


Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Milestone 3

Ampliación de Matemáticas I

16 de octubre de 2022

Autor:

- Sergio Cavia Fraile

Índice

1. Introducción

El objetivo del trabajo será comparar los errores y los ratios de convergencia de distintos esquemas temporales a la hora de integrar órbitas de Kepler.

2. Ecuaciones y Métodos

2.1. Extrapolación de Richardson

Los errores de los métodos numéricos mediante la extrapolación de Richardson se calculan mediante la siguiente ecuación:

$$E = \frac{U^{2N} - U^N}{1 - \frac{1}{2^q}}$$

Donde q es el orden del esquema temporal.

2.2. Ratio de convergencia

Los ratios de convergencia de los métodos numéricos se calculan mediante la siguiente ecuación:

$$RC = U^{2N} - U^N$$

Para una mejor visualización, en las gráficas se representará el logaritmo de RC respecto al logaritmo de N, siendo N el número de subdivisiones del intervalo temporal.

3. Código

En primer lugar se han definido 2 funciones, una para calcular el error y otra para calcular el ratio de convergencia.

Y posteriormente se ha escrito el código que dará el tiempo final, el paso temporal y las condiciones iniciales y calculará el error y el ratio de convergencia.

3.1. Error

El error se ha implementado mediante las siguientes líneas de código:

```
"Error según Richardson"

def Error(F, t, U0, Scheme):

    N = size(t)
    E = zeros([N,size(U0)])

    t1 = t
    t2 = linspace(0, t[N-1], N*2)

    #Calculo soluciones con dos pasos distintos
    U2 = Cauchy(F, t2, U0, Scheme)
    U1 = Cauchy(F, t1, U0, Scheme)

    #Defino el orden del esquema
    if Scheme == RK4:
        q = 4
    elif Scheme == CN:
        q = 2
    else:
        q = 1

    #Calculo el error
    for i in range(0,N):
        E[i,:] = (U2[2*i,:] - U1[i,:]) / (1 - 1 / (2**q))

    return E
```

3.2. Ratio de convergencia

El ratio de convergencia se ha implementado mediante las siguientes líneas de código:

```
"Ratio de convergencia"

def Conv(F, t, U0, Scheme,p):

    #p es el numero de puntos de la gráfica que se van a calcular

    #Calculo la primera solución con N divisiones
    N = size(t)
    tf = t[N-1]
    U = Cauchy(F, t, U0, Scheme)

    #Inicializo los vectores
    E = zeros(p)
    Elog = zeros(p)
    Nlog = zeros(p)
    N = 2*N


    for i in range(0,p):

        #Calculo la solución con paso mitad
        t2 = linspace(0, tf, (2**i)*N)
        U2N = Cauchy(F, t2, U0, Scheme)


       #Calculo el ratio de convergencia y su logaritmo
       E[i] = norm((U2N[int((2**i)*N-1),:] - U[int((2**i)*N/2-1),:]))
       Elog[i] = log10(E[i])
       Nlog[i] = log10((2**i)*N)

       U = U2N

    return [Elog, Nlog]
```

