


Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestones 6

Ampliación de Matemáticas 1

12 de diciembre de 2022

Autor:

- Javier Zatón Miguel

Índice

1. Objetivo del software

El objetivo de este hito es determinar la posición de los puntos de Lagrange, su estabilidad y las orbital alrededor de los mismos. Para ello se ha desarrollado una función para el CR3BP (Circular restricted three body problem), y para la integración del problema orbital, se implementa un método de Runge-Kutta embebido.

2. Desarrollo de código

2.1. Runge-Kutta Embebido

El nuevo esquema numérico se ha desarrollado y añadido dentro del módulo de 'Scheme_module', donde se encuentran el resto de esquemas como Euler, Euler Inverso, Crank Nicolson, Runge Kutta 4 y Leap-frog.

```python
def RKEmb(U, F, dt, t):

    tol = 1e-9
    orders, Ns, a, b, bs, c = ButcherHeunEuler()
    est1 = RK(1, U, t, dt, F)
    est2 = RK(2, U, t, dt, F)
    h = min(dt, StepSize(est1-est2, tol, dt,  min(orders)))
    N_n = int(dt/h)+1
    n_dt = dt/N_n
    est1 = U
    est2 = U

    for i in range(N_n):
        time = t + i*dt/int(N_n)
        est1 = est2
        est2 = RK(1, est1, time, n_dt, F)

    return est2
```

Sin embargo el código de este esquema es mas complejo y requiere de otras subfunciones, estas se encuentran en un módulo que tendrá que llamar 'schemes' denominado 'RKE', este cuenta con tres funciones, 'RK', 'StepSize' y 'ButcherHeunEuler'.

```python
from numpy import zeros, matmul, size
from numpy.linalg import norm

def RK(order, U1, t, dt, F):
    orders, Ns, a, b, bs, c = ButcherHeunEuler()
    k = zeros([Ns, len(U1)])
    k[0,:] = F(U1, t + c[0]*dt)

    if order == 1:
        for i in range(1,Ns):
            Up = U1
            for j in range(i):
                Up = Up + dt*a[i,j]*k[j,:]
            k[i,:] = F(Up, t + c[i]*dt)
        U2 = U1 + dt*matmul(b,k)

    elif order == 2:
        for i in range(1,Ns):
            Up = U1
            for j in range(i):
                Up = Up + dt*a[i,j]*k[j,:]
            k[i,:] = F(Up, t + c[i]*dt)
        U2 = U1 + dt*matmul(bs,k)

    return U2
```

El RK Embebido que se ha usado es el de Heun-Euler, cuya tabla de Butcher viene definida en la función ButcherHeunEuler. La función StepSize ayuda a que el paso temporal cumpla con cierta tolerancia impuesta en el método.

```
def StepSize(dU, tol, dt, orders):
    error = norm(dU)
    if error > tol:
        step_size = dt*(tol/error)**(1/(orders+1))
    else:
        step_size = dt
    return step_size

def ButcherHeunEuler():
    orders = [2,1]
    Ns = 2

    a = zeros([Ns,Ns-1])
    b = zeros([Ns])
    bs = zeros([Ns])
    c = zeros([Ns])

    c = [0., 1.]
    a[0,:] = [0]
    a[1,:] = [  1. ]
    b[:]  = [1./2,1./2]
    bs[:] = [1.,0.]
    return orders, Ns, a, b, bs, c
```

Dentro del módulo RK, viene definido el método de orden 1 de Euler y el método de orden 2 de Heun, para ser utilizados por el RK Embebido.

2.2. CR3BP

Para el problema simplificado de tres cuerpos, se ha creado la función CR3BP en un módulo denominado 'CR3BPF', este problema supone que hay tres cuerpos, dos de ellos con masas mucho mayores que el tercero (cuya órbita es la que nos interesa), este escenario vale para una órbita en el entorno Tierra-Luna o Sol-Tierra, los cuales tendrán distintos baricentros (parámetro mu), la distancia del baricentro es un input de la función.

```
def CR3BP(U, mu):

    p1 = sqrt((U[0]+mu)**2 + U[1]**2)
    p2 = sqrt((U[0]-1+mu)**2 + U[1]**2)

    dvdt_x = 2*U[3] + U[0] -(1-mu)*(U[0]+mu)/(p1**3) - mu*(U[0]-1+mu)/(p2**3)
    dvdt_y = -2*U[2] + U[1] -(1-mu)*U[1]/(p1**3) - mu*U[1]/(p2**3)

    return array([U[2], U[3], dvdt_x, dvdt_y])
```

En este problema, existen posiciones en el plano (puntos de Lagrange) donde la aceleración del cuerpo con masa muy pequeña tiene resultante nula.

