Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestone 6: Puntos de Lagrange y su estabilidad

Ampliación de Matemáticas I:
Cálculo numérico

11 de diciembre de 2022

Autor:

- Daniel Cerón Granda

ÍNDICE

PÁGINA

1. **Introducción……3**
2. **Esquema Runge-Kutta de alto orden embebido ……3**
3. **Función del problema de los 3 cuerpos restringido…… 6**
4. **Determinación de los puntos de Lagrange …… 8**
5. **Estabilidad de los puntos de Lagrange……10**
6. **Órbitas alrededor de los puntos de Lagrange……12**
7. **Relaciones funcionales del programa utilizado ……12**
8. **Crítica del código …… 12**

1.Introducción

El objetivo principal de este Milestone es la determinación de las órbitas en torno a los puntos de Lagrange, además de la estabilidad de estas. Para ello, se creará un esquema de Runge-Kutta embebido de alto orden, ya que es un problema que necesita una precisión muy alta. Se creará una función para la simulación del problema de los 3 cuerpos restringido y se determinarán los puntos de Lagrange para un sistema determinado.

2.Esquema Runge-Kutta de alto orden embebido

Los esquemas Runge-Kutta embebidos son de la forma:

$$U^{n+1} = U^n + \Delta t_n * \sum_{i=1}^{s} b_i * k_i$$

Ecuación 1: Expresión general de un esquema Runge-Kutta embebido

$$k_i = F(U^n + \Delta t_n * \sum_{j=1}^{s} a_{ij} * k_i \ , t_n + c_i * \Delta t_n)$$

Ecuación 2: Expresión de los coeficientes k_i

Los coeficientes a, b y c son característicos de cada esquema, así como el número de etapas s. Los coeficientes a forman la matriz de Butcher, una matriz de tamaño s por s, que determina si el esquema es implícito o explícito. Si la matriz es triangular inferior y los elementos de la matriz principal son nulos, el esquema es explícito, en cualquier otro caso, es un esquema implícito.

El esquema elegido para este trabajo es el esquema Dormand Prince, un esquema de siete etapas. Calcula soluciones de cuarto y quinto orden, siendo la diferencia entre ambas el error que se estima para la solución de cuarto orden. Los coeficientes están elegidos para minimizar el error de la solución de quinto orden. Es un esquema explícito, siendo los coeficientes de la matriz de Butcher:

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	−56/15	32/9				
8/9	19372/6561	−25360/2187	64448/6561	−212/729			
1	9017/3168	−355/33	46732/5247	49/176	−5103/18656		
1	35/384	0	500/1113	125/192	−2187/6784	11/84	
	35/384	0	500/1113	125/192	−2187/6784	11/84	0
	5179/57600	0	7571/16695	393/640	−92097/339200	187/2100	1/40

Figura 1: Coeficientes de la matriz de Butcher del esquema Dormand Prince

Siendo las filas de abajo los coeficientes b (utilizados en la solución de cuarto orden) y b_s (utilizados en la solución de quinto orden).

Finalmente, para asegurar que el error está acotado respecto a una tolerancia determinada, el paso de tiempo utilizado debe ser:

$$\Delta t_{max} = \Delta t * \left(\frac{\varepsilon}{\| U_*^{n+1} - U^{n+1} \|} \right)^{1/q}$$

Ecuación 3: Expresión del paso de tiempo máximo para acotación del error

Siendo ε la tolerancia del error elegida, q el orden del esquema, U^{n+1} la solución con un error de orden q+1 y U_*^{n+1} la solución con un error de orden q. Δt es el paso de tiempo utilizado, si el paso de tiempo máximo es mayor que Δt, se debe utilizar un paso de tiempo Δt dividido entre N (parte entera del cociente entre el paso de tiempo utilizado y el máximo más 1); si es menor, basta con utilizar el Δt.

