


Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestones 5 & 6

Ampliación de Matemáticas I

26 de noviembre de 2022

Autor:

- Sergio Cavia Fraile

Índice

1. Introducción

El objetivo del milestone 5 será simular el problema de los N cuerpos y comentar los resultados de distintas condiciones iniciales.

El objetivo del milestone 6 será

2. Ecuaciones

2.1. El problema de los N cuerpos

La fórmula que rige el problema de los N cuerpos es la siguiente:

$$m_i \frac{\delta^2 r_i}{\delta t^2} = \sum_{j=1, j\neq i}^{N} \frac{G m_i m_j (r_j - r_i)}{\parallel r_j - r_i \parallel^3}$$

Por simplicidad tomaremos todas las masas y la constante gravitacional iguales a la unidad, de tal forma que la fórmula se convierte en:

$$\frac{\delta^2 r_i}{\delta t^2} = \sum_{j=1, j\neq i}^{N} \frac{(r_j - r_i)}{\parallel r_j - r_i \parallel^3}$$

2.2. Puntos de Lagrange

3. Código

3.1. Problema de los N cuerpos

El problema de los N cuerpos se ha implementado mediante las siguientes líneas de código:

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

def NBody(U, t, Nb, Nc):
    F =  zeros(len(U))
    Us  = reshape(U,(Nb, Nc, 2))
    dUs = reshape(F,(Nb, Nc, 2))

    r = reshape(Us[:,:,0],(Nb, Nc))
    v = reshape(Us[:,:,1],(Nb, Nc))
    dr = reshape(dUs[:,:,0], (Nb,Nc))
    dv = reshape(dUs[:,:,1], (Nb,Nc))

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

    return F
```

Donde simplemente utilizamos la función reshape para facilitarnos el trabajo y creamos dos bucles anidados en los que se dan valores a la velocidad y aceleración según las fórmulas que ya comentamos antes.

3.2. Código final N cuerpos

El código que realmente integra el problema de los N cuerpos es el siguiente:

Primero defino una función que permite introducir las condiciones iniciales (así como el número de cuerpos y de coordenadas por cuerpo) de una forma cómoda. Para ello, sitúo la función en el propio programa principal para que no sea necesario buscar el módulo que contiene las condiciones iniciales cada vez que se quieran cambiar.

```
def Init():

    #Función para dar las condiciones iniciales.
    #Definir nº coordenadas, cuerpos y velocidad y posición inicial de cada cuerpo

    Nc = 3
    Nb = 3

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

    #Cuerpo 1
    r0[0,:] = [0.5, -0.5, 0]
    v0[0,:] = [0.1, 0.1, 0.3]
    #Cuerpo 2
    r0[1,:] = [0, 0, 0]
    v0[1,:] = [-0.1, -0.1, 0.5]
    #Cuerpo 3
    r0[2,:] = [-0.4, 0.5, 0]
    v0[2,:] = [-0.35, -0.3, 0.3]

    return U0, Nc, Nb
```

Posteriormente se definen los tiempos y pasos en los que se integrará y se llama a la función de condiciones iniciales. Tras esto se llama a Cauchy, creando primero una función de paja que permita que Cauchy lea la función del problema de los N cuerpos pese a contar con más argumentos de entrada de los que Cauchy tenía en cuenta.

```
#Defino tiempo final, numero divisiones y condiciones iniciales
tf = 4
N = 2000
t = linspace(0, tf, N+1)
U0, Nc, Nb = Init()

# Llamo a Cauchy para calcular la función del problema de los N cuerpos pese a que tiene más inputs que (U, t)
def F(U, t):
   return NBody(U, t, Nb, Nc)
U = Cauchy(F, t, U0, RK4)
```

Finalmente se grafican los resultados en 3D y en el plano XY con las siguientes líneas de código.

