

Milestone 3

Ampliación de Matemáticas 1

19 de octubre de 2022

Javier Garrido Castillo

I. Introducción

En este tercer hito de la asignatura, se debe realizar una estimación del error de cada una de las soluciones numérica desarrolladas en hitos anteriores. Para ello, se definirá una función que evaluará el error de las integraciones numéricas a partir de la extrapolación de Richardson. Se empleará para estudiar los esquemas temporales: Euler, Euler Inverso, Crank Nicolson y Runge-Kutta de orden 4. A continuación, se definirá una función para evaluar el ratio de convergencia de los distintos esquemas numéricos mencionados y compararlo con diferentes pasos de tiempo.

II. Problema

A. Cálculo error numérico por extrapolación de Richardson

Se define el error de una solución numérica como la diferencia entre la solución exacta y la solución aproximada, ambas en el mismo instante de tiempo. Este error se estima a partir del método de Richardson como:

$$E = \frac{U_1^n - U_2^n}{1 - 1/2^q}$$

Donde, E es el error estimado, U_1^n es la solución calculada para el final del paso de tiempo dado, U_2^n la solución calculada en el tiempo final con la mitad del paso de tiempo dado, y q es el orden del esquema temporal empleado. Los esquemas temporales empleados poseen los siguientes ordenes: 1 para Euler, 1 para Euler Inverso, 2 para Crank Nicolson y 4 para Runge-Kutta de orden 4.

B. Ratio de convergencia de esquemas temporales

En este segundo apartado del problema del hito 3, se estudiará el ratio de convergencia para los distintos esquemas temporales. Para ello, se representará la siguiente función para cada uno de ellos:

$$\log\|E^n\| = C - q \log N$$

Un esquema numérico será de orden q si su error numérico es del orden de Δt^q , esto significa que si el paso del tiempo escogido es lo suficientemente pequeño, el error tenderá a cero con la misma velocidad que Δt^q . Al representar esta función para cada caso, deberá aparecer una línea de pendiente negativa q, que será el orden del esquema correspondiente. Para ello, será necesario calcular el error mediante la extrapolación de Richardson explicada anteriormente.