En el código se ha definido el esquema Dormand Prince y sus funciones asociadas de la siguiente manera:

```
12  #Función de un paso de Runge-Kutta embebido: Dormand-Prince
13  def Emb_RK_DOPRI(U0, deltat, t, F):
14      Ord4 = RK_DOPRI("4", U0, t, deltat, F)
15      Ord5 = RK_DOPRI("5", U0, t, deltat, F)
16
17      (a, b, bs, c, q, Ne) = Butcher_matriz_DOPRI()
18
19      tolerance = 1e-9
20      h = min( deltat, deltatmax(Ord4 - Ord5, tolerance, min(q), deltat) )
21
22      N = int( deltat / h ) + 1
23      h = deltat / N
24
25      for i in range(N):
26          time = t + i * deltat / N
27          Ord4 = Ord5
28          Ord5 = RK_DOPRI("5", Ord4, time, h, F)
29
30      U = Ord5
31
32      return U
```

Figura 2: Función para un paso del esquema RK Dormand Prince embebido

Los inputs de la función son una condición inicial, un paso de tiempo, el tiempo de la iteración y la función a integrar. Se definen las soluciones de cuarto y quinto orden del esquema, los coeficientes de la matriz de Butcher (aunque en esta parte sólo es necesario el orden del esquema), se calcula el paso de tiempo necesario para las iteraciones a partir del cual se define el paso de tiempo y se pasa a calcular la función U que debe devolver el esquema, utilizando la solución del paso anterior como condición inicial para la iteración del paso siguiente y empezando como iteración inicial en la

solución de orden 4.

La función para integrar un paso del esquema Dormand Prince es:

```
34  #Función de un paso del esquema Dormand Prince
35  def RK_DOPRI(Order, U0, t, deltat, F):
36  
37      (a, b, bs, c, q, Ne) = Butcher_matriz_DOPRI()
38      N = len(U0)
39      k = zeros( [Ne, N] )
40      k[0,:] = F( U0, t + c[0]*deltat )
41  
42      if Order == "4":
43  
44          for i in range(1,Ne):
45              Up = U0
46  
47              for j in range(i):
48                  Up = Up + deltat * a[i,j]*k[j,:]
49  
50              k[i,:] = F( Up, t + c[i]*deltat )
51  
52          U1 = U0 + deltat * matmul(b,k)
53  
54      elif Order == "5":
55  
56          for i in range(1,Ne):
57              Up = U0
58  
59              for j in range(i):
60                  Up = Up + deltat * a[i,j]*k[j,:]
61  
62              k[i,:] = F(Up, t + c[i] * deltat)
63  
64          U1 = U0 + deltat * matmul(bs,k)
65  
66      return U1
```

Figura 3: Función para la integración de un paso del esquema Dormand Prince

Donde se obtienen los coeficientes de la matriz de Butcher, que aquí sí que van a utilizarse. Después, tiene dos caminos, según el orden de la solución pedida por el usuario, para orden 4 utiliza los coeficientes b y para orden 5 utiliza los coeficientes b_s. Se hace uso de la función matmul, que aplica un producto escalar a los vectores que contienen los coeficientes b y k para simular la operación del sumatorio de los productos de estos coeficientes.

La función para determinar el paso de tiempo a utilizar es la siguiente:

```
68  #Definición de paso de tiempo utilizado
69  def deltatmax(dU, tolerance, q, dt):
70  
71      if( norm(dU) > tolerance ):
72          deltat =  dt *(tolerance/norm(dU))**(1/(q+1))
73      else:
74          deltat = dt
75  
76      return deltat
```

Figura 4: Definición del paso de tiempo utilizado

Donde se da la diferencia entre las soluciones de un orden y de un orden superior (4 y 5 en el caso del esquema Dormand Prince) y devuelve el Δt a utilizar.

