


Universidad Politécnica de Madrid

Ampliación de Matemáticas I

Informe Hito 1

Autor:
Guillermo García del Río

Contenidos

Lista de figuras

1 Introducción

En este informe se presentan y comentan los resultados obtenidos en el hito 1 propuesto en la asignatura de Ampliación de Matemáticas I. En primer lugar se mostrará el enunciado que marca las pautas del proyecto, seguido de un análisis del problema a resolver, la exposición del código y los resultados obtenidos con él. Para terminar se comentarán y compararán dichos resultados.

2 Enunciado del hito

Los objetivos de este hito quedan establecidos por los siguientes puntos:

- Escribir un código que integre órbitas de Kepler con el método de Euler.
- Escribir un código que integre órbitas de Kepler con el método de Crank-Nicolson.
- Escribir un código que integre órbitas de Kepler con el método de Runge-Kutta de cuarto orden.
- Variar el paso de tiempo dado en la integración y comparar los resultados obtenidos.

3 Problema a resolver

Lo primero que hay que hacer es tener claro el problema que se pretende resolver. En este caso se trata de integrar una órbita de Kepler conociendo las condiciones iniciales de la misma. Las ecuaciones que rigen este problema son las siguientes:

$$\ddot{\vec{r}} = -\frac{\vec{r}}{|\vec{r}|^3} \tag{3.0.1}$$

Con las condiciones iniciales de posición y velocidad respectivamente:

$$\begin{aligned} \vec{r}(0) &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \dot{\vec{r}}(0) &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned} \tag{3.0.2}$$

De tal manera que en realidad a lo que nos estamos enfrentando es a un problema de Cauchy de la forma:

$$\begin{aligned} \frac{d\vec{U}}{dt} &= \vec{F}(U,t) \\ \vec{U}(0) &= \vec{U}_0 \end{aligned} \tag{3.0.3}$$

Donde U es el vector de estado formado por las componentes de la posición y la velocidad :

$$\vec{U} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} \tag{3.0.4}$$

Y a partir de su derivada se obtiene la función $F(U,t)$ que en este caso será:

$$\vec{F}(U,t) = \frac{d\vec{U}}{dt} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dfrac{-x}{(x^2+y^2)^{3/2}} \\ \dfrac{-y}{(x^2+y^2)^{3/2}} \end{pmatrix} \tag{3.0.5}$$

Como se puede observar, tanto el vector de estado como la $\vec{F}(U)$ variará dependiendo del problema que estemos tratando de resolver. En este caso la $\vec{F}(U)$ no depende del tiempo, por lo que es un sistema autónomo. La manera de resolver este problema dependerá del esquema numérico que se esté utilizando.

4 Exposición del código

De cara a poder entender lo que se está haciendo se va a proceder a exponer el código desarrollado para solucionar el problema propuesto al mismo tiempo que se introducirá la matemática de cada esquema numérico que se ha implementado.

Por otro lado, se ha llevado a cabo una declaración previa de parámetros y de funciones que se han definido para facilitar la implementación del código.

```
print('Por favor, defina las condiciones iniciales del problema en el siguiente orden: X0, Y0, Vx0, Vy0')

X0 = float(input())
Y0 = float(input())
Vx0 = float(input())
Vy0 = float(input())

U0 = np.array([X0,Y0,Vx0,Vy0])

def Fs(X,Y,Vx,Vy):
    return np.array([Vx,Vy,-X/(X**2+Y**2)**(3/2),-Y/(X**2+Y**2)**(3/2)])

print('Sus condiciones iniciales son: ' + str(U0))

print('Defina el paso de tiempo para la integración')

dt = float(input())

print('Defina cuántos segundos quiere que dure la simulación')

t = float(input())
n = int(t/dt)
Dt = np.linspace(0,t,n)

print('Por último, escoja el método de integración para la órbita escribiendo el número de la opción:')
print('1- Euler')
print('2- Crank')
print('3- Runge Kutta de orden 4')
```

Figura 4.1: Declaración previa

En esta primera parte del código se declaran las condiciones iniciales del vector de estado $\vec{U}$, se define una función para facilitar el cálculo de la función de estado $\vec{F}(U)$ y se pregunta al usuario tanto cuánto quiere que sea el paso de integración como cuánto dure la simulación. Además se le da la opción de elegir el esquema numérico con el que se integrará el problema.