Para calcular estas posiciones se utiliza la función 'Lpoints' que insertando unas condiciones iniciales de posición (cercanas al punto de Lagrange) resuelve la función CR3BP donde las aceleraciones son nulas.

```
def Lpoints(U0, Np, mu):
    LP = zeros([Np,2])

    def F(Y):
        X = zeros(4)
        X[0:2] = Y
        X[2:4] = 0
        FX = CR3BP(X, mu)
        return FX[2:4]

    for i in range(Np):
        LP[i,:] = fsolve(F, U0[i,0:2])

    return LP
```

Conocidas las posiciones de estos puntos, se puede calcular su estabilidad analizando los autovalores de la matriz Jacobiana, para esto se usa una función 'StabLP' para obtener los autovalores de la matriz 'Jacobian'.

```
def Jacobian(F, U):
    N = size(U)
    Jac = zeros([N,N])
    t = 1e-10

    for i in range(N):
        xj = zeros(N)
        xj[i] = t
        Jac[:,i] = (F(U + xj) - F(U - xj))/(2*t)
    return Jac

def StabLP(U0, mu):

    def F(Y):
        return CR3BP(Y, mu)

    A = Jacobian(F, U0)
    values, vectors = eig(A)

    return values
```

Existe un inconveniente para la función CR3BP, los input son (U,mu), sin embargo el módulo de Cauchy solo puede utilizar funciones con inputs (U, t), para ello, se han creado dos funciones (MoonEarthCR3BP y EarthSunCR3BP) definiendo los valores del baricentro de cada problema y haciendo que devuelva la función CR3BP(U, mu).

De esta manera, se puede utilizar la función de Cauchy usada en anteriores Hitos.

```
def MoonEarthCR3BP(U,t):

    mu = 1.2151e-2 #Tierra-Luna

    return CR3BP(U, mu)

def EarthSunCR3BP(U,t):

    mu = 3.0039e-7 #Sol-Tierra

    return CR3BP(U, mu)
```

2.3. Programa principal

En el programa principal se ha realizado el caso del sistema Tierra-Luna. Primero se inserta un vector U0LP, que contiene cinco vectores fila, en estos vectores se insertan unos valores iniciales para iterar su posición de los puntos de Lagrange con la función 'Lpoints'.

```
from numpy import array, linspace, zeros
import matplotlib.pyplot as plt
from random import random
from Modulos.Cauchy_module import Cauchy
from Modulos.Scheme_module import RKEmb
from Modulos.CR3BPF import Lpoints, StabLP
from Modulos.Functions import MoonEarthCR3BP

##Datos
N =1000
tf = 10
t = linspace(0, tf, N)
mu = 1.2151e-2 #Tierra-Luna
#mu = 3.0039e-7 #Sol-Tierra

# puntos iniciales para iterar y determinar la posición de los puntos de Lagrange
U0LP = array([[0.5, 0, 0, 0],[1.5, 0, 0, 0],[-0.5, 0, 0, 0],[0.5, 0.5, 0, 0],[0.5, -0.5, 0, 0]])
LagPoints = Lpoints(U0LP, 5, mu)
print(LagPoints)
```

Con las posiciones obtenidas, se puede calcular la estabilidad utilizando un bucle sencillo y la función 'StabLP'.

```
#Estudio la estabilidad de los puntos de Lagrange

U0S = zeros(4)
U0SLP = zeros((5,4))
for i in range(5):
    U0S = zeros(4)
    U0S[0:2] = LagPoints[i,:]
    eingvalues = StabLP(U0S, mu)
    U0SLP[i,:] = eingvalues.real
print(U0SLP)

```

Para estudiar las órbitas alrededor de los cinco puntos de Lagrange, se ha creado una matriz de tres dimensiones, cuatro filas para almacenar posición y velocidad en ejes x e y, para un número de pasos temporales , todo esto para cada uno de los puntos de Lagrange (5 en total).