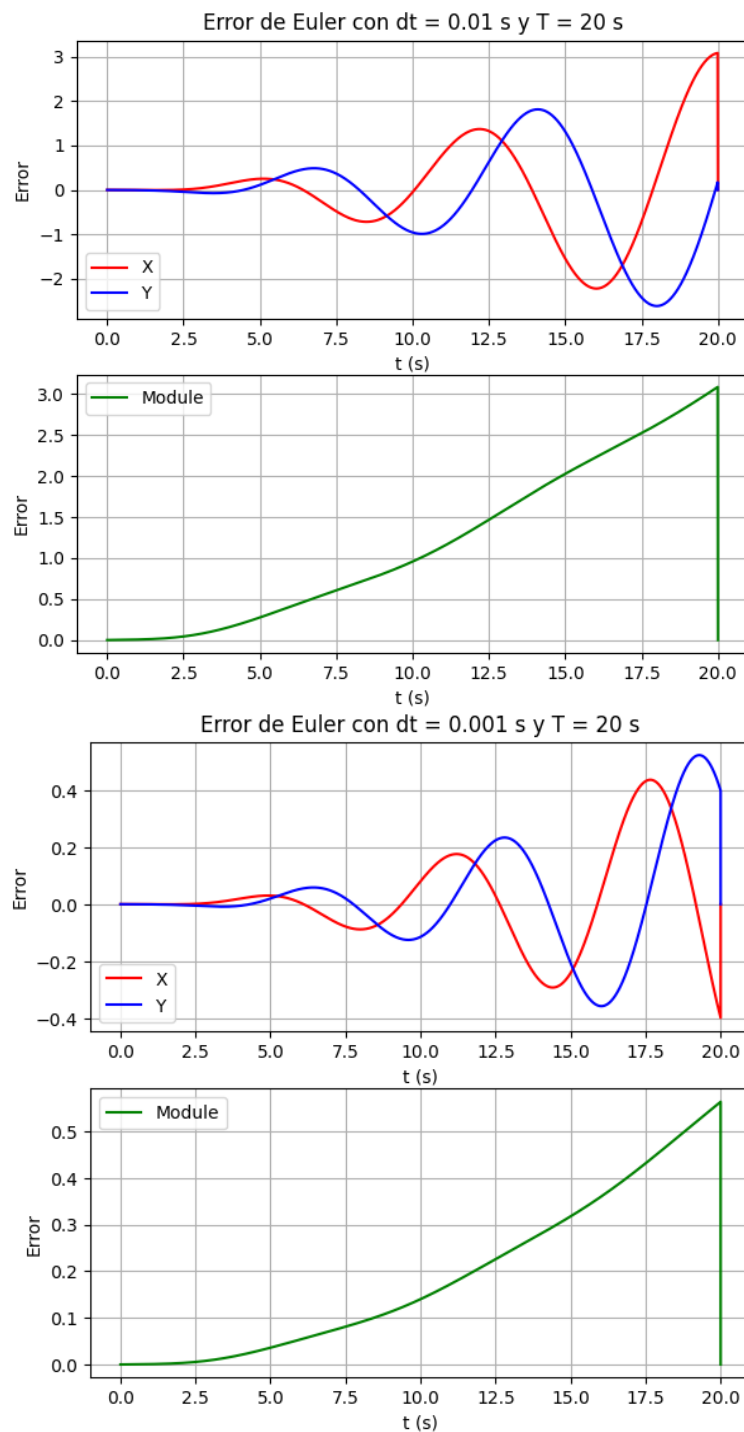
III. Resultados

Para la obtención de resultados se han empleado los siguientes valores de paso de tiempo:

$\Delta t(s)$	
0.01	0.001

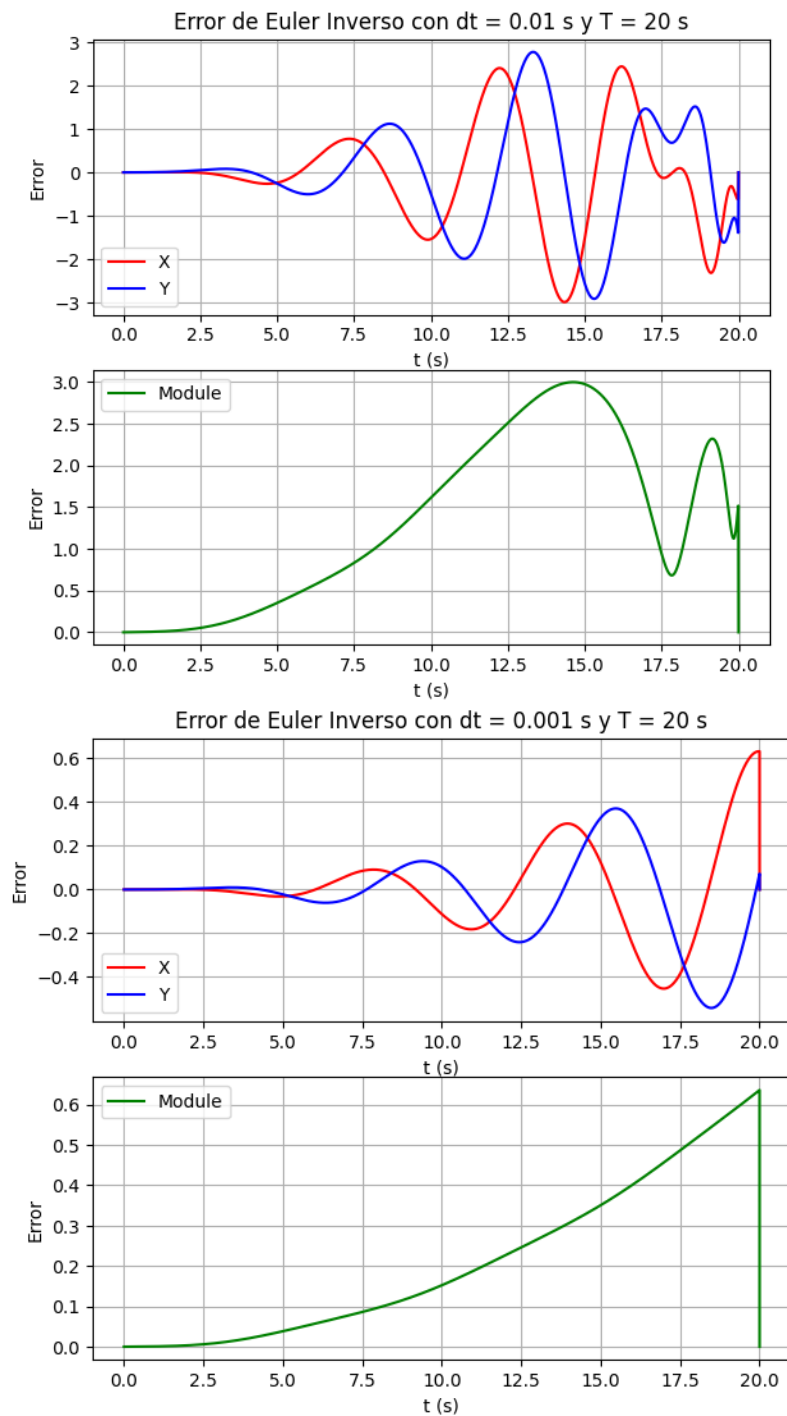
A. Cálculo error numérico por extrapolación de Richardson

