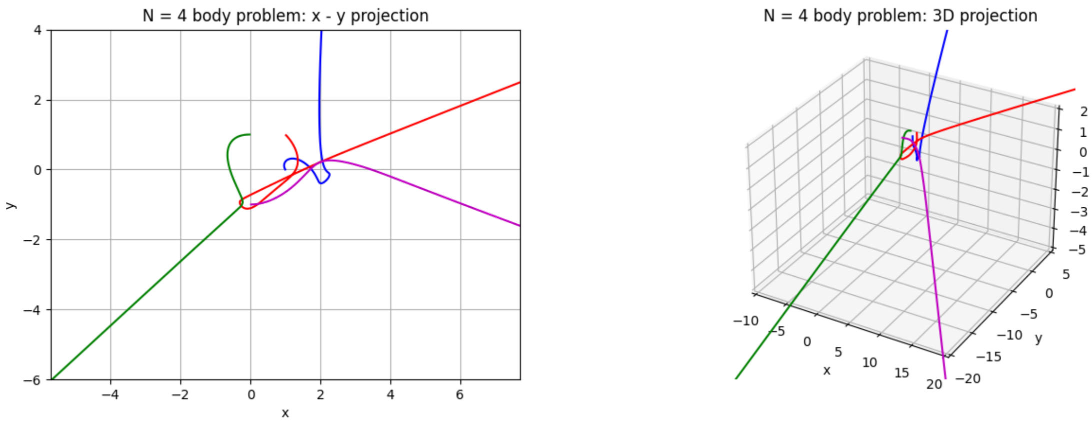
En este caso, se ha empleado el esquema numérico *Leap Frog* y un máximo de 4 cuerpos con las siguientes condiciones iniciales para cada uno de ellos:

N	**1**	**2**	**3**	**4**
$\vec{r_i}$	(1, 0, 0)	(1, 1, 0)	(0, 1, 0)	(0, -1, 0)
$\vec{v_i}$	(0, 0, -1)	(1, -1, -1)	(-1, 0, 0)	(1, 0, 0)

Tabla 1: Condiciones iniciales de los N cuerpos

3. Resultados

Empleando *Leap Frog* como esquema númerico para resolver el problema con 10.000 pasos de tiempo y un tiempo total de 200 segundos, se obtienen los siguientes resultados:

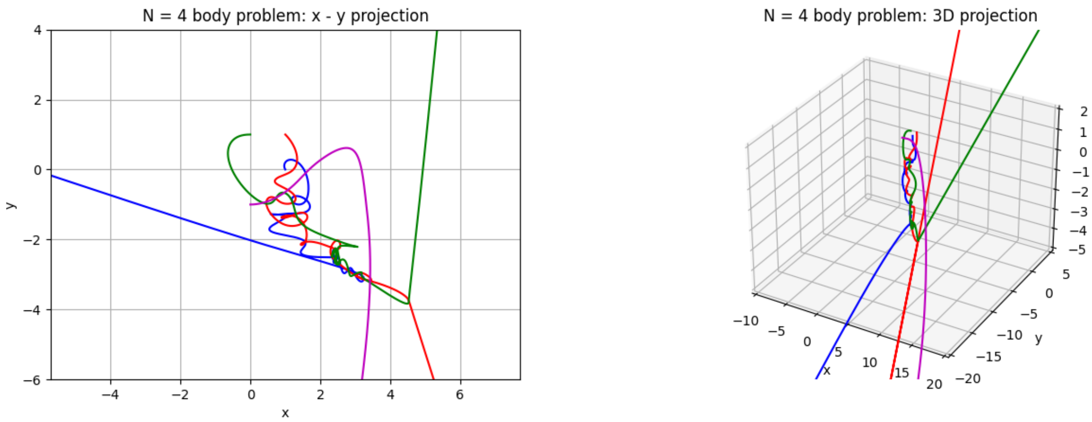


(a) Proyección en plano X-Y

(b) Proyección 3D

Figura 1: Resultados con *Leap Frog*

Del mismo modo, empleando *Runge-Kutta* de orden 4 como esquema númerico para resolver el problema con 10.000 pasos de tiempo y un tiempo total de 200 segundos, se obtienen los siguientes resultados:



(a) Proyección en plano X-Y

(b) Proyección 3D

Figura 2: Resultados con *Runge-Kutta* de orden 4

4. Código

Empleando de nuevo la estructura Top-Down seguida en el resto de hitos de la asignatura, se emplea nuevamente la carpeta *Functions* donde se ordenarán las distintas funciones empleadas en este hito. Se emplean de nuevo los archivos de *Temporal_Schemes* y *Cauchy_Problem*. Adicionalmente, se ha definido un nuevo archivo denominado N_{Body} donde se definirá la función del problema de los N cuerpos.