4.1 Método de Euler

El método de Euler es el esquema numérico más sencillo, pues es de primer orden y considera que $F(u,t)$ es constante al evaluar la integral entre t_n y t_{n+1}:

$$\vec{U}^{n+1} = \vec{U}^n + (t_{n+1} - t_n) \cdot \vec{F}^n) \tag{4.1.1}$$

El código implementado para llevar a cabo esta integración se muestra en la figura 4.2

```
45      print('Has elegido Euler')
46
47      F0 = Fs(X0,Y0,Vx0,Vy0)
48      F  = F0
49      U  = U0
50
51      for i in range(1,n+1):
52
53          U = U + dt*F
54          [X,Y,Vx,Vy] = U
55          F = Fs(X,Y,Vx,Vy)
56
57          Xplot = [X0] + [U[0]]
58          Yplot = [Y0] + [U[1]]
59
60          plt.title('Órbita con Euler con dt = '+ str(dt) + ' y ' + str(t) + ' segundos de integración')
61          plt.xlabel("X")
62          plt.ylabel("Y")
63          plt.plot(Xplot,Yplot,'b.')
```

Figura 4.2: Código de Euler

4.2 Crank-Nicolson

El esquema numérico de Crank-Nicolson se caracteriza por ser un esquema implícito de un paso y de segundo orden (lo cual ya es una mejora respecto al primer orden de Euler). Se puede expresar de manera general como:

$$U^{n+1} = U^n + \frac{\Delta t_n}{2}(F^{n+1} + F^n) \tag{4.2.1}$$

El código implementado para llevar a cabo este esquema numérico se muestra en la figura 4.3

```
elif choice == 2:

    print('Has elegido Crank')

    F0 = Fs(X0,Y0,Vx0,Vy0)
    F  = F0
    U  = U0
    def sistema(Un1):

        x,y,z,s = Un1
        return(x - U[0] - dt/2*U[2] - dt/2*z,
               y - U[1] - dt/2*U[3] - dt/2*s,
               z - U[2] -dt/2*(-U[0]/(U[0]**2 + U[1]**2)**(3/2)) - dt/2*(-x/(x**2 + y**2)**(3/2)),
               s - U[3] -dt/2*(-U[1]/(U[0]**2 + U[1]**2)**(3/2)) - dt/2*(-y/(x**2 + y**2)**(3/2)))


    for i in range(1,n+1):

        x,y,z,s = fsolve(sistema, (U[0],U[1],U[2],U[3]))

        U = np.array([x,y,z,s])

        Xplot = [X0] + [U[0]]
        Yplot = [Y0] + [U[1]]

        plt.title('Órbita con Crank-Nicolson con dt = '+ str(dt) + ' y ' + str(t) + ' segundos de integración')
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.plot(Xplot,Yplot,'bo')
```

Figura 4.3: Código de Crank-Nicolson

4.3 Método de Runge-Kutta de cuarto orden

Con el método de Runge-Kutta de cuarto orden ya se da un salto en la precisión, ya que pasamos a un equema explícito de cuatro etapas y de cuarto orden. Este esquema puede describirse de manera general de la siguiente manera:

$$\begin{aligned} k_1 &= F(U^n; t_n) \\ k_2 &= F(U^n + \Delta t_n k_1/2; t_n + \Delta t_n/2) \\ k_3 &= F(U^n + \Delta t_n k_2/2; t_n + \Delta t_n/2) \\ k_4 &= F(U^n + \Delta t_n k_3; t_n + \Delta t_n) \\ U^{n+1} &= U^n + \frac{\Delta t_n}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{4.3.1}$$