1. Esquema Euler explícito



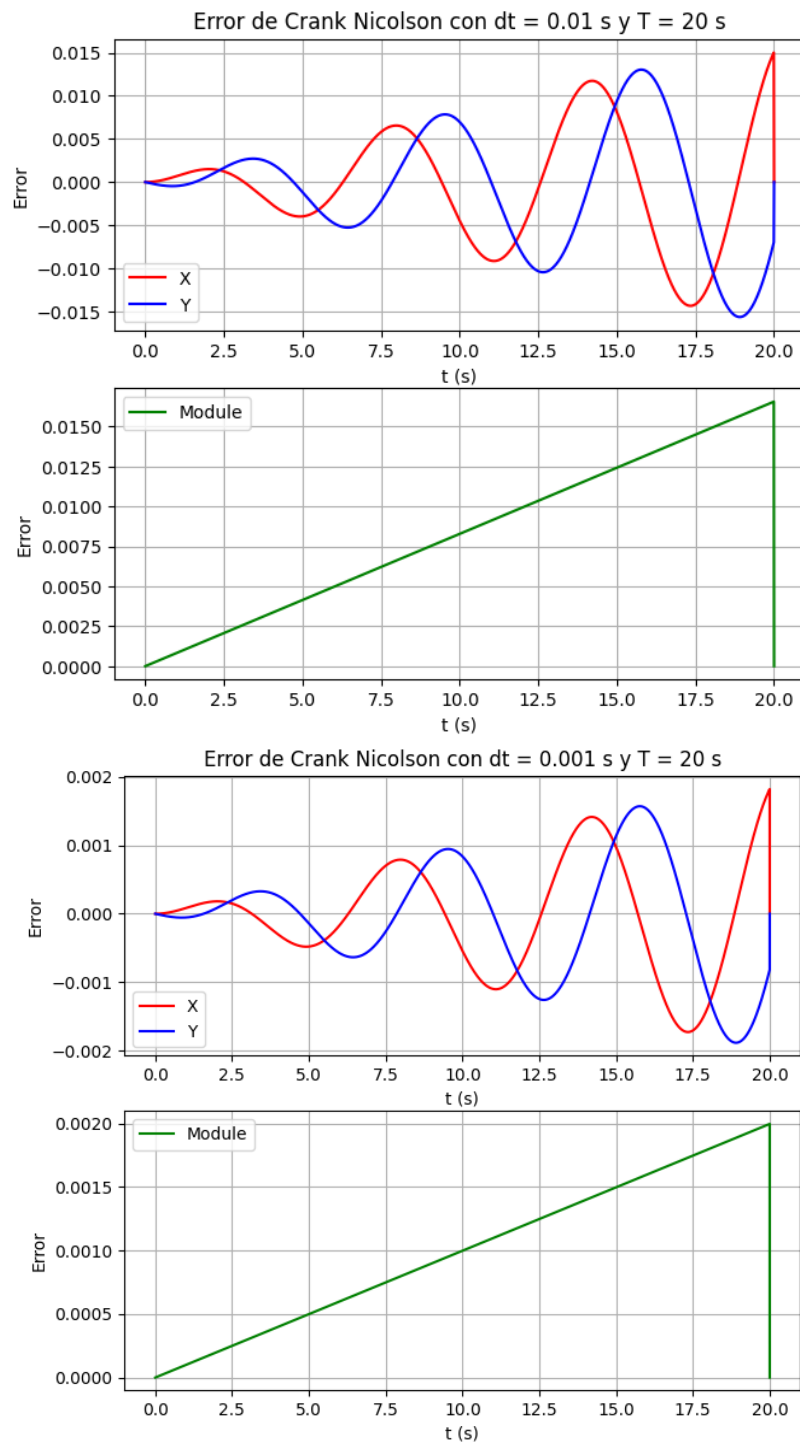
Como se observa en las figuras anteriores, el error del esquema de Euler se reduce al disminuir el paso de integración. En cuanto al módulo del error, es una curva y no una recta, aumentando con el tiempo final de integración.

