Universidad Politécnica de Madrid

Máster Universitario en Sistemas Espaciales

Milestone 4

Ampliación de Matemáticas I

Javier Garrido Castillo

3 de noviembre de 2022

Contents

1 Introducción

En este informa se presentan los resultados obtenidos en el hito 4 de la asignatura de ampliación de matemáticas. Este cuarto hito consiste en el estudio de problemas lineales y regiones de estabilidad absolutas. Para ello, se integrará un oscilador lineal con condiciones iniciales usando Euler, Euler Inverso, Leap-Frog, Crank Nicolson y Runge-Kutta de orden 4. Posteriormente, se estudiará la región de estabilidad de cada uno de estos métodos temporales.

2 Problema

2.1 Integrar el oscilador lineal

Se deberá integrar un oscilador armónico lineal para unas condiciones iniciales dadas. Para ello, se emplea la siguiente expresión del oscilador armónico:

$$\ddot{x} + x = 0 \tag{1}$$

Para nuestro caso, el vector de estado U posee como primera componente la posición (x) y de segunda componente la velocidad ($\dot{x}$). Y como:

$$F(U,t) = \frac{dU}{dt} \tag{2}$$

Las componentes de F(U,t) serán la velocidad ($\dot{x}$) y la segunda la posición cambiada de signo ($-x$). Como condición inicial, se establecerá $U_0 = (1\ 0)$.

2.2 Leap-Frog

Para este hito, es preciso introducir este nuevo esquema numérico. Este se emplea para calcular la solución U_2 para un tiempo t_{n+1} a partir de una solución U_1 en t_n de la siguiente forma:

$$U^{n+1} = U^{n-1} + 2\Delta t F^n \tag{3}$$

Se obtienen con este esquema los puntos intermedios, siendo así un esquema de dos pasos y siendo preciso una condición inicial adicional. Para ello se integra el primer paso con Euler y se procede posteriormente a emplear el Leap-Frog.

2.3 Regiones de estabilidad

Se debe representar la región de estabilidad absoluta de cada uno de los esquemas temporales usados en este Milestone. Esta se define como la región del plano complejo donde las soluciones numéricas son asintóticamente estables. Un método numérico es absolutamente estable para ω si los ceros de $\Pi(z,\omega) = p(z) - \omega q(z)$ (polinomio característico de estabilidad) tiene módulo menor o igual que 1. Cada uno de los esquemas posee su propio polinomio característico:

- Euler: $\Pi(r, \omega) = r - 1 - \omega$
- Euler inverso: $\Pi(r, \omega) = r - \frac{1}{1-\omega}$
- Crank Nicolson: $\Pi(r, \omega) = r - \frac{1+\frac{\omega}{2}}{1-\frac{\omega}{2}}$
- Runge Kutta de orden 4: $\Pi(r, \omega) = r - 1 - \omega - \frac{\omega^2}{2!} - \frac{\omega^3}{3!} - \frac{\omega^4}{4!}$
- Leap-Frog: $\Pi(r, \omega) = r^2 - 1$

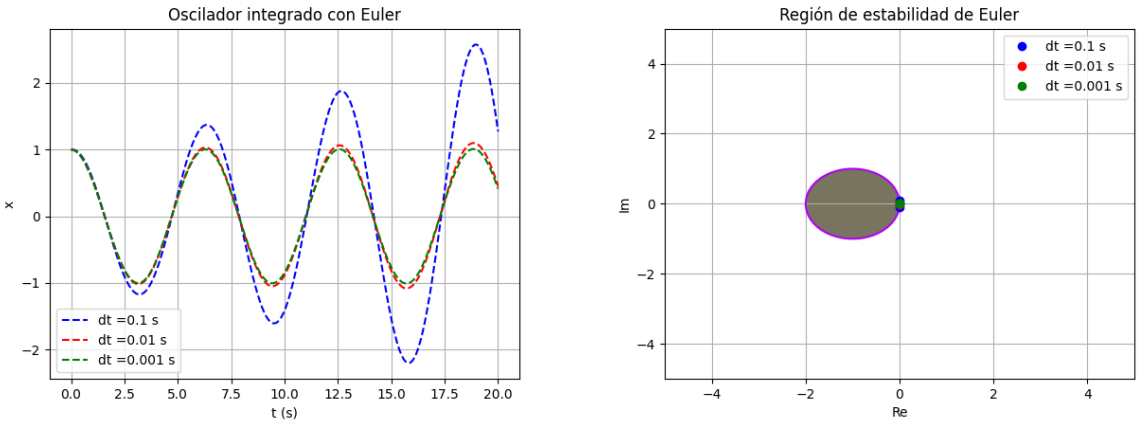
3 Resultados

Para este apartado, se obtendrán resultados para un tiempo total de 20 segundos y diversos pasos de integración. Los dt escogidos son:

$\Delta t(s)$		
0.1	0.01	0.001

Table 1: Pasos de tiempo empleados

3.1 Euler

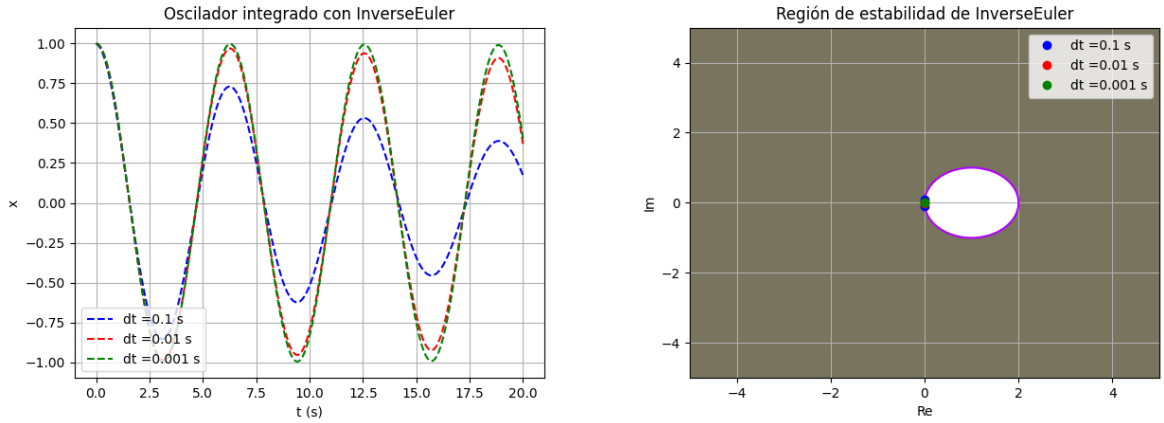


(a) Oscilador integrado para distintos dt

(b) Frontera de la región de estabilidad

Figure 1: Resultados para Euler

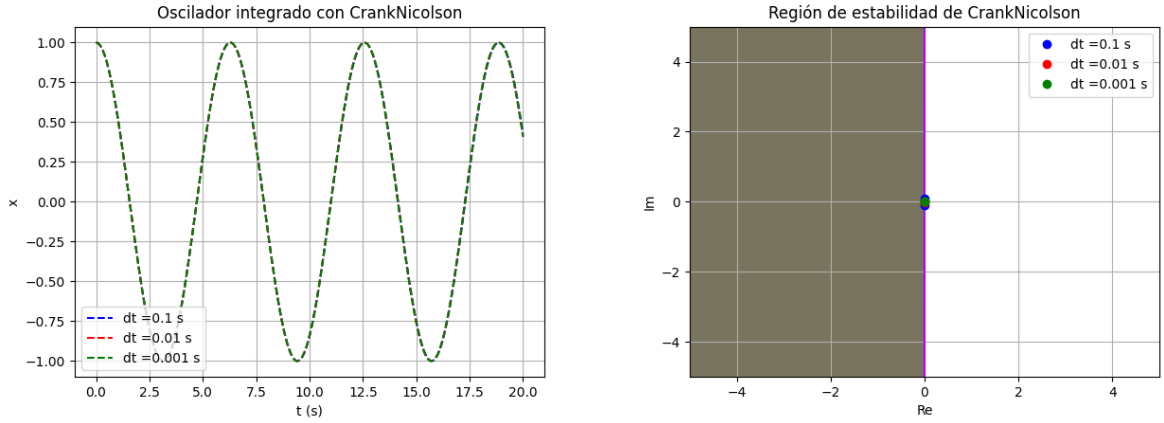
3.2 Euler Inverso



(a) Oscilador integrado para distintos dt (b) Frontera de la región de estabilidad

