


Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestone 2

Ampliación de Matemáticas I

28 de septiembre de 2022

Autor:

- Sergio Cavia Fraile

Índice

1. Introducción

El objetivo del trabajo será integrar órbitas de Kepler mediante distintos métodos numéricos utilizando el lenguaje Python.

2. Ecuaciones y Métodos

Las órbitas de Kepler se definen mediante el siguiente problema de Cauchy:

$\ddot{r} = \frac{\vec{r}}{|\vec{r}|^3} = F(U,t)$

$\vec{r}(0) = (1,0)$

$\dot{\vec{r}}(0) = (0,1)$

2.1. Método de Euler

El método de Euler se describe de la siguiente forma:

$U^1 = U^0 + \Delta t * F^0$

2.2. Método de Runge-Kutta de 4°orden

El método de Runge-Kutta de 4°orden se describe de la siguiente forma:

$U^1 = U^0 + \frac{\Delta t}{6} * (k_1 + k_2 + k_3 + k_4)$

Donde k_1, k_2, k_3, k_4 son:

$k_1 = f(x_1, y_1)$

$k_2 = f(x_1 + \frac{\Delta t}{2}, y_1 + \frac{k_1 * \Delta t}{2})$

$k_3 = f(x_1 + \frac{\Delta t}{2}, y_1 + \frac{k_2 * \Delta t}{2})$

$k_4 = f(x_1 + \Delta t, y_1 + k_3 * \Delta t)$

2.3. Método de Crank-Nicholson

El método de Crank-Nicholson se describe de la siguiente forma:

$U^1 = U^0 + \frac{\Delta t}{2} * (F^0 + F^1)$

2.4. Método de Euler inverso

El método de Euler inverso se describe de la siguiente forma:

$U^1 = U^0 + \Delta t * F^1$

3. Código

En primer lugar se han definido 4 funciones, una por cada esquema temporal definido.

Posteriormente definimos la función que integraremos, en nuestro caso la ecuación de órbitas de Kepler.

Continuaremos definiendo una última función que resuelva el problema de Cauchy.

Por último escribimos el código que dará el tiempo final, el paso temporal y las condiciones iniciales e integrará la función mediante el esquema que queramos.

3.1. Método de Euler

El método de Euler se ha implementado mediante las siguientes líneas de código:

```
def Euler(U, dt, t, F):

    return U + dt*F(U, t)
```

3.2. Método de Runge-Kutta de 4°orden

El método de Runge-Kutta de 4°orden se ha implementado mediante las siguientes líneas de código:

```
def RK4(U, dt, t, F):

    k1 = F(U, t)
    Unew1 = U[:] + dt*k1[:]/2
    k2 = F(Unew1, t)
    Unew2 = U[:] + dt*k2[:]/2
    k3 = F(Unew2, t)
    Unew3 = U[:] + dt*k3[:]
    k4 = F(Unew3, t)

    U = U + dt*(k1 + 2*k2 + 2*k3 + k4)/6

    return U
```

3.3. Método de Crank-Nicholson

El método de Crank-Nicholson se ha implementado mediante las siguientes líneas de código:

```
def CN(U, dt, t, F):

    def func_CN(x):
        return x - U -(F(x,t+dt) + F(U, t))*dt/2
    U = fsolve(func_CN, [U])

    return U
```

3.4. Método de Euler inverso

El método de Euler inverso se ha implementado mediante las siguientes líneas de código:

```
def EulerInv(U, dt, t, F):

    def func_EI(x):

        return x - U - F(x,t+dt)*dt
    U = fsolve(func_EI, [U])

    return U
```

3.5. Función de Kepler

La ecuación de órbitas de Kepler se ha implementado mediante las siguientes líneas de código:

```
def Kepler(U, t):

    return  array([U[2],
                   U[3],
                   -U[0]/((U[0]**2 + U[1]**2)**1.5),
                   -U[1]/((U[0]**2 + U[1]**2)**1.5) ] )
```

3.6. Problema de Cauchy

El problema de Cauchy se ha implementado mediante las siguientes líneas de código:

```
def Cauchy(F,t,U0,Scheme):

    U = zeros((len(t),len(U0)))
    U[0,:] = U0
    for n in range(len(t)-1):
        U[n+1,:] = Scheme(U[n, :], t[n+1] - t[n], t[n],  F)

    return U
```

3.7. Código final

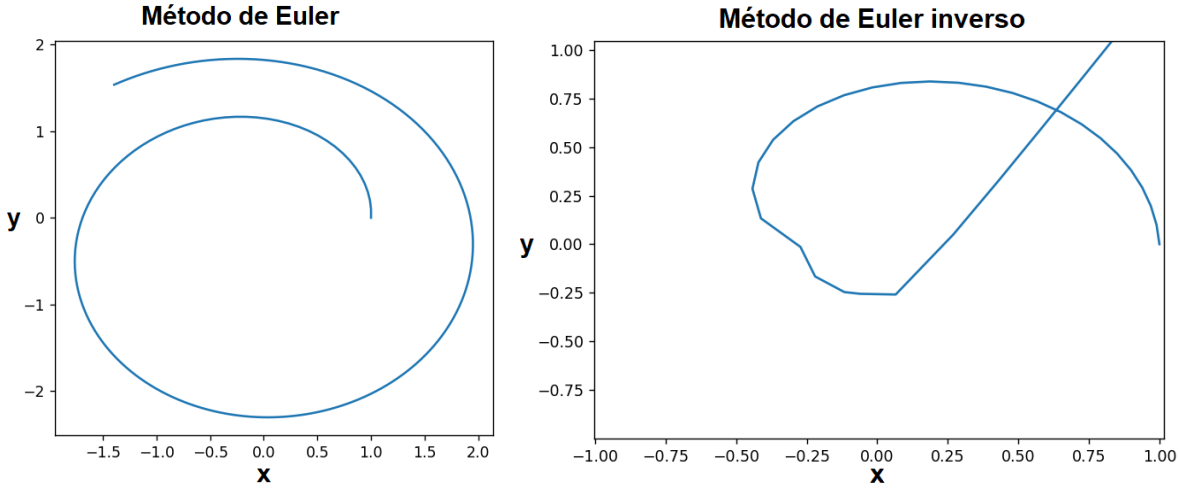
El código que realmente integra la función mediante el esquema temporal que decimos es el siguiente:

```
tf = 20
N = 200
t = linspace(0, tf, N)
U0 = array([1, 0, 0, 1])

U =  Cauchy(Kepler, t, U0, EulerInv)

plt.plot(U[:,0] , U[:,1])
plt.show()
```