2. Esquema Euler Inverso



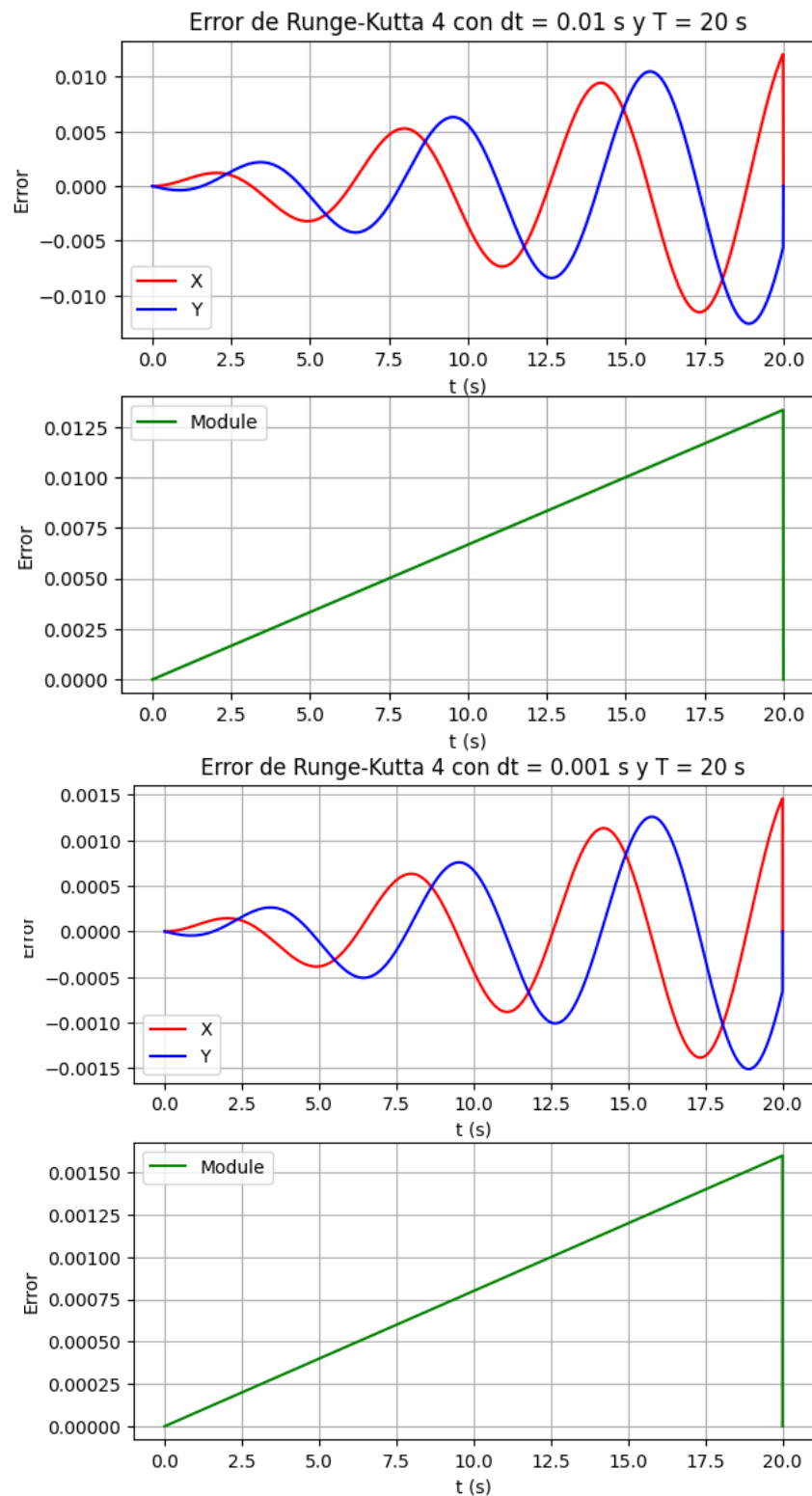
Este esquema presenta unos resultados que difieren notablemente de los del Euler explícito. Variando claramente la representación del módulo del error para el mayor de los pasos del tiempo, así como las dos componentes del error. Sin embargo, al reducir el paso del tiempo los resultados obtenidos se aproximan más al Euler explícito.