Finalmente, la función que devuelve los coeficientes de la matriz de Butcher para el esquema Dormand Prince es la siguiente:

```
78  #Definición de la matriz de Butcher para el esquema Dormand-Prince
79  def Butcher_matriz_DOPRI():
80      q = [5,4]
81      Ne = 7
82  
83      a = zeros( [Ne, Ne-1] )
84      b = zeros( [Ne] )
85      bs = zeros( [Ne] )
86      c = zeros( [Ne] )
87  
88      c[:] = [ 0., 1./5, 3./10, 4./5, 8./9, 1., 1. ]
89  
90      a[0,:] = [          0.,           0.,           0.,         0.,           0.,     0. ]
91      a[1,:] = [      1./5  ,           0.,           0.,         0.,           0.,     0. ]
92      a[2,:]= [      3./40 ,        9./40,           0.,         0.,           0.,     0. ]
93      a[3,:] = [     44./45 ,      -56./15,        32./9,         0.,           0.,     0. ]
94      a[4,:] = [ 19372./6561, -25360./2187,  64448./6561,  -212./729,           0.,     0. ]
95      a[5,:] = [  9017./3168,    -355./33 ,  46732./5247,    49./176, -5103./18656,     0. ]
96      a[6,:]= [    35./384 ,           0.,    500./1113,   125./192, -2187./6784 , 11./84 ]
97  
98      b[:]  = [ 35./384   , 0.,   500./1113,  125./192,  -2187./6784  ,  11./84  ,     0.]
99      bs[:] = [5179./57600, 0., 7571./16695,  393./640, -92097./339200, 187./2100, 1./40 ]
100 
101     return (a, b, bs, c, q, Ne)
```

Figura 5: Definición de los coeficientes de la matriz de Butcher para el esquema Dormand Prince

3.Función del problema de los 3 cuerpos restringido

El problema de los 3 cuerpos restringido es una particularización del problema de los N cuerpos. Asume que, de los 3 cuerpos estudiado, la masa de uno de ellos es despreciable frente a la de los otros dos, sólo estudiando las fuerzas gravitatorias de los cuerpos con mayores masas. Es uno de los pocos problemas de los N cuerpos que tiene solución analítica, debido principalmente a que es un problema de los 2 cuerpos (que también tiene solución analítica) pero estudiando cómo influyen estos dos cuerpos en otro cuerpo de masa despreciable que está suficientemente cerca de estos como para verse sometido por sus fuerzas gravitatorias.

Para este problema se va a tratar el sistema Tierra-Luna, y se van a estudiar las ecuaciones de Arenstorf, que definen una órbita estable periódica entre estos dos cuerpos. La órbita resultante en ejes inerciales respecto de la Tierra es:

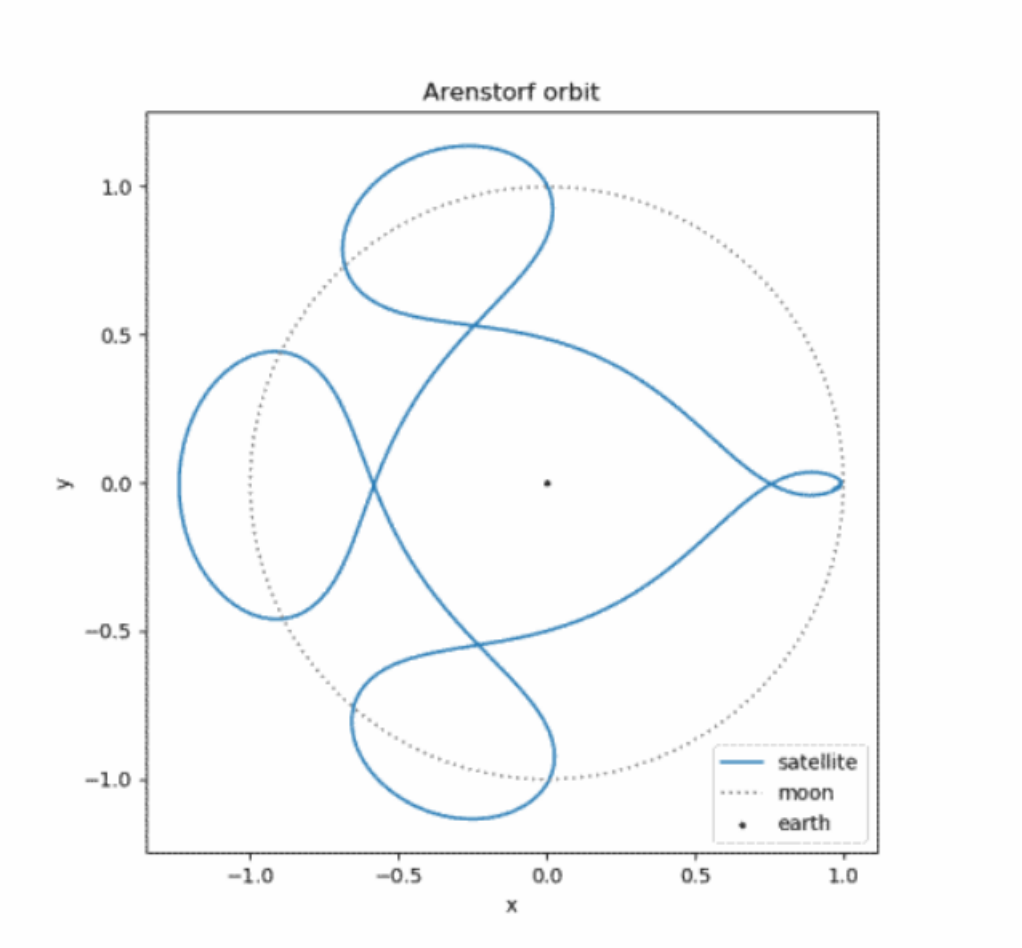


Figura 6: Órbita de Arenstorf

El sistema de ecuaciones diferenciales de la órbita de Arenstorf es:

$$x'' = x + 2y' - \mu' \frac{x+\mu}{D_1} - \mu \frac{x-\mu'}{D_2}$$
$$y'' = y - 2x' - \mu' \frac{y}{D_1} - \mu \frac{y}{D_2}$$

$$D_1 = ((x+\mu)^2 + y^2)^{3/2}$$
$$D_2 = ((x-\mu')^2 + y^2)^{3/2}$$

Ecuación 4: Sistema de ecuaciones diferenciales de la órbita de Arenstorf

Donde:

$$\mu = \frac{M_L}{M_T + M_L} = 0{,}0122741$$

$$\mu' = \frac{M_T}{M_T + M_L} = 1 - \mu = 0{,}9877259$$

Ecuación 5: Expresiones de μ y μ'

Siendo M_L la masa de la Luna y M_T la masa de la Tierra.

En el código, la función que representa este sistema de ecuaciones diferenciales es:

```
103   def R3BP(U0, t, mu):
104   |
105   |   r = U0[0,1]
106   |   v = U0[2,3]
107   |
108   |   D1 = ( ( r[0] + mu )**2 + r[1]**2 )**(3/2)
109   |   D2 = ( (r[0] - 1 + mu)**2 + r[1]**2 )**(3/2)
110   |
111   |   dvdt_1 =  r[0]+2*v[1]-(1-mu)*(r[0]+mu)/D1-mu*(r[0]-1+mu)/D2
112   |   dvdt_2 = r[1]-2*v[0]-(1-mu)*r[1]/D1-mu*r[1]/D2
113   |
114   |   return array( [ v[0], v[1], dvdt_1, dvdt_2] )
```

Figura 7: Función del problema de los tres cuerpos restringido

Donde se definen los vectores posición y velocidad, los parámetros D1 y D2 y se obtienen las derivadas de la velocidad en x e y, devolviendo las velocidades (derivadas de la posición) y sus derivadas.

4.Determinación de los puntos de Lagrange

Los puntos de Lagrange en un sistema de tres cuerpos restringido son aquellos cinco puntos del espacio donde el objeto de masa despreciable puede teóricamente estar en equilibrio estacionario respecto a los otros dos cuerpos. Marcan las posiciones donde la atracción gravitatoria combinada de las dos masas puede generar la fuerza necesaria para rotar centrípetamente respecto a la menor de ellas. Son las soluciones del problema de los tres cuerpos restringido a órbitas circulares, provocando que el cuerpo pueda mantener su posición relativa con respecto a los otros dos.

De forma teórica, los puntos L1, L2 y L3 deben estar en la línea imaginaria que une los dos cuerpos, mientras que L4 y L5 deben estar fuera de ella, pero equidistantes a la misma. L1 es el punto situado entre los dos cuerpos, L2 el punto situado más cerca de la masa menor, L3 el punto situado más cerca de la masa mayor, L4 el punto situado por encima y L5 el punto situado por debajo.