Las condiciones iniciales del satélite son las de la posición del punto de lagrange añadiendo un épsilon (para evitar que quede estacionado en el punto fijo).

```
33  ULG = zeros((4,N,5))
34  for LG in range(5):
35      U0 = zeros(4)
36      U0[0:2] = LagPoints[LG-1,:]
37      eps = 1e-4*random()
38      U0 = U0 + eps
39      U = Cauchy(MoonEarthCR3BP, U0, t, RKEmb)
40      ULG[:,:,LG-1] = U[:,:]
41
42  print(ULG)
```

Finalmente se crea varias gráficas donde se pueden ver los puntos de Lagrange del sistema Tierra-Luna y las distintas órbitas alrededor de los cinco LP, de esta forma, se puede comprobar la estabilidad o inestabilidad de dichos puntos de Lagrange de forma visual.

```
#Grafico

fig, axs = plt.subplots(2, 3)
axs[0, 0].plot(ULG[0,:,0], ULG[1,:,0],'-',color = "r")
axs[0, 0].plot(ULG[0,:,1], ULG[1,:,1],'-',color = "r")
axs[0, 0].plot(ULG[0,:,2], ULG[1,:,2],'-',color = "r")
axs[0, 0].plot(ULG[0,:,3], ULG[1,:,3],'-',color = "r")
axs[0, 0].plot(ULG[0,:,4], ULG[1,:,4],'-',color = "r")
axs[0, 0].plot(-mu, 0, 'o', color = "b")
axs[0, 0].plot(1-mu, 0, 'o', color = "#808080")
for i in range(5):
    axs[0, 0].plot(LagPoints[i,0], LagPoints[i,1] , 'o', color = "k")
axs[0, 0].set_title("Lagrange point View")

axs[0, 1].plot(ULG[0,:,4], ULG[1,:,4],'-',color = "r")
axs[0, 1].plot(LagPoints[4,0], LagPoints[4,1] , 'o', color = "k")
axs[0, 1].set_title("L5 View")

axs[0, 2].plot(ULG[0,:,3], ULG[1,:,3],'-',color = "r")
axs[0, 2].plot(LagPoints[3,0], LagPoints[3,1] , 'o', color = "k")
axs[0, 2].set_title("L4 View")

axs[1, 0].plot(ULG[0,:,2], ULG[1,:,2],'-',color = "r")
axs[1, 0].plot(LagPoints[2,0], LagPoints[2,1] , 'o', color = "k")
axs[1, 0].set_title("L3 View")

axs[1, 1].plot(ULG[0,:,1], ULG[1,:,1],'-',color = "r")
axs[1, 1].plot(LagPoints[1,0], LagPoints[1,1] , 'o', color = "k")
axs[1, 1].set_title("L2 View")

axs[1, 2].plot(ULG[0,:,0], ULG[1,:,0],'-',color = "r")
axs[1, 2].plot(LagPoints[0,0], LagPoints[0,1] , 'o', color = "k")
axs[1, 2].set_title("L1 View")


plt.show()
```

3. Discusión de Resultados

Las posiciones de los puntos de lagrange (x,y) obtenidas son las siguientes:

L1(0.869,0)

L2(1.156,0)

L3(-1.005,0)

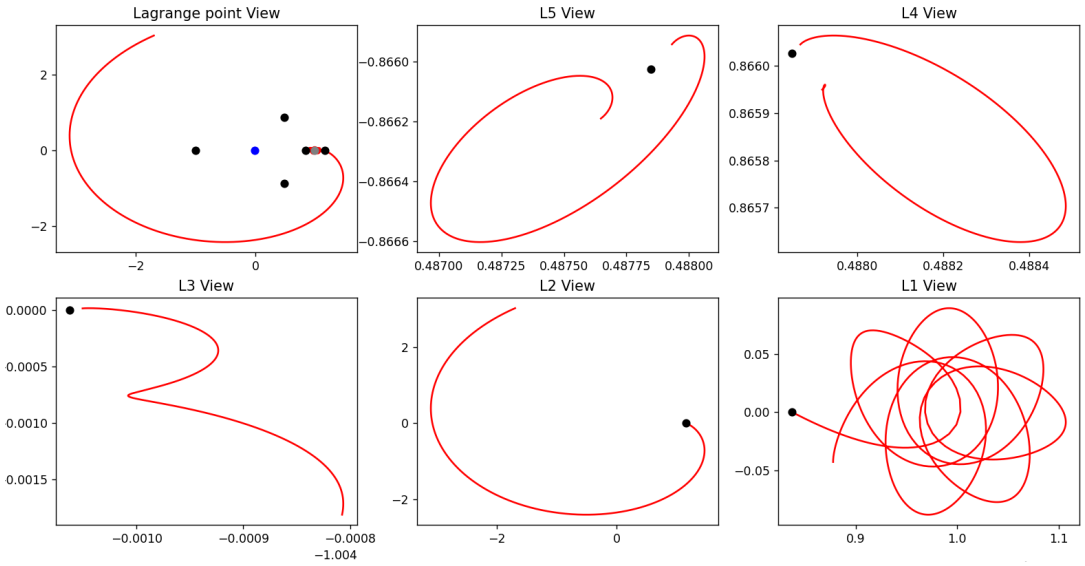
L4(0.488,0.866)