3. Esquema Crank-Nicolson



Se observa claramente que los resultados obtenidos con este esquema temporal son notablemente mejores que los obtenidos a partir del Euler, Crank Nicolson presente un error de dos ordenes de magnitud menor para tiempos de simulación y paso de tiempo iguales. En este caso, además, el módulo del error se representa como una recta de pendiente positiva.

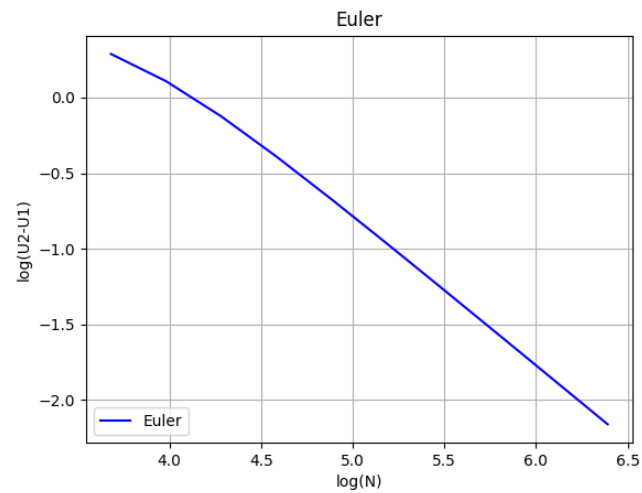
4. Esquema Runge-Kutta orden 4



Al igual que lo comentado en el Crank Nicolson, el Runge-Kutta de orden 4 presenta un error mucho menor que el de Euler debido a su mayor orden que aumenta la precisión de los resultados obtenidos. Siendo estos más exactos al igual que en todos los anteriores cuanto menor es el paso de tiempo escogido. De nuevo, el módulo del error se representa con una recta de pendiente positiva.

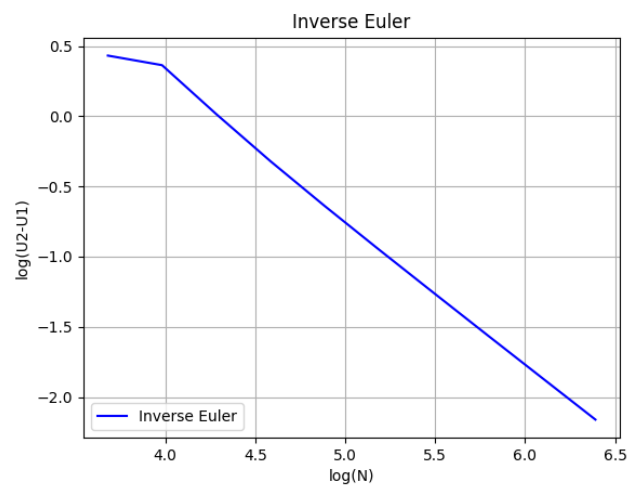
B. Ratio de convergencia de esquemas temporales