```
#Grafico 3D
r   = reshape(U,(N+1, Nb, Nc, 2))[:,:,:,0]
fig = plt.figure()
ax1 = fig.add_subplot(111,projection='3d')
col = ["blue","red","green","purple","yellow","orange","black"]
for i in range(Nb):
    ax1.plot_wireframe(r[:,i,0].reshape((-1, 1)), r[:,i,1].reshape((-1, 1)), r[:,i,2].reshape((-1, 1)), color = col[i])
plt.title("N-body problem")
ax1.set_xlabel("x")
ax1.set_ylabel("y")
ax1.set_zlabel("z")
plt.grid()
plt.show()
#Grafico 2D
for i in range(Nb):
   plt.plot(r[:, i, 0], r[:, i, 1], color = col[i])
plt.axis('equal')
plt.title("N-body problem X-Y Plane")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.show()
```

3.3. Puntos de Lagrange

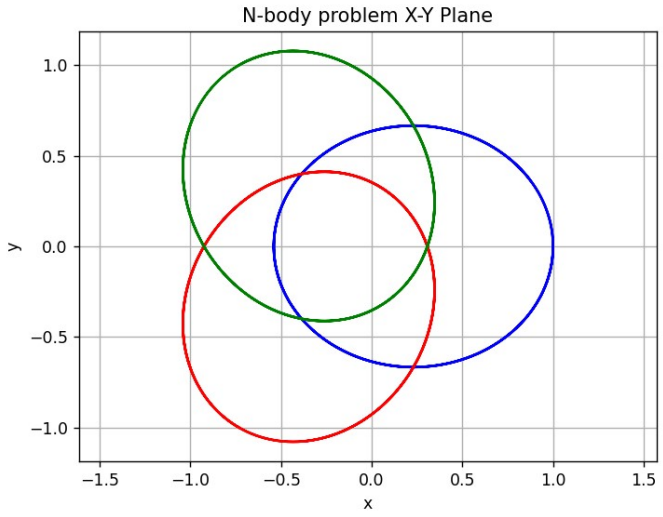
3.4. Código final puntos de Lagrange

4. Resultados

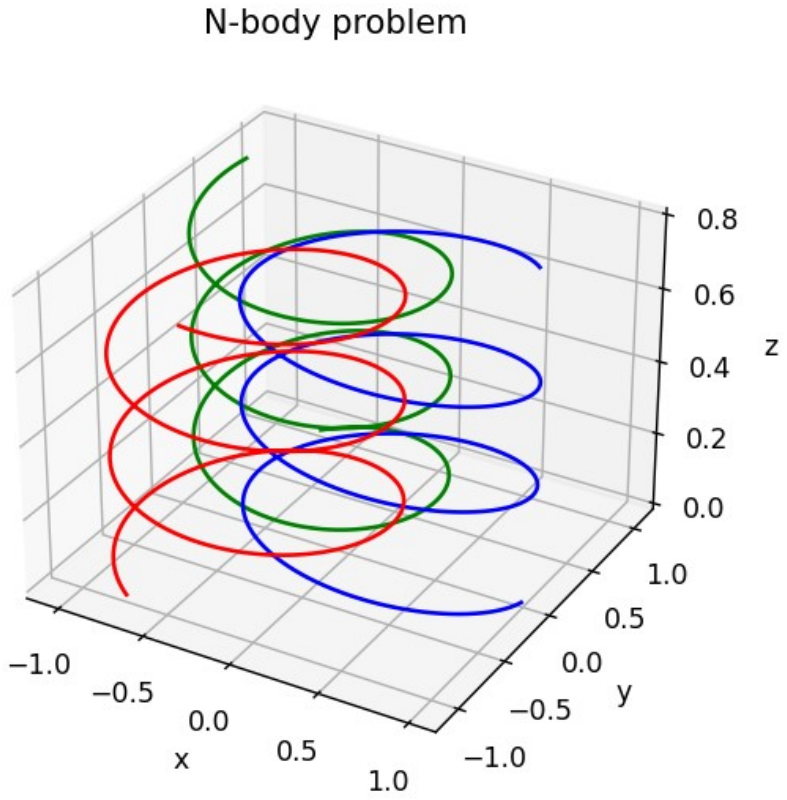
4.1. Problema de los N cuerpos

Se ha probado el programa con diversas condiciones iniciales:

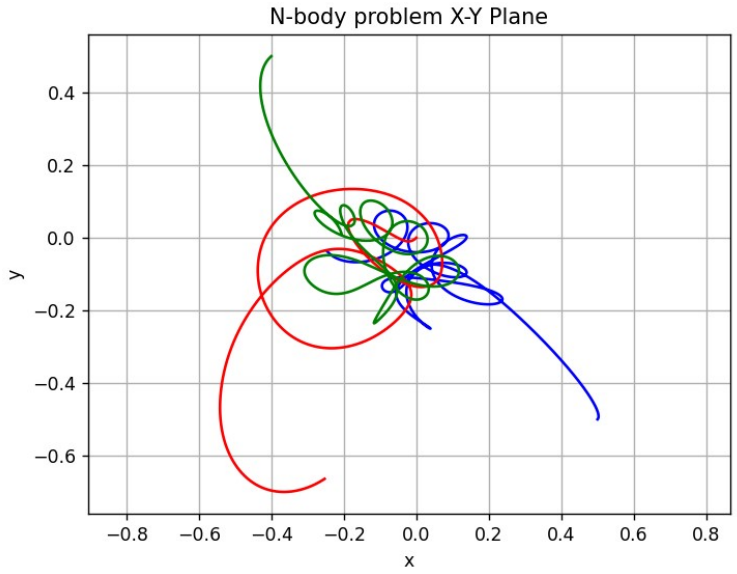
3 cuerpos estables



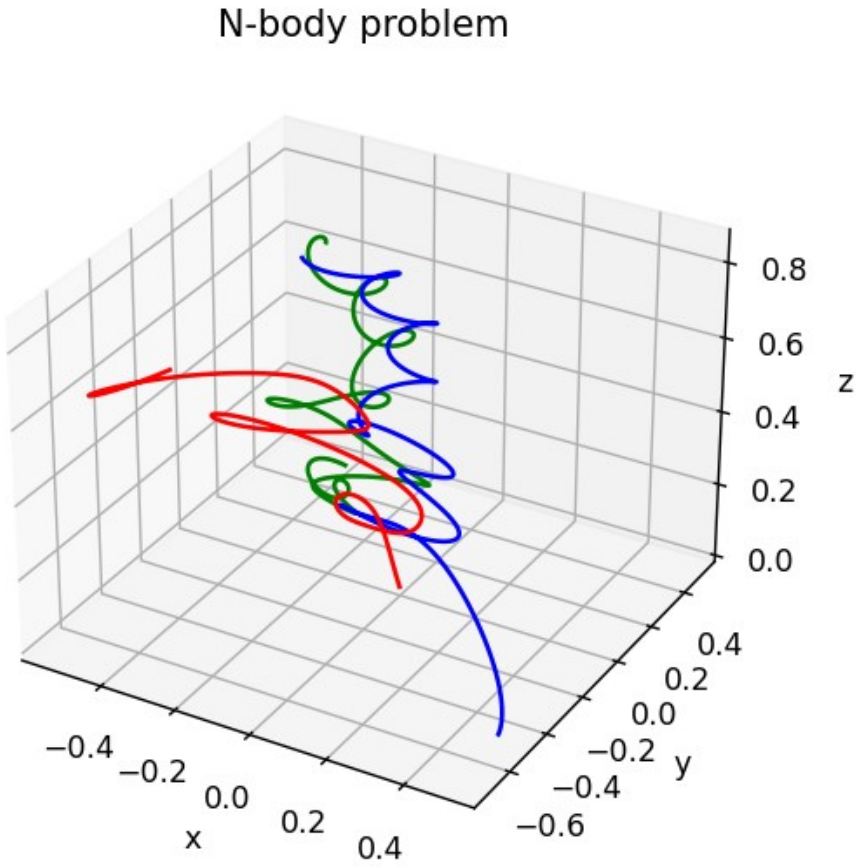