3.3. Código final

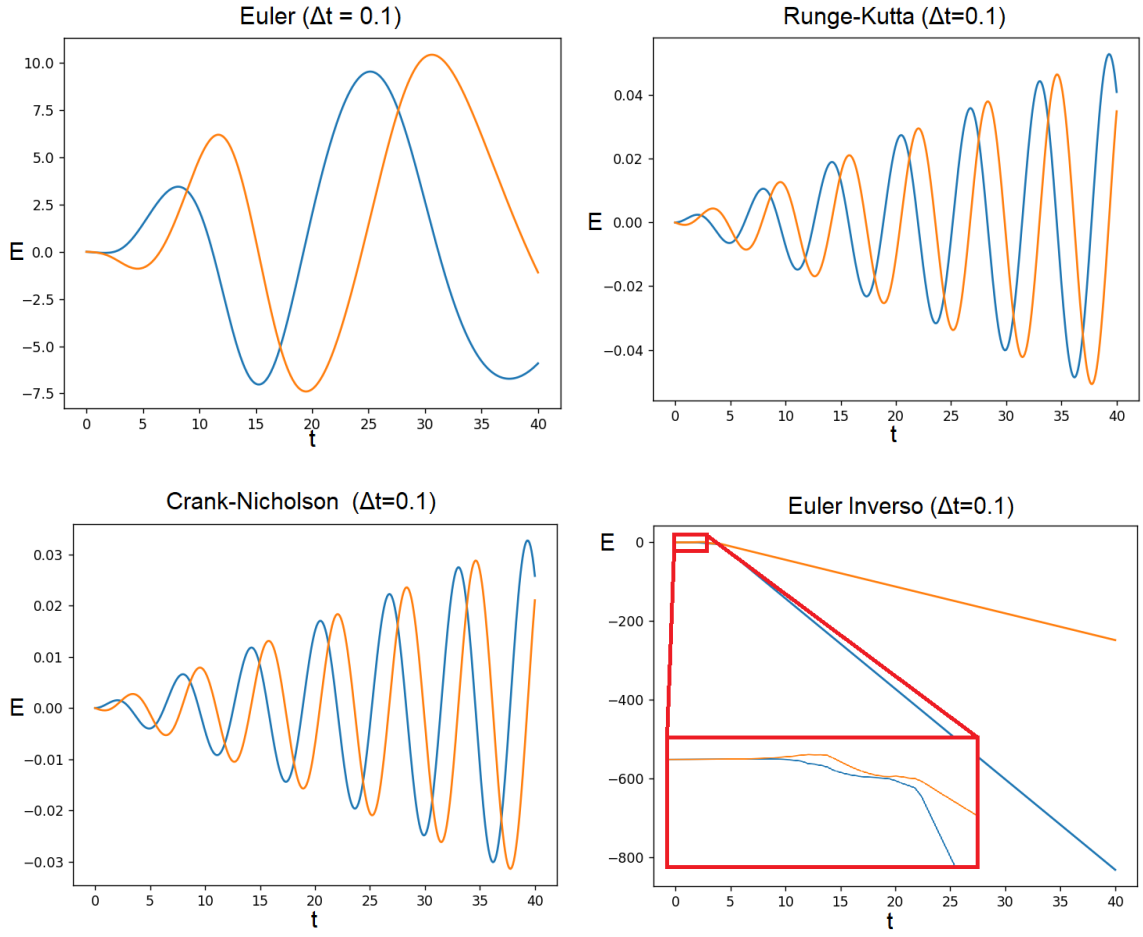
El código que realmente da valores y calcula los errores y ratios de convergencia es el siguiente:

```
"Run"

#Defino tiempo final, numero divisiones y conodiciones iniciales
tf = 20
N = 100
t = linspace(0, tf, N)
U0 = array([1, 0, 0, 1])

#Calculo el error
E = Error(Kepler, t, U0, CN)
#Lo grafico
plt.plot(t, E[:,0])
plt.show()

#Calculo los logaritmos de los ratios de convergencia y numero de subdivisiones
[log_E, log_N] = Conv(Kepler, t, U0, CN, 8)
#Lo grafico
plt.plot(log_N, log_E)
plt.show()
```