1. Esquema Euler explícito



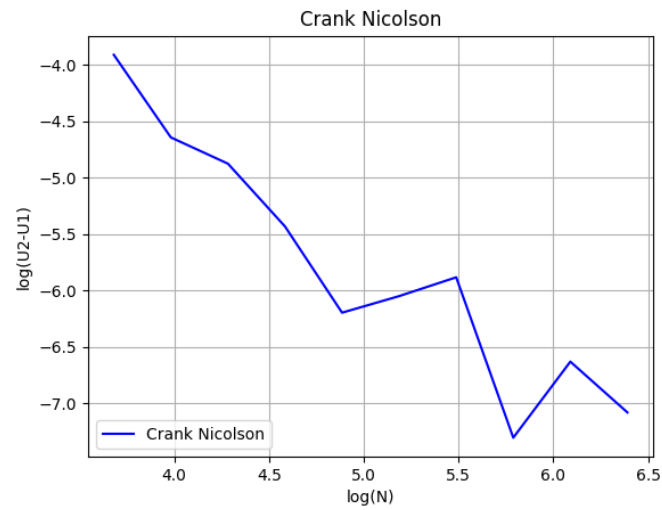
Se comprueba de nuevo fácilmente, que al disminuir el Δt la representación se acerca más a una línea completamente recta de pendiente negativa.

2. Esquema Euler implícito



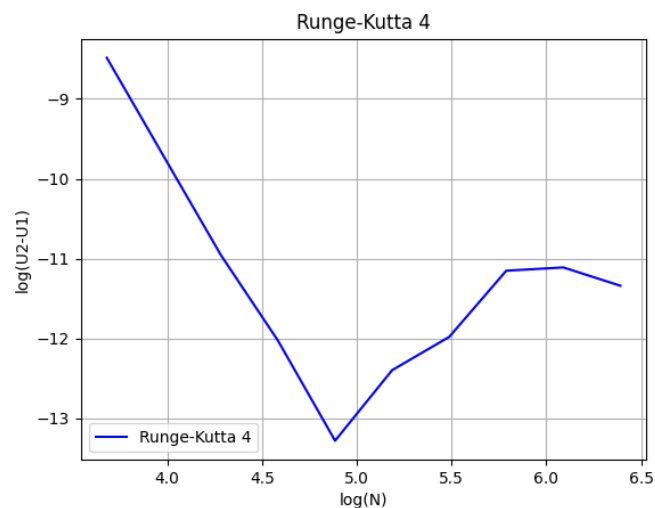
De nuevo, se repite que se obtiene una mejor convergencia, acercándose la representación a una recta, cuanto menor es el paso del tiempo escogido.

3. Esquema Crank Nicolson



Este esquema temporal de nuevo, el error es muy pequeño a pesar de no representarse como una línea recta.





4. Esquema Runge-Kutta orden 4



En este caso, los resultados obtenidos son también correctos, a pesar de no observarse una recta completa y es que el valor del error es muy pequeño.

IV. Código

En este tercer hito, se ha implementado de nuevo la estructura Top-Down, organizando las funciones creadas en distintos archivos en base a su funcionalidad. Se emplea de nuevo la carpeta de "Functions" donde se encuentran los distintos archivos que reúnen las funciones empleadas en este hito:

-  Cauchy_Problem
-  Error
-  Physics
-  Temporal_Schemes

De esta forma, desde el código principal “Milestone3”, se importan las funciones de los distintos archivos que se emplean en este hito.

Los archivos de “Cauchy_Problem”, “Physics” y “Temporal_Schemes” ya se mostraron en el hito 2. En este caso, se ha desarrollado un nuevo archivo donde incluiremos las funciones empleadas en este hito 3, “Error”. Se definen las funciones de “Richardson” donde se calcula el error a partir del método de Richardson y “Convergence_Rate” donde se calcula el ratio de convergencia.