```
vini = 0.5
#Cuerpo 1
r0[0,:] = [1, 0, 0]
v0[0,:] = [0, -vini, 0.05]
#Cuerpo 2
r0[1,:] = [-sqrt(3)+1, -1, 0]
v0[1,:] = [-vini*sqrt(3)/2, vini/2, 0.05]
#Cuerpo 3
r0[2,:] = [-sqrt(3)+1, 1, 0]
v0[2,:] = [vini*sqrt(3)/2, vini/2, 0.05]
```



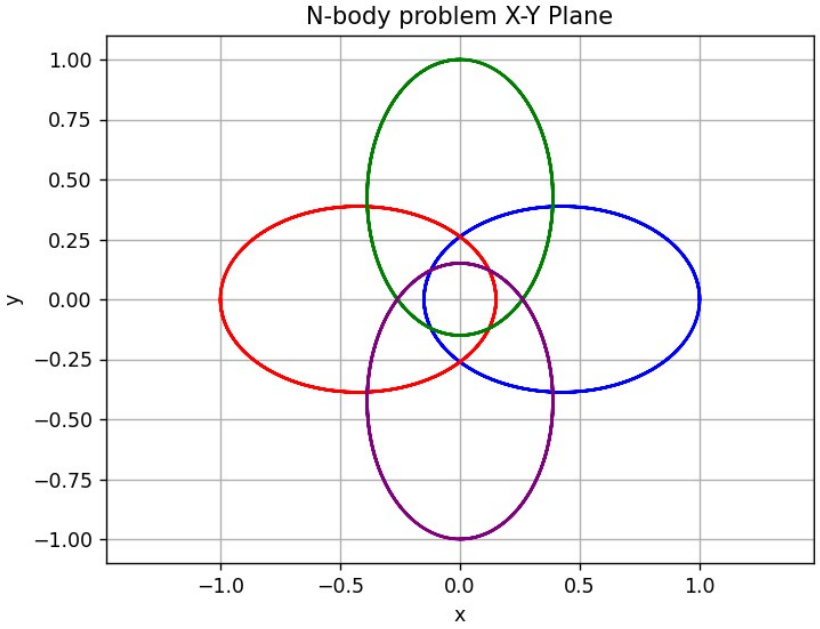
3 cuerpos inestables



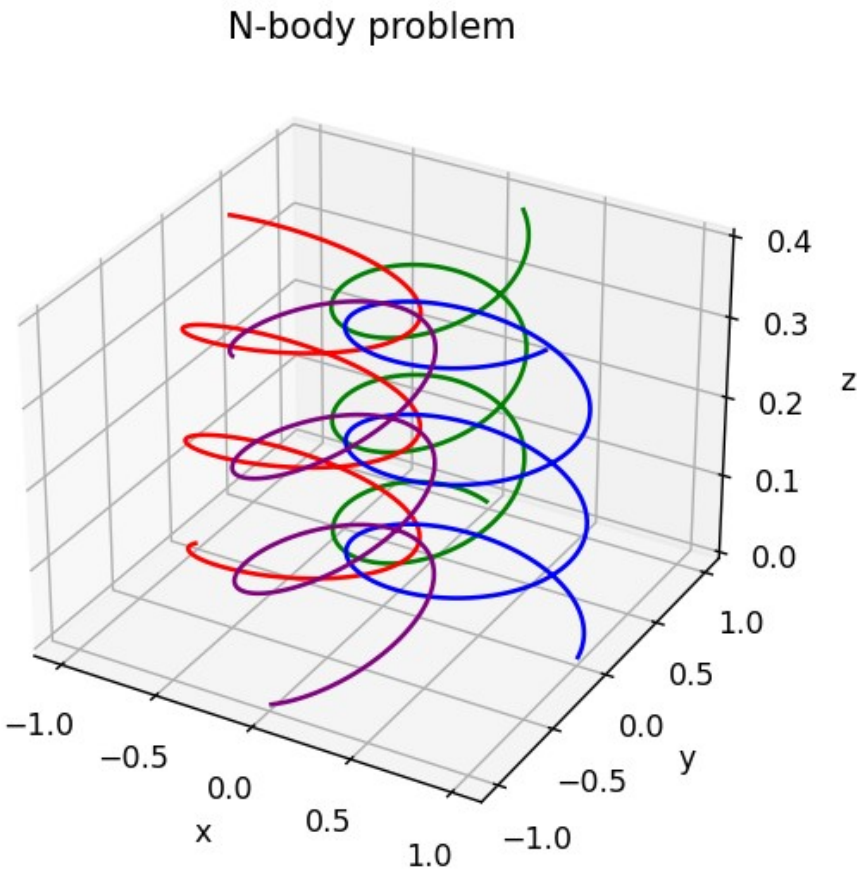