El código implementado para llevar a cabo este esquema numérico se muestra en la figura 4.4

```
elif choice == 3:
    print('Has elegido Runge Kutta de orden 4')

    F0 = Fs(X0,Y0,Vx0,Vy0)
    F  = F0
    U  = U0

    for i in range(1,n+1):

        [X,Y,Vx,Vy] = U
        F  = Fs(X,Y,Vx,Vy)
        k1 = F
        k2 = Fs(X + k1[0]*dt/2,Y + k1[1]*dt/2,Vx + k1[2]*dt/2,Vy + k1[3]*dt/2)
        k3 = Fs(X + k2[0]*dt/2,Y + k2[1]*dt/2,Vx + k2[2]*dt/2,Vy + k2[3]*dt/2)
        k4 = Fs(X + k3[0]*dt,Y + k3[1]*dt,Vx + k3[2]*dt,Vy + k3[3]*dt)
        U  = U + (dt/6)*(k1+2*k2+2*k3+k4)

        Xplot = [X0] + [U[0]]
        Yplot = [Y0] + [U[1]]

        plt.title('Órbita con RK4 con dt = '+ str(dt) + ' y ' + str(t) + ' segundos de integración')
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.plot(Xplot,Yplot,'bo')
```

Figura 4.4: Código de Runge-Kutta de cuarto orden

5 Resultados

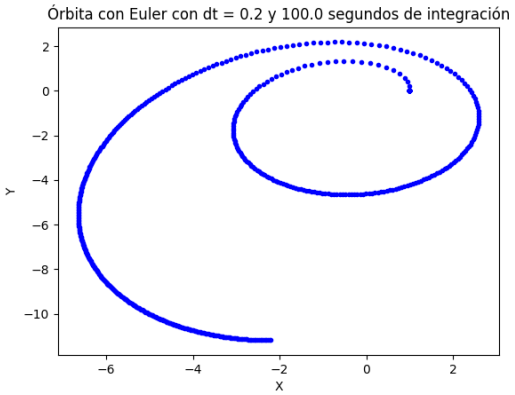
Una vez presentado el código, se va a pasar a comparar los distintos resultados obtenidos por cada código variando el paso de integración. Todos estos resultaros se han obtenido para un tiempo de simulación de 100 segundos y los siguientes valores de paso de tiempo:

Tabla 5.1: Valores del paso de tiempo empleados

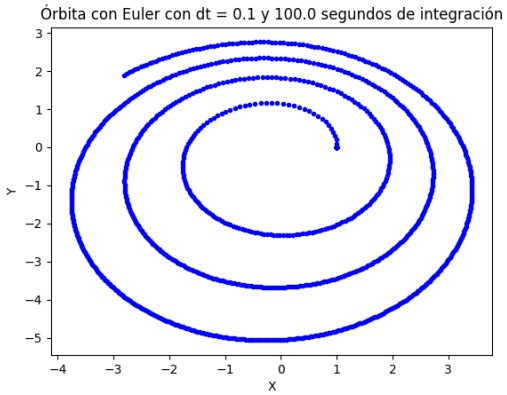
$\Delta t(s)$			
0.2	0.1	0.05	0.01

5.1 Euler

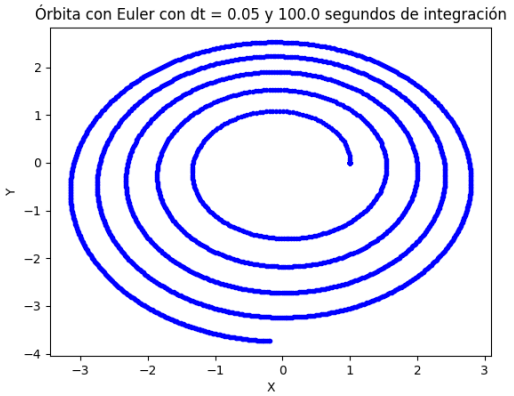
Los resultados obtenidos por el esquema numérico de Euler se presentan en la figura 5.1.