L5(0.488,-0.866)

Si se revisa los valores reales de los autovalores calculados, se puede observar que a excepción de L4 y L5, tienen autovalores de orden unidad, lo que significa que son inestables y los satélites que orbiten cerca de ese punto acabará alejándose.

```
[[-2.93206097e+00  2.93206097e+00 -4.57966998e-16 -4.57966998e-16]
 [ 2.15867055e+00 -2.15867055e+00 -5.09946027e-13 -5.09946027e-13]
 [ 5.65140382e-16  5.65140382e-16 -1.77878383e-01  1.77878383e-01]
 [-6.43554046e-07 -6.43554046e-07  6.43554046e-07  6.43554046e-07]
 [-1.36096367e-06 -1.36096367e-06  1.36096367e-06  1.36096367e-06]]
```

Para un tf = 10 se obtienen las siguientes órbitas.

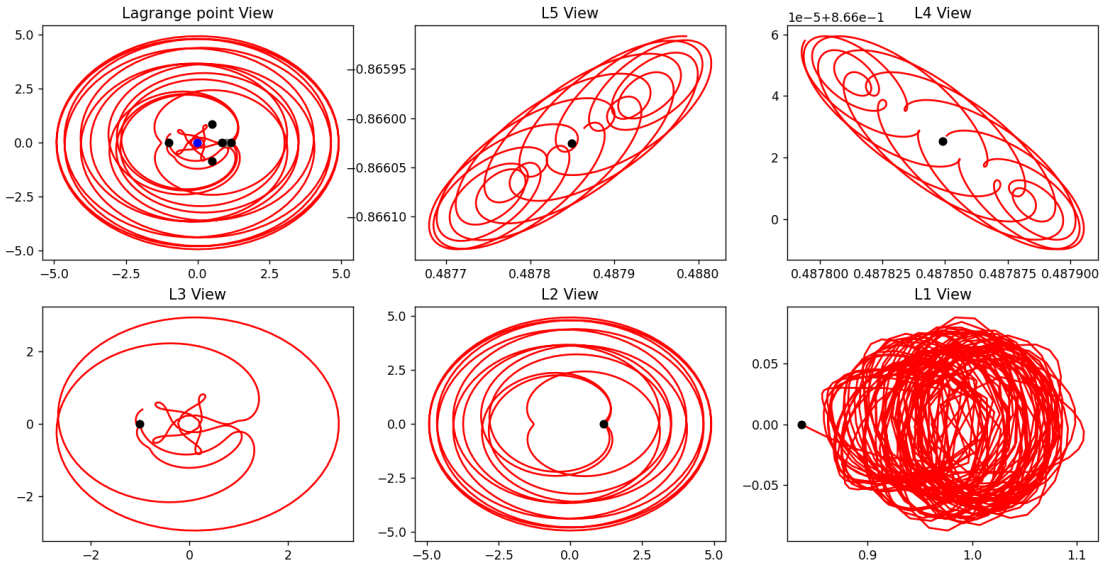


Si se observa el primer gráfico se observa la disposición de los puntos de Lagrange (puntos negros), la Tierra(punto azul) y la Luna (punto gris).

Se han creado cinco gráficas específicas para observar las orbitas de cada punto de Lagrange por separado.

La órbita de L5 se observa que es estable cerca del punto, al igual que el caso de L4, como ya se ha podido comprobar ambos son estables.

El punto L3, es inestable y tiende a alejarse del punto, aunque esta orbita no se aprecia aun en la gráfica 'Lagrange point View', al contrario de el caso de L2, que casi ha completado una órbita a la Tierra. En el caso de L1, este se aleja pero cae en el campo gravitatorio de la Luna.



Si aumentamos el parámetro tf a un valor de 100, se puede apreciar que las órbitas L5 y L4 siguen sin alejarse.

La órbita de L3 parece que después de alejarse del punto de Lagrange, fue atrapado por el campo gravitacional del Sol hasta cruzar por el punto L1 y realizar dos orbitas alrededor de la Tierra para luego regresar nuevamente al punto L3.

Con el caso de L2 sucede algo similar, aunque se ve con menor claridad, si que se puede apreciar que al realizar numerosas órbitas alrededor de la Tierra, ha vuelto a cruzar por la posición de origen.

Para el punto L1 no sucede esto ya que el satélite parece haber sido atrapado por el campo gravitacional de la Luna de forma indefinida.