```
#Cuerpo 1
r0[0,:] = [0.5, -0.5, 0]
v0[0,:] = [0.1, 0.1, 0.3]
#Cuerpo 2
r0[1,:] = [0, 0, 0]
v0[1,:] = [-0.1, -0.1, 0.5]
#Cuerpo 3
r0[2,:] = [-0.4, 0.5, 0]
v0[2,:] = [-0.35, -0.3, 0.3]
```



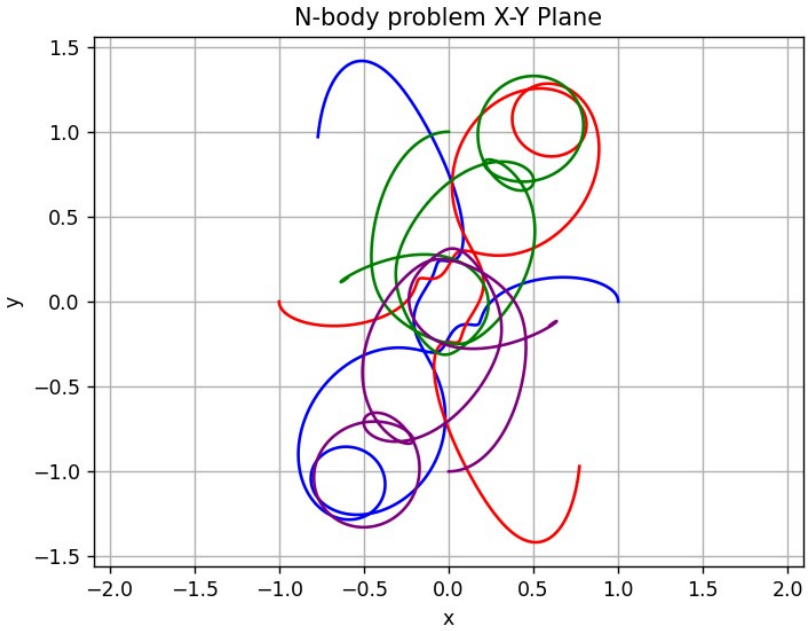
4 cuerpos estables



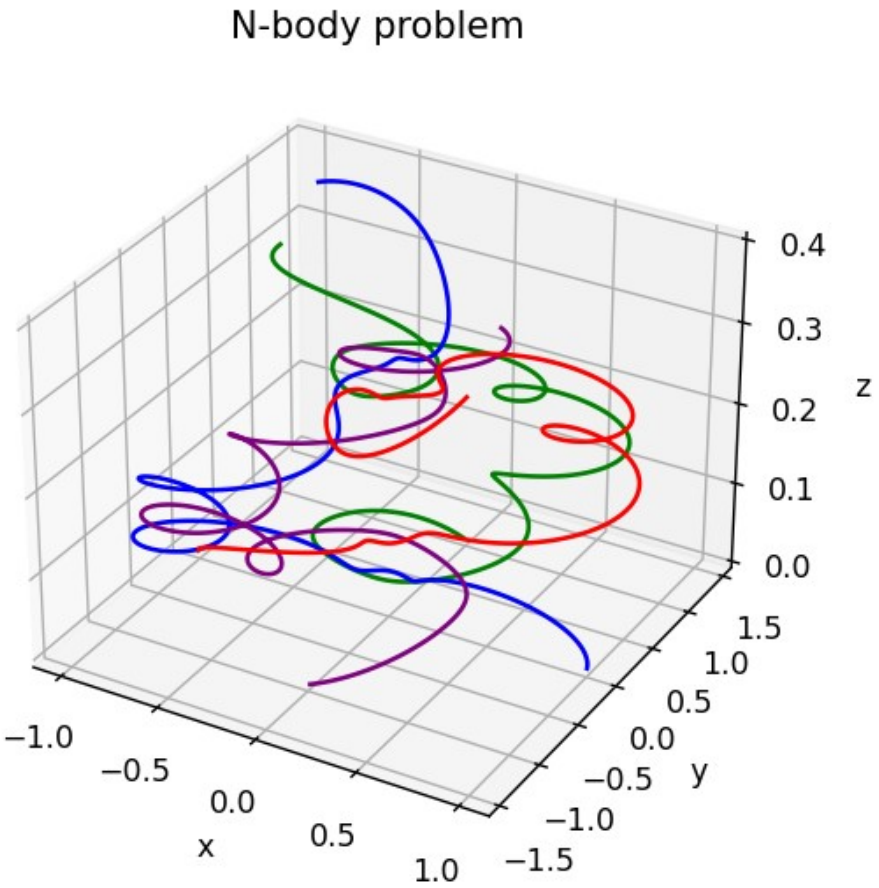