4. Resultados

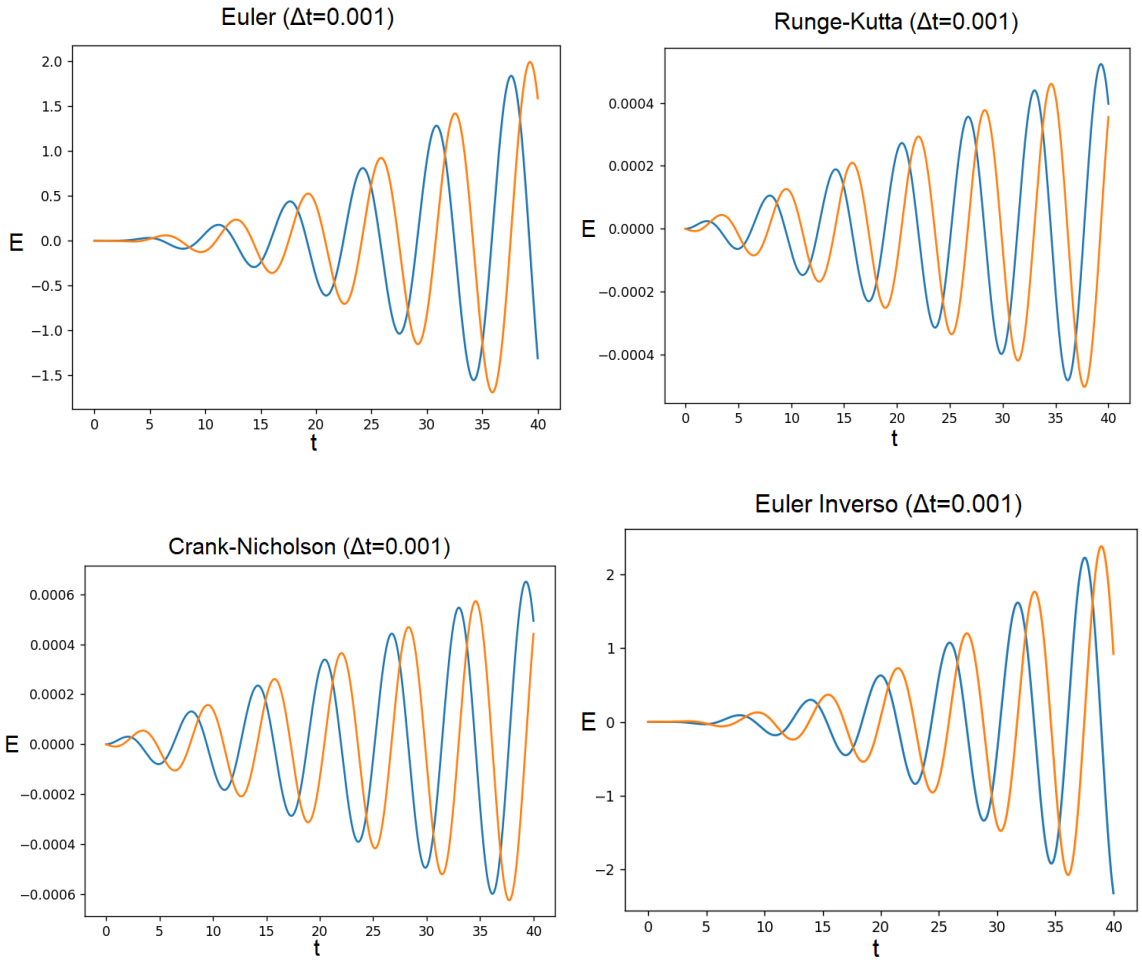
4.1. Error

En las siguientes gráficas se pueden observar los errores en X (azul) e Y (naranja) de cada uno de los métodos al simular 40 unidades de tiempo con un intervalo temporal de 0.1 unidades de tiempo:



Como podemos ver, el Runge-Kutta de orden 4 y el Crank-Nicholson tienen errores similares y muy bajos (siendo menores en el Cranck-Nicholson). El método de Euler tiene mayores errores, y el Eules inverso diverge tras algunas iteraciones por haber usado un intervalo temporal demasiado grande que ha provocado alguna división entre cero.

Para solucionar esto, y tener una mejor vista de los 4 métodos se decide disminuir en dos órdenes de magnitud el intervalo temporal hasta 0.001 unidades de tiempo. Con ello se obtienen las siguientes gráficas:

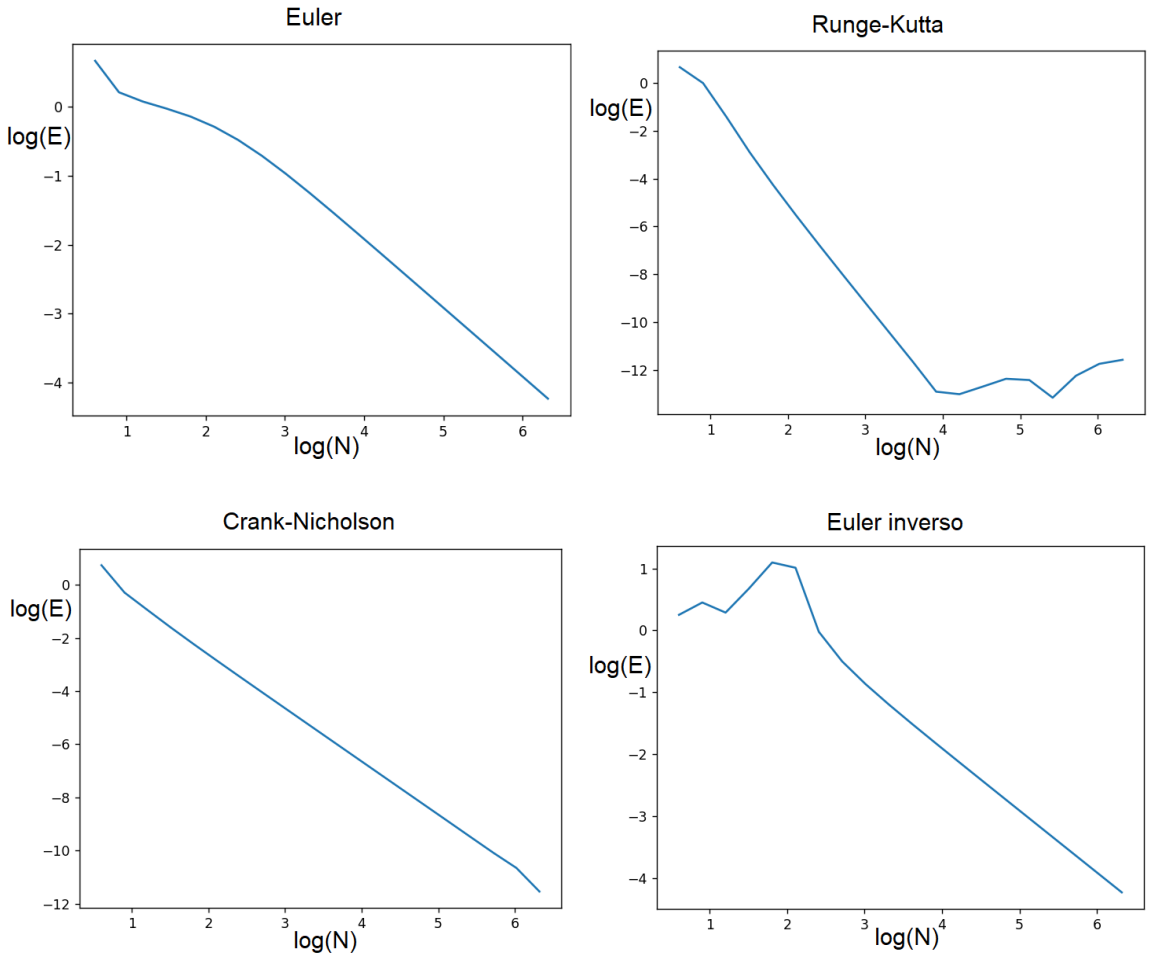


Con estas gráficas ya sí que podemos comparar el Euler inverso con el resto de métodos. En ellas vemos como los errores de los dos métodos de Euler son muy similares, siendo algo mayor en el caso del Euler inverso.

En estas gráficas también podemos observar como para un intervalo temporal menos, el error del Crank-Nicholson ha superado al del Runge-Kutta de orden 4 pese a ser ambos muy bajos y similares.

4.2. Ratio de convergencia

Para el ratio de convergencia se han graficado 20 puntos incrementando el número de subdivisiones N en potencias de 2, empezando en 2 y acabando en más de 1.000.000 de subdivisiones y manteniendo el tiempo final en 2 unidades temporales para obtener las siguientes gráficas:



Como podemos ver, el mejor método es el Runge-Kutta de orden4 que para $N \sim 10^4$ ya converge a un error menor que el generado por el propio ordenador $E_{ordenador} \sim 10^{-12}$. Cranck Nicholson también converge al error de la máquina pero lo hace cuando $N \sim 10^6$.

El esquema de Euler converge para altos N, pero no llega a valores tan bajos como los métodos anteriormente comentados. Euler inverso logra un resultado muy similar para altos N, sin embargo, la parte izquierda de la curva, cuando los intervalos temporales son grandes, muestra grandes irregularidades, por lo que el método no converge hasta que $N \sim 10^3$.

Las gráficas coinciden con lo visto en las clases de Ampliación de Matemáticas I, ya que las curvas son lineales siempre y cuando el intervalo temporal no sea demasiado pequeño o se consiga un error por debajo de 10^{-12}, donde no se puede conseguir un error menor que el de la propia máquina.