La función para calcular los puntos de Lagrange es la siguiente:

```
116  def Lagrange_Points(U0, N_LP, mu):
117
118      LP = zeros( [5,2] )
119
120      def F(Y):
121          X = zeros(4)
122          X[:2] = Y
123          X[2:] = 0
124          Z = R3BP(X,0,mu)
125          return Z[2:4]
126
127      for i in range(N_LP):
128          LP[i,:] = fsolve(F,U0[i,:2])
129
130      return LP
```

Figura 8: Obtención de los puntos de Lagrange

Que con los inputs de un vector de condiciones iniciales para la iteración, el numero de puntos de Lagrange buscados (será cinco) y el μ del sistema, estudia el problema de los tres cuerpos restringido de la función anterior y obtiene las soluciones estacionarias a partir de la función fsolve.

Desde el código se llama a la función de la siguiente forma:

```
155  #Condiciones iniciales
156  mu_earth_Moon = 1.22741e-2
157  N = 10000
158  t = linspace(0, 100, N)
159
160  # Condiciones iniciales para la obtención de los puntos de Lagrange
161  U0 = zeros( [5,4] )
162  U0[0,:] = [0.8, 0.6, 0 , 0]
163  U0[1,:] = [0.8, -0.6, 0, 0]
164  U0[2,:] = [-0.1, 0, 0, 0]
165  U0[3,:] = [0.1, 0, 0, 0]
166  U0[4,:] = [1.01, 0, 0, 0]
167
168  #Obtención de los puntos de Lagrange
169  L_p = Lagrange_Points(U0, 5, mu_earth_Moon)
170  print("\n" + str(L_p) + "\n")
```

Figura 9: Obtención de los puntos de Lagrange del sistema

Donde se definen las condiciones iniciales del sistema y de la obtención de los puntos de Lagrange y se llama a la función para obtenerlos. Los puntos obtenidos son:

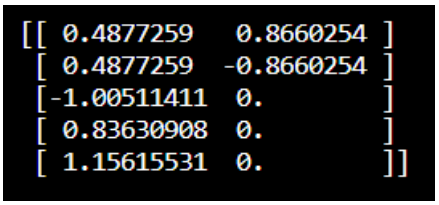


Figura 10: Obtención de los puntos de Lagrange con el código

Siendo el primero L5, el segundo L4, el tercero L3, el cuarto L1 y el quinto L2. Las coordenadas están situadas de tal forma que los ejes XY están situados en la Tierra y la Luna está en el (1,0), por lo que los puntos de Lagrange están adimensionalizados respecto a esta distancia. Se puede ver coherencia con los resultados esperados, L5 y L4 presentan simetría respecto a la línea de unión Tierra-Luna (eje X), hay un punto entre la Tierra y la Luna (L1 tiene una distancia X menor que 1), uno pasado la Luna pero cercano a ella y otro más a la izquierda de la Tierra. Además, L1, L2 y L3 están en el eje Tierra-Luna, por lo que los resultados parecen correctos.

5.Estabilidad de los puntos de Lagrange

Los puntos L1, L2 y L3 tienen un equilibrio inestable, una pequeña perturbación hace que se alejen rápidamente de su posición de equilibrio, mientras que los puntos L4 y L5 tienen un equilibrio estable, una perturbación sólo provoca oscilaciones respecto a ese punto que acaban amortiguándose por efecto Coriolis y tendiendo hacia él.

Como son puntos de equilibrio, se puede estudiar la estabilidad de las soluciones planteando cómo son los autovalores de la matriz del sistema del problema de los 3 cuerpos planteada. Si los autovalores obtenidos tienen todos parte real negativa (en los que sea nula, deben ser únicos), el punto de Lagrange asociado a ese autovalor será estable. Si algún autovalor tiene parte real positiva, el sistema será inestable, y si hay un autovalor múltiple con parte real nula, no se puede asegurar su estabilidad.