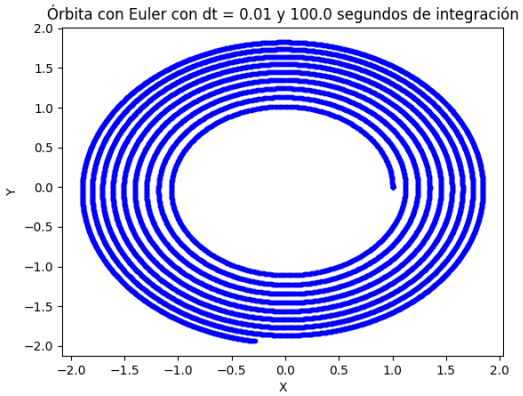
(a) Resultados obtenidos para un $dt = 0.2s$



(b) Resultados obtenidos para un $dt = 0.1s$



(c) Resultados obtenidos para un $dt = 0.05s$



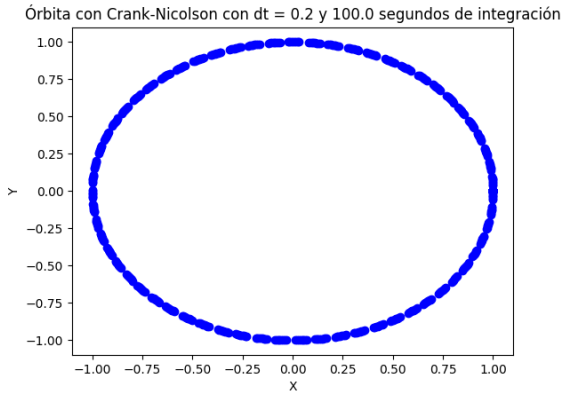
(d) Resultados obtenidos para un $dt = 0.01s$

Figura 5.1: Resultados obtenidos con el esquema numérico de Euler

Como se puede observar en la figura 5.1, si bien es cierto que cuanto menor es el paso de integración mejor es la solución, siempre se podrá apreciar una divergencia con respecto a la solución analítica. Esta divergencia es debida a que el radio espectral del esquema numérico de Euler es mayor que uno, por lo que el error se va acumulando.

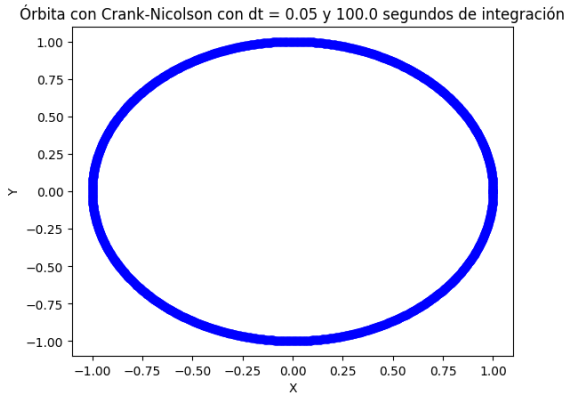
5.2 Crank-Nicolson

Los resultados obtenidos por el esquema numérico de Crank-Nicolson se presentan en la figura 5.2.