4. Resultados

4.1. Métodos de Euler

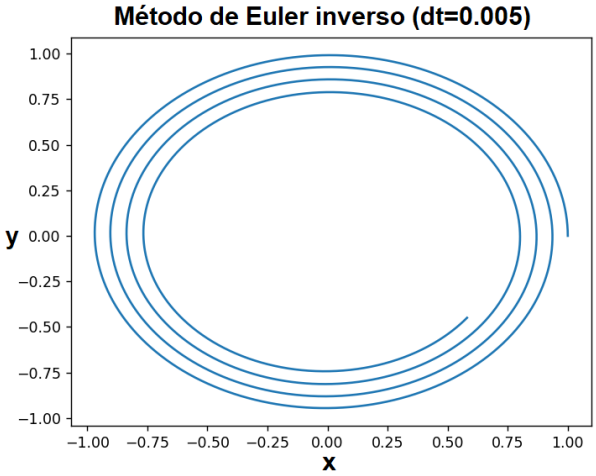
En las siguientes gráficas se pueden ver las órbitas que se obtienen integrando por los dos métodos de Euler explicados durante 20 unidades de tiempo con un paso de 0.1.



Como se puede ver, la energía no se conserva en ninguno de los dos métodos de Euler. El método de Euler hace aumentar la energía del orbitador dando como resultado una espira que se abre hacia fuera, mientras que el método de Euler inverso roba energía en cada paso temporal dando como resultado una espiral que cae hacia el centro.

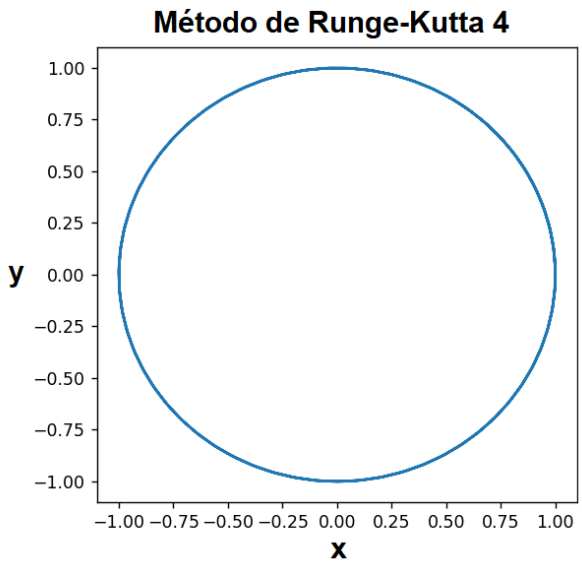
Además, podemos ver que el método de Euler inverso pierde energía tan rápidamente que llega a anular las coordenadas del orbitador, lo que da problemas (al provocar divisiones entre cero) y hace que el cuerpo salga disparado hacia el infinito.

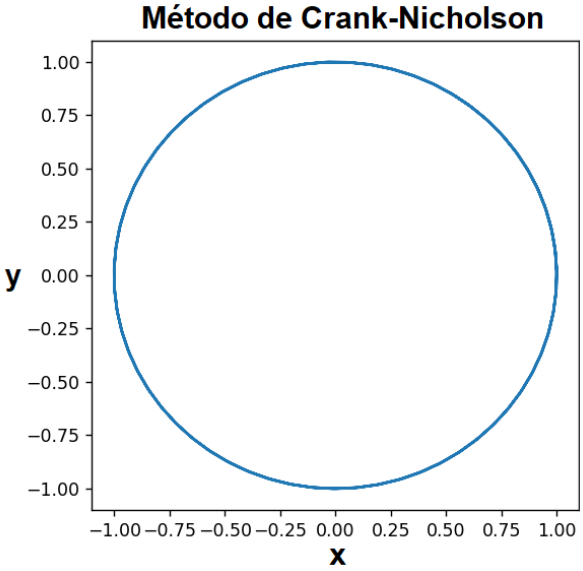
Esto se puede resolver utilizando pasos temporales más pequeños, por ejemplo, en la siguiente imagen se ha utilizado un paso de 0.005 unidades de tiempo durante varias órbitas:



4.2. Métodos de Runge-Kutta de 4°orden y de Crank-Nicholson

En las siguientes gráficas se pueden ver las órbitas que se obtienen integrando por los dos métodos restantes durante 20 unidades de tiempo con un paso de 0.1.





Utilizando los otros dos métodos no se ve diferencia a simple vista con una circunferencia de radio unidad, por lo que estos métodos tienen un error mucho inferior que los métodos de Euler.