El programa calcula los autovalores de la matriz del problema de los N cuerpos con la siguiente función:

```
132  def Jacobian(F, x0):
133      N = len(x0)
134      dx = 1e-9
135      Jacobiano = zeros([N,N])
136      for j in range(N):
137          x = zeros(N)
138          x[j] = dx
139          Jacobiano[:,j] = ( F(x0 + x ) - F(x0 - x) ) /(2*dx)
140      return Jacobiano
141
142  def stb_Lp(U0, mu):
143      def F(X):
144          return R3BP(X, 0 , mu)
145
146      J = Jacobian(F, U0)
147      Autovalor, autovector = eig(J)
148      return Autovalor
```

Figura 11: Función de estabilidad y Jacobiano

Donde se ha definido una función que devuelve el jacobiano, que se utiliza para obtener la matriz de las derivadas del problema de los 3 cuerpos restringido. A continuación, se obtienen los autovalores y autovectores de la misma. En el programa se llama a la función y se obtiene la estabilidad de la siguiente forma:

```
167  #Estabilidad de los puntos de Lagrange
168  L_p_stb = zeros( 4 )
169
170  for i in range(5):
171      L_p_stb[:2] = L_p[i, :]
172      estab = around( stb_Lp(L_p_stb, mu_earth_Moon), 5)
173      print(str(estab) + "\n")
```

```
[-0.+0.95398j -0.-0.95398j  0.+0.29986j  0.-0.29986j]

[ 0.+0.95398j  0.-0.95398j -0.+0.29987j -0.-0.29987j]

[-0.     +1.01052j -0.     -1.01052j -0.17877+0.j      0.17877+0.j     ]

[-2.93358+0.j      2.93358+0.j      0.     +2.33535j  0.     -2.33535j]

[-2.15755+0.j      2.15755+0.j      0.     +1.86199j  0.     -1.86199j]
```

Figura 12: Estabilidad de los puntos de Lagrange

Para los puntos L4 y L5, todos los autovalores tienen parte real nula, pero son únicos, por lo que son puntos estables. Para L1, L2 y L3, hay un autovalor que presenta parte real positiva, por lo que son inestables. Los resultados obtenidos son perfectamente coherentes con la teoría.

6.Órbitas alrededor de los puntos de Lagrange

El código para la obtención de los puntos de Lagrange es:

```
#Órbitas alrededor de los puntos de Lagrange
U_0LPO = zeros( [shape(U0)[0], 4] )
eps = 1e-5
for i in range( shape(U0)[0] ):
    U_0LPO[i, :2] = L_p[i, :] + eps
    U_0LPO[i, 2:] = eps

def F(U,t):
   return R3BP(U, t, mu_earth_Moon)

F0 = zeros(len(U_0LPO))
Scheme = [RK4, Euler, Euler_inverso, CN, Emb_RK_DOPRI]
for j in range (len(Scheme)):
    for i in range(shape(U0)[0]):
        U_LP = Cauchy_problem(Scheme[j], U_0LPO[i,:], N, 100/N, F, F0, len(U_0LPO))
        LP = ["L4", "L5", "L3", "L1", "L2"]
```

```
        fig, ax = plt.subplots( figsize = (10, 10) )

        for k in range( len(L_p) ):
            ax.plot( L_p[k, 0],L_p[k, 1], 'o', label = LP[i] )

        ax.plot( U_LP[0,:], U_LP[1,:], color = 'b', label = "Orbit" )
        ax.set_xlabel("x")
        ax.set_ylabel("y")
        ax.set_title("Puntos de Lagrange del sistema Tierra-Luna y órbita alrededor de" + LP[j])
        plt.legend()
        ax.grid()
        if LP[j] == "L4" or LP[j] == "L5":
            fig, ay = plt.subplots( figsize = (7,7) )
            ay.plot( L_p[j, 0],L_p[j, 1], 'o', label = LP[j] )
            ay.plot( U_LP[0,:], U_LP[1,:], color = 'b', label = "Orbit"  )
            ay.set_xlabel("x")
            ay.set_ylabel("y")
            ay.set_title("Orbit arround" + LP[j])
            plt.legend()
            ay.grid()
```