Figure 2: Resultados para Euler Inverso

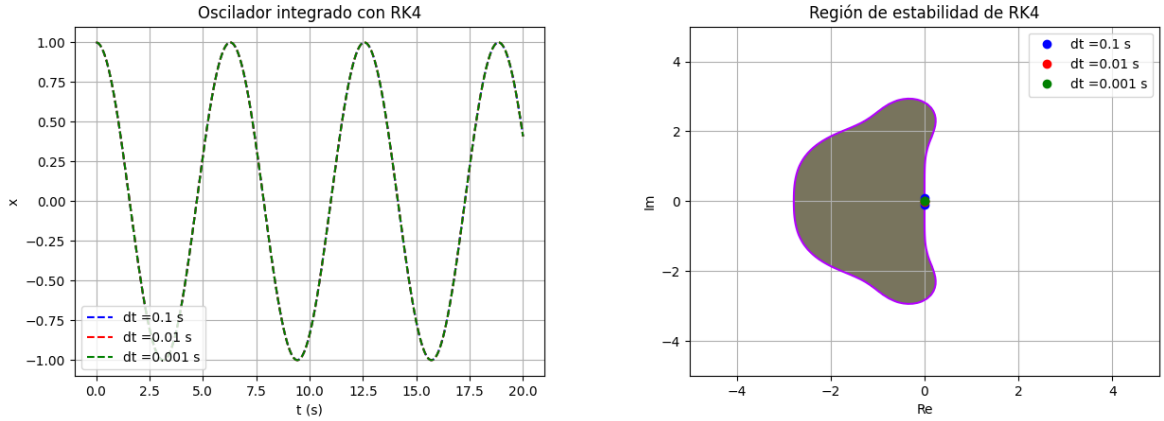
3.3 Crank Nicolson



(a) Oscilador integrado para distintos dt (b) Frontera de la región de estabilidad

Figure 3: Resultados para Crank Nicolson

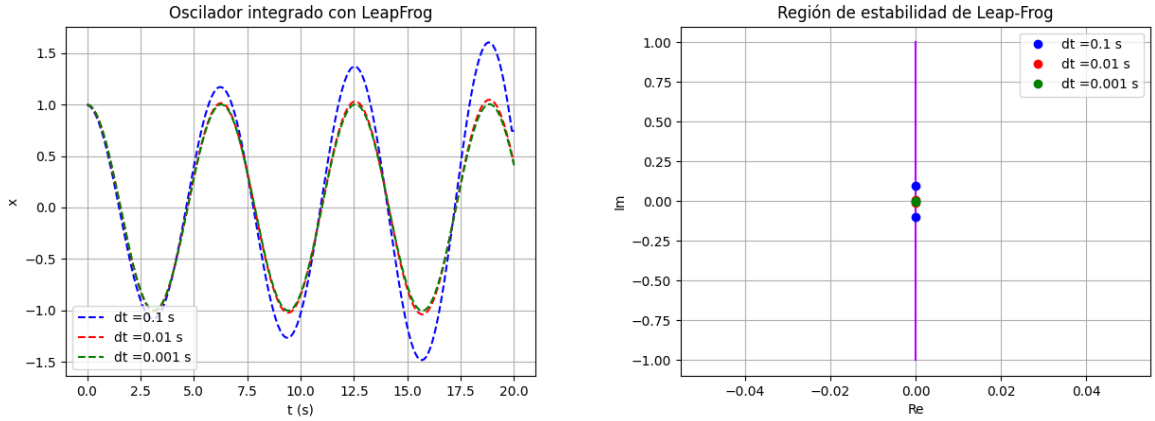
3.4 Runge-Kutta orden 4



(a) Oscilador integrado para distintos dt (b) Frontera de la región de estabilidad

Figure 4: Resultados para Runge-Kutta 4

3.5 Leap-Frog



(a) Oscilador integrado para distintos dt (b) Frontera de la región de estabilidad

Figure 5: Resultados para Leap-Frog

4 Código

Empleando de nuevo la estructura Top-Down seguida en el resto de hitos de la asignatura, se emplea de nuevo la carpeta *Functions* donde se ordenarán las distintas funciones empleadas en este hito. Se emplean de nuevo los archivos de *Physics*, *Temporal_Schemes*, *Error* y *Cauchy_Problem*.