```
#Cuerpo 1
r0[0,:] = [1, 0, 0]
v0[0,:] = [0, 0.5, 0.05]
#Cuerpo 2
r0[1,:] = [-1, 0, 0]
v0[1,:] = [0, -0.5, 0.05]
#Cuerpo 3
r0[2,:] = [0, 1, 0]
v0[2,:] = [-0.5, 0, 0.05]
#Cuerpo 4
r0[3,:] = [0, -1, 0]
v0[3,:] = [0.5, 0, 0.05]
```



4 cuerpos inestables




```
#Cuerpo 1
r0[0,:] = [1, 0, 0]
v0[0,:] = [0, 0.25, 0.05]
#Cuerpo 2
r0[1,:] = [-1, 0, 0]
v0[1,:] = [0, -0.25, 0.05]
#Cuerpo 3
r0[2,:] = [0, 1, 0]
v0[2,:] = [-0.5, 0, 0.05]
#Cuerpo 4
r0[3,:] = [0, -1, 0]
v0[3,:] = [0.5, 0, 0.05]
```



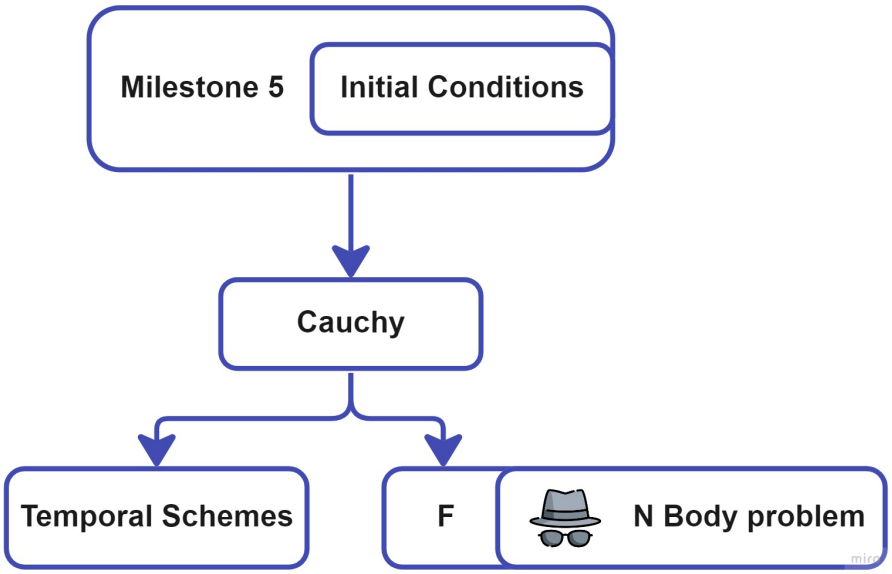
Como se puede observar, los resultados tienen sentido y son coherentes entre sí.

Para una mejor visualización de los resultados se han animado y guardado en formato de vídeo en el GitHub del alumno:

`https://github.com/jahrTeaching/milestones-SCaviaF/tree/main/doc/NBodyAnimations`

Creo necesario señalar que el esquema temporal que se ha utilizado es el Runge-Kutta de cuarto orden y que el paso elegido ha sido 0.002 unidades temporales.

Finalmente, la estructura del programa es la siguiente:



Donde se puede ver cómo el problema principal incluye la función de condiciones iniciales y cómo Cauchy llama a una F de paja que realmente está llamando a la función del problema de los N cuerpos.

4.2. Puntos de Lagrange