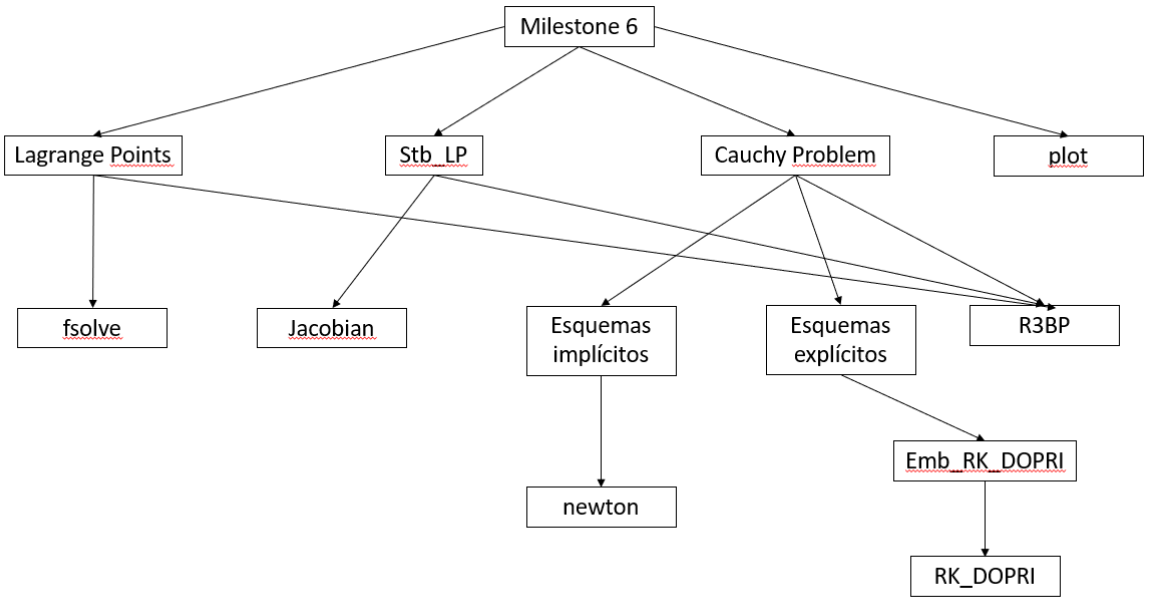
Figuras 13 y 14: Código para la representación de las órbitas alrededor de los puntos de Lagrange

En primer lugar, se definen las condiciones iniciales de las perturbaciones, con un épsilon pequeño (pero que en puntos inestables es más que suficiente para desplazar la órbita); y la función del problema de los tres cuerpos. Después, se hace la representación gráfica.

Este código está a unos pocos retoques de poder ser funcional al 100%, pero aún no es capaz de representar las órbitas.

7.Relaciones funcionales del programa utilizado

Las relaciones funcionales del programa utilizado son:



8.Crítica del código

La crítica principal del código está clara, no está completo al 100%. Faltan algunos retoques para que las órbitas puedan crearse, ya que los esquemas fallan a la hora de operar los vectores porque se obtienen de tamaños distintos. Tras intentar varios cambios en busca de la convergencia, no se ha podido encontrar la causa del fallo, y la falta de tiempo ha hecho que no se haya podido entregar completo. Aun así, el código cumple con la mayor parte de los objetivos del Milestone, obtiene tanto los puntos de Lagrange como su estabilidad, además de proporcionar funciones para la simulación del problema de los 3 cuerpos restringido y un esquema Runge-Kutta embebido de alto orden, además de tener gran parte del código preparado para la representación de las órbitas.

Se ha intentado seguir aplicando los preceptos de la programación funcional, reduciendo en la medida de lo posible el número de líneas, pero es cierto que en un programa más complejo como este no se ha podido reducir tanto como en otros Milestones.

Pese a no poder completar el hito al 100%, el progreso realizado durante el curso en los Milestones ha sido satisfactorio, mejorando mucho en la comprensión del lenguaje Python y mejorando en gran medida mis conocimientos de programación.