(a) Resultados obtenidos para un $dt = 0.2s$

(b) Resultados obtenidos para un $dt = 0.1s$



(c) Resultados obtenidos para un $dt = 0.05s$

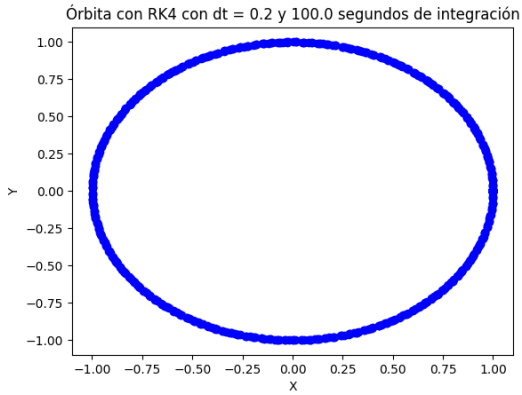
(d) Resultados obtenidos para un $dt = 0.01s$

Figura 5.2: Resultados obtenidos con el esquema numérico de Crank-Nicolson

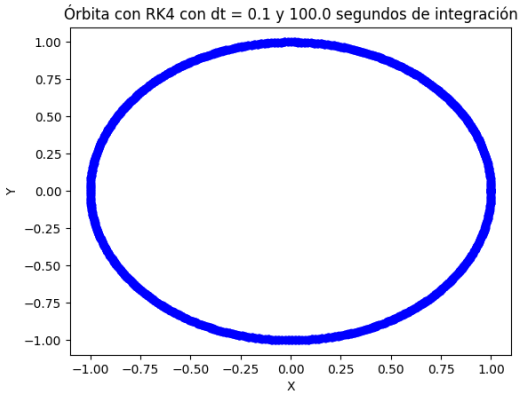
Se puede apreciar que el resultado obtenido con este esquema numérico es bastante mejor que el obtenido con Euler, esto se debe a que es de segundo orden. Por otro lado, la variación del valor de paso de integración no tiene ningún efecto en la solución más allá de que hay más puntos definidos.

5.3 Runge-Kutta de cuarto orden

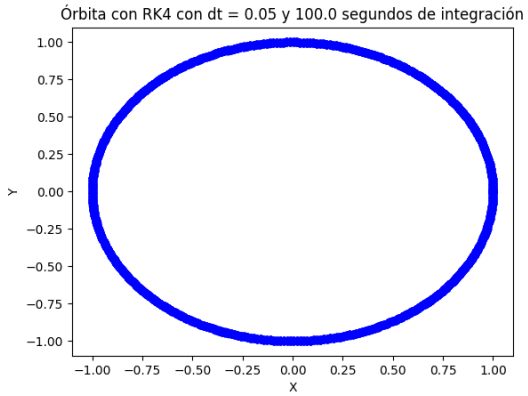
Los resultados obtenidos por el esquema numérico de Runge-Kutta de cuarto orden se presentan en la figura 5.3.



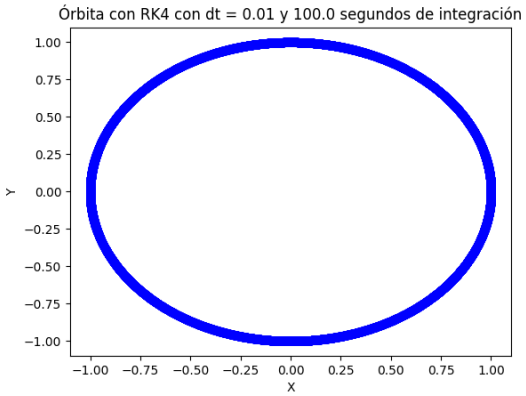
(a) Resultados obtenidos para un $dt = 0.2s$



(b) Resultados obtenidos para un $dt = 0.1s$



(c) Resultados obtenidos para un $dt = 0.05s$



(d) Resultados obtenidos para un $dt = 0.01s$

Figura 5.3: Resultados obtenidos con el esquema numérico de Runge-Kutta de cuarto orden

De la misma manera que con Crank-Nicolson, se puede observar que este esquema numérico es bastante mejor que el obtenido con Euler, esto se debe a que es de cuatro etapas y de cuarto orden y por lo tanto mayor precisión. Además, la variación del valor de paso de integración no tiene ningún efecto en la solución más allá de que hay más puntos definidos.