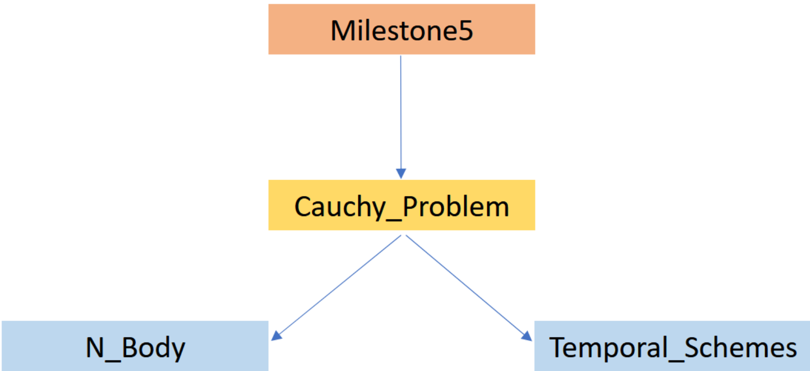


Figura 3: Estructura Top-Down del problema

4.1. N_Body

```
from numpy import zeros, reshape
from numpy.linalg import norm


Nb = 4 # n   cuerpos
Nc = 3 # n   coord


# U --> vector estado
# r,v --> posicion, velocidad
# dr(i)/dt = v(i)
# dv(i)/dt = sum(j) [G*m(j)*(r(j) - r(i)) / |r(j) - r(i)|^3]

def F_NBody(U, t):

    U_sol = reshape(U,(Nb,2,Nc)) #Reshape de la U: dividir en arrays para cada cuerpo (i), cada cuerpo tiene en fila 0 la r(i) y en fila 1 la v(i)
    F = zeros(len(U))
    dU_sol = reshape(F, (Nb, 2, Nc)) # Los valores introducidos en dU_sol ser n insertados en F

    r = reshape(U_sol[:, 0, :], (Nb, Nc)) # Guarda en array posiciones cada uno de los r(i) --> POSICIONES
    v = reshape(U_sol[:, 1, :], (Nb, Nc)) # Guarda en array velocidades cada uno de los v(i) --> VELOCIDADES
```

```
    drdt = reshape(dU_sol[:, 0, :], (Nb, Nc)) # Guarda en array derivada
    posiciones cada uno de los dr(i)/dt --> VELOCIDADES
    dvdt = reshape(dU_sol[:, 1, :], (Nb, Nc)) # Guarda en array derivada
    velocidades cada uno de los dv(i)/dt --> ACELERACIONES

    #dvdt[:,:] = 0

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

    return F
```

Listing 1: N$_{B}$*odycode*

4.2. Código principal: Milestone5

```
from Functions.Temporal_Schemes import Euler, Crank_Nicolson, RK4,
    Inverse_Euler, LeapFrog
from Functions.Cauchy_Problem import Cauchy_problem
from Functions.N_Body import F_NBody
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from numpy import array, linspace, zeros, size, linspace, reshape

N = 10000 # time steps
t0 = 0 # timepo inicial
tf = 200 # tiempo final
dt = [0.1, 0.01, 0.001]
#N = int(tf/dt[j])
t = linspace(t0, tf, N)

Nb = 4 # n  cuerpos
Nc = 3 # n  coord

colors = ['b','r','g','m','y','c']

# CONDICIONES INICIALES

U0 = zeros(Nb*2*Nc)
U_0 = reshape(U0, (Nb, 2, Nc) )
r0 = reshape(U_0[:, 0, :], (Nb, Nc))
v0 = reshape(U_0[:, 1, :], (Nb, Nc))

# body 1
r0[0,:] = [ 1, 0, 0]
v0[0,:] = [ 0, 0, -1]

# body 2
r0[1,:] = [ 1, 1, 0]
v0[1,:] = [ 1, -1, -1]
```

```
# body 3
r0[2, :] = [ 0, 1, 0 ]
v0[2, :] = [ -1, 0., 0. ]

# body 4
r0[3, :] = [ 0, -1, 0 ]
v0[3, :] = [ 1, 0., 0. ]

# SOLUTION

U = Cauchy_problem(RK4, F_NBody, t, U0)

U_s  = reshape( U, (N, Nb, 2, Nc) )
r    = reshape( U_s[:, :, 0, :], (N, Nb, Nc) )


# 2D PLOT
for i in range(Nb):
    plt.figure(1)
    plt.plot(r[:, i, 0], r[:, i, 1], colors[i])
plt.title(f'N = {Nb} body problem: x - y projection')
plt.xlabel("x")
plt.ylabel("y")
plt.axis('equal')
plt.xlim((-4,6))
plt.ylim((-6,4))
plt.grid(True)
plt.show()


# 3D PLOT
fig = plt.figure(2)
ax = fig.add_subplot(projection='3d')
ax.plot(r[:, 0, 0], r[:, 0, 1], r[:, 0, 2], colors[0])
ax.plot(r[:, 1, 0], r[:, 1, 1], r[:, 1, 2], colors[1])
ax.plot(r[:, 2, 0], r[:, 2, 1], r[:, 2, 2], colors[2])
ax.plot(r[:, 3, 0], r[:, 3, 1], r[:, 3, 2], colors[3])
plt.title(f'N = {Nb} body problem: 3D projection')
plt.xlabel("x")
plt.ylabel("y")
plt.axis('equal')
ax.set_xlim3d(-10, 20) # viewrange for z-axis should be [-4,4]
ax.set_ylim3d(-20,5) # viewrange for y-axis should be [-2,2]
ax.set_zlim3d(-5, 2) # viewrange for x-axis should be [-2,2]
plt.show()
```

Listing 2: Milestone5 code

5. Conclusión

La evolución y mejora de este problema sería integrar en él las masas de cada uno de los cuerpos involucrados en el mismo. Esto proporcionaría mejor precisión al problema y le dotaría de una mayor complejidad. Esto será realizado por mi grupo de trabajo en el hito 7, donde paralelizaremos el cálculo de cada una de las posiciones y velocidades para cada uno de los cuerpos empleando la GPU. De esta forma, se podrá resolver el problema para un mayor número de cuerpos y de forma mucho más rápida al realizar los cálculos de forma simultánea.