4.1 Leap-Frog

En esta ocasión, se ha implementado un oscilador lineal en el archivo de *Physics* y el esquema numérico de Leap-Frog en *Temporal_Schemes*:

```
def Linear_Oscilator(U, t):
    r = U[0]
    drdt = U[1]
    return array([drdt, -r])
```

```
def LeapFrog(U, dt, t, F):
    p = int(len(U)/2)

    V = U
    A = F(V,t)

    V[p:] += A[p:]*dt/2.0
    V[:p] += A[:p]*dt

    A = F(V,t)

    V[p:] += A[p:]*dt/2.0

    return V
```

4.2 Región de estabilidad

Adicionalmente, para el cálculo de la región de estabilidad, es preciso definir las siguientes funciones en un nuevo archivo denominado *Stability*:

```
from Functions.Temporal_Schemes import Euler, Crank_Nicolson, RK4,
                                       Inverse_Euler, LeapFrog
from numpy import array, zeros, size, sqrt


def Stability_Region(Temporal_Scheme,x,y):
    n = size(x)
    W = zeros([n,n],dtype=complex)

    for i in range(n):
        for j in range(n):
            W[n-1-j,i] = complex(x[i],y[j])

    return absolute(array(Stability_Polynomial(Temporal_Scheme, W)))



def Stability_Polynomial(Temporal_Scheme, w):
    if Temporal_Scheme == Euler:
        r = 1 + w
    elif Temporal_Scheme == Inverse_Euler:
        r = 1/(1-w)
    elif Temporal_Scheme == Crank_Nicolson:
        r = (2+w)/(2-w)
    elif Temporal_Scheme == RK4:
        r = 1 + w + (w**2)/2 + (w**3)/6 + (w**4)/24
    elif Temporal_Scheme == LeapFrog:
        r = sqrt(1)

    return r
```

4.3 Principal: Milestone 4

En esta ocasión, siguiendo la corrección de los anteriores hitos, se ha implementado el uso de listas para los esquemas numéricos y disminuyendo así el número de líneas de código. En este apartado, se presenta el código principal del Hito 4. Se definen las condiciones iniciales, se establece la lista de esquemas numéricos y pasos temporales, y posteriormente, se integra el oscilador lineal y se pinta la región de estabilidad.

```
from Functions.Physics import Kepler, Linear_Oscilator
from Functions.Temporal_Schemes import Euler, Crank_Nicolson, RK4,
                                       Inverse_Euler, LeapFrog
from Functions.Error import Richardson, Covergence_Rate
from Functions.Cauchy_Problem import Cauchy_problem
from Functions.Stability import Stability_Region
import matplotlib.pyplot as plt
from numpy import array, linspace, zeros, size, hstack
```

```
tf = 20 # valor final
dt = [0.1, 0.01, 0.001]

# CONDICIONES INICIALES
U0 = array([1,0])
print(U0)


Temp_Schemes = [Euler, RK4, Crank_Nicolson, Inverse_Euler, LeapFrog]
T_S_list = ['Euler', 'RK4', 'CrankNicolson', 'InverseEuler', 'LeapFrog']
colors = ['b--','r--','g--']
print(size(Temp_Schemes))

# INTEGRAR OSCILADOR LINEAR
for i in range(size(Temp_Schemes)):


    for j in range (size(dt)):
        N = int(tf/dt[j])
        t = linspace(0, tf, N)
        U = Cauchy_problem( Temp_Schemes[i], Linear_Oscilator, t, U0)

        plt.figure(i)
        plt.plot(t, U[:,0],colors[j], label = 'dt =' + str(dt[j]) + ' s'
                                                )
        plt.title(f'Oscilador integrado con {T_S_list[i]}')
        plt.xlabel("t (s)")
        plt.ylabel("x")
        plt.legend(loc = "lower left")
        plt.grid()
    #plt.show()


# REGION DE ESTABILIDAD
a = linspace(-5, 5, num=100);
b = linspace(-5, 5, num=100);


for i in range(size(Temp_Schemes)):
    Stab_Region = Stability_Region(Temp_Schemes[i],a,b)
    print(Stab_Region)
    plt.figure(i+size(Temp_Schemes))

    if Temp_Schemes[i] == 'LeapFrog':
        Im = linspace(-1,1,100)
        Re = zeros(100)
        print('im',Im)
        print('re',Re)
        plt.plot(Re, Im, color = '#b300ff')
        plt.show()

    else:
        plt.contour(a,b, Stab_Region, levels = [0, 1], colors = ['#
                                                b300ff'])
```

```
        plt.contourf(a,b, Stab_Region, levels = [0, 1], colors =['#
                                                78745d'])

    colors = ['b','r','g']

    for j in range(len(dt)):
        plt.plot([0,0], [dt[j],-dt[j]], 'o', color = colors[j], label =
                                                'dt =' + str(dt[j]) + ' s')
    plt.ylabel("Im")
    plt.xlabel("Re")
    plt.title(f'Region de estabilidad de {T_S_list[i]}')
    plt.legend()
    plt.grid()
    plt.show()
```