En “Richardson”, se deben introducir como inputs: el esquema temporal empleado, la función F de Kepler, el tiempo, la condición inicial y el orden del esquema temporal. Por otra parte, en “Convergence_Rate” se introducen como inputs: el esquema temporal empleado, la función F de Kepler, el tiempo y la condición inicial.

```

1  from Functions.Cauchy_Problem import Cauchy_problem
2  from numpy import linspace, zeros, size, log10
3  from numpy.linalg import norm
4
5  def Richardson(Temporal_Scheme, F, t, U0, order):
6
7      N1 = len(t) - 1
8      N0 = len(U0)
9      Error = zeros((N1+1, N0))
10     t1 = t
11     t2 = linspace(0, t1[N1-1], 2*size(t1))
12
13     U1 = Cauchy_problem(Temporal_Scheme, F, t1, U0)
14     U2 = Cauchy_problem(Temporal_Scheme, F, t2, U0)
15
16     for i in range(N1):
17         Error[i,:] = (U2[2*i,:] - U1[i,:]) / (1 - 1/(2**order))
18
19     return Error
20
21  def Convergence_Rate(Temporal_Scheme, F, t1, U0):
22
23     N2 = size(t1)
24     T = t1[N2-1]
25     m = 10
26     U1 = Cauchy_problem(Temporal_Scheme, F, t1, U0)
27
28     log_E = zeros(m)
29     log_N = zeros(m)
30
31     for i in range(m):
32
33         N2 = 2*N2
34         t2 = linspace(0,T,N2)
35         U2 = Cauchy_problem(Temporal_Scheme, F, t2, U0)
36
37         log_E[i] = log10(norm(U2[int(N2-1),:] - U1[int(N2/2-1),:]))
38         log_N[i] = log10(N2)
39
40         t1 = t2
41         U1 = U2
42
43
44     return [log_E, log_N]

```

Finalmente, el código para este tercer hito, “Milestone3”, se encarga de importar las distintas funciones empleadas, a partir de los códigos mostrados anteriormente, y definir las condiciones del problema: N° pasos, tiempo final, deltat y condición inicial de la solución. Finalmente, se emplean las funciones y se representan las distintas soluciones para cada uno de los esquemas temporales obteniendo los resultados previamente expuestos para cada uno de los apartados de este hito.

```

from Functions.Physics import Kepler
from Functions.Temporal_Schemes import Euler, Crank_Nicolson, RK4, Inverse_Euler
from Functions.Error import Richardson, Covergence_Rate
import matplotlib.pyplot as plt
from numpy import array, linspace, zeros, size, hstack

N = 20000 # N° pasos
tf = 20# valor final
dt = tf/N
print(dt)

# CONDICIONES INICIALES
r0 = array([1, 0])
v0 = array([0, 1])

U0 = hstack((r0,v0)) # Initial condition

t = linspace(0, tf, N)

Error_Euler = Richardson(Euler, Kepler, t, U0, 1)
Norm_Euler = zeros(size(Error_Euler[:,0]))
for i in range(0,size(Error_Euler[:,0])):
    Norm_Euler[i] = (Error_Euler[i,0]**2 + Error_Euler[i,1]**2)**(1/2)

Error_InvEuler = Richardson(Inverse_Euler, Kepler, t, U0, 1)
Norm_InvEuler = zeros(size(Error_InvEuler[:,0]))
for i in range(0,size(Error_InvEuler[:,0])):
    Norm_InvEuler[i] = (Error_InvEuler[i,0]**2 + Error_InvEuler[i,1]**2)**(1/2)

Error_CN = Richardson(Crank_Nicolson, Kepler, t, U0, 2)
Norm_CN = zeros(size(Error_CN[:,0]))
for i in range(0,size(Error_CN[:,0])):
    Norm_CN[i] = (Error_CN[i,0]**2 + Error_CN[i,1]**2)**(1/2)

Error_RK4 = Richardson(RK4, Kepler, t, U0, 4)
Norm_RK4 = zeros(size(Error_RK4[:,0]))
for i in range(0,size(Error_RK4[:,0])):
    Norm_RK4[i] = (Error_RK4[i,0]**2 + Error_RK4[i,1]**2)**(1/2)

[log_E_Euler, log_N_Euler] = Covergence_Rate(Euler, Kepler, t, U0)
[log_E_InvEuler, log_N_InvEuler] = Covergence_Rate(Inverse_Euler, Kepler, t, U0)
[log_E_CN, log_N_CN] = Covergence_Rate(Crank_Nicolson, Kepler, t, U0)
[log_E_RK4, log_N_RK4] = Covergence_Rate(RK4, Kepler, t, U0)

```