


Universidad Politécnica de Madrid

Ampliación de Matemáticas I

Informe Hito 4

Autor:
Guillermo García del Río

Índice

Índice de figuras

1. Introducción

En este informe se presentan y comentan los resultados obtenidos en el hito 4 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:

1 Integrar el oscilador lineal $\ddot{x} + x = 0$ con algunas condiciones iniciales. Se deben usar los esquemas numéricos de Euler, Euler inverso, Leap-Frog, Crank-Nicolson y Runge Kutta de orden 4.

2 Calcular y representar las regiones de estabilidad absoluta de los métodos anteriormente mencionados.

3 Explicar los resutados numéricos basándose en las regiones de estabilidad absoluta.

3. Problema a resolver

3.1. Oscilador lineal

El objetivo de este hito es integrar un oscilador armónico lineal para unas condiciones iniciales dadas. La expresión del oscilador armónico es la siguiente:

$$\ddot{x} + x = 0 \tag{3.1.1}$$

En este caso el vector de estado U quedaría definido por la posición y la velocidad:

$$U = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \tag{3.1.2}$$

Por lo tanto, este problema se puede formular como un sistema dos ecuaciones de primer orden:

$$F(U,t) = \frac{dU}{dt} = \frac{d}{dt}\begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} \dot{x} \\ -x \end{pmatrix} \tag{3.1.3}$$

Por último, se van a aplicar las siguientes condiciones iniciales al problema:

$$U_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{3.1.4}$$

Con todo esto ya se está en disposición de aplicar los esquemas numéricos anteriormente desarrollados para integrar el problema.

3.2. Esquema numérico Leap-Frog

Otro de los objetivos de este hito es implementar el esquema numérico de Leap-Frog o regla del punto medio. Este esquema numérico se emlpea para obtener la solución U_2 en un tiempo t_{n+1} desde una solución dada U_1 en t_n.

$$U^{n+1} = U^{n-1} + 2\Delta t F^n \tag{3.2.1}$$

Este esquema se usa para determinar la solución en los puntos internos de cualquier nivel. Además, ya que es un esquema de dos pasos, se necesitará una condición inicial adicional (cómo arranca el sistema). Esta condición inicial suele venir dada por el método de Euler, es decir, el primer paso se integra con Euler y a partir de ahí se utiliza el Leap-Frog.

3.3. Regiones de estabilidad absoluta

Se define la región de estabilidad absoluta como la región del plano complejo donde las soluciones numéricas son asintóticamente estables. El método para encontrar dicha región es determinar el lugar geométrico del contorno de la región, lugar donde alguna de las raíces del polinomio característico de estabilidad $\Pi(r, \omega)$ (cada esquema tiene uno) tienen módulo igual a la unidad. Si se sustituye $r = e^{i\theta}$ se puede obtener $\omega = \omega(\theta)$, es decir, el lugar geométrico del plano complejo donde alguna de las raíces tiene módulo unidad.

De esta forma, el plano complejo queda dividido en dos regiones: una donde el módulo de alguna de las raíces es mayor que la unidad y otra donde el módulo de todas las raíces es menor que la unidad.

Teorema de estabilidad: si los autovalores λ multiplicados por Δt están dentro de la región de estabilidad, entonces el error está acotado.

Donde λ son los autovalores de la matriz B:

$$U^{n+1} = BU^n \tag{3.3.1}$$

Por lo tanto, la estabilidad de un sistema normalmente se puede alcanzar reduciendo el paso de integración Δt. Al aumentar el Δt, el módulo de alguna de las raíces del polinomio característico puede hacerse mayor que uno y en ese momento la solución numérica se vuelve inestable.

Los polinomios característicos de los esquemas numéricos empleados en este hito son los siguientes.

Polinomio característico de Euler:

$$\Pi(r, \omega) = r - 1 - \omega \tag{3.3.2}$$

Polinomio característico de Euler inverso:

$$\Pi(r,\omega) = r - \frac{1}{1-\omega} \tag{3.3.3}$$

Polinomio característico de Crank Nicolson:

$$\Pi(r,\omega) = r - \frac{1+\frac{\omega}{2}}{1-\frac{\omega}{2}} \tag{3.3.4}$$

Polinomio característico de Runge Kutta de cuarto orden:

$$\Pi(r,\omega) = r - 1 - \omega - \frac{\omega^2}{2} - \frac{\omega^3}{6} - \frac{\omega^4}{24} \tag{3.3.5}$$

Polinomio característico del Leap-Frog:

$$\Pi(r,\omega) = r^2 - 1 \tag{3.3.6}$$

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. En este informe tan solo se incluirán los códigos desarrollados para el la integración del oscilador, el esquema numérico Leap Frog, el cálculo de las regiones de estabilidad y el archivo principal del Hito 4.

4.1. Módulo de osciladores

En este módulo se irán incluyendo los códigos para diferentes osciladores según se vayan desarrollando, de momento solo cuenta con el oscilador lineal planteado para este problema. La matematica implicada se ha expuesto previamente.

```
1
2  ## OSCILATORS ##
3
4   #   Inputs:
5   #           U : state vector at tn
6   #           t : tn
7   #
8   #   Return:
9   #
10  #           F(U,t) : Function derivated from the state vector dU/dt = F(U,t)
11
12 def OscilatorX(U,t):
13
14     x    = U[0]
15     dxdt = U[1]
16
17     F = array([dxdt,-x])
18
```

```
19      return F
```

Algoritmo 4.1: Oscilador armónico lineal

4.2. Esquema numérico Leap Frog

```
1
2  def leapfrog (U, dt, t, F):
3
4    # First step with Euler
5
6    if t == 0 :
7
8        U +=  dt * F(U,t)
9
10    # Leap-Frog
11
12   else:
13
14       # Assume that the first half of U correspond to position and the second to
       velocity
15
16       mid = int(size(U)/2)
17
18       Ua  = U
19
20       # Acceleration at n
21
22       new = F(Ua,t)
23
24       # New position and halfstep velocity
25
26       Ua[mid:] += new[mid:] * dt * 0.5
27       Ua[:mid] += new[:mid] * dt
28
29       # Acceleration at n + 1
30
31       new = F(Ua,t)
32
33       # New velocity
34
35       Ua[mid:] += new[mid:] * dt * 0.5
36
37       # Final state vector
38
39       U = Ua
40
41
42   return U
```

Algoritmo 4.2: Leap Frog

Como se puede comprobar en el código, primero se calcula la aceleración en el instante n. Con ella se calcula la nueva posición y la velocidad de la mitad de un paso, datos con los que se vuelve a calcular la aceleración para $n + 1$. Por último, con esta nueva aceleración se calcula la nueva velocidad.

4.3. Función para el cálculo de regiones de estabilidad

El código implementado para llevar a cabo el cálculo de las regio:

```
def Characteristic_Polynomia(Scheme):

    ## Create theta variable for the complex number "r"

    theta = linspace(0,8*pi, 200)

    ## Create the empty "R" and "I" vectors in order to store the polynomia
    solution in them
    #
    R = zeros(size(theta))
    I = zeros(size(theta))

    ## Create the initial condition for the equaiton to iterate.

    x0 = zeros(2)

    ## Loop to find the complex number w that satisfies the polynomia for every
    variation of |r| = 1

    for i in range(size(theta)):

        ## Forcing |r| = 1

        x = cos(theta[i])
        y = sin(theta[i])
        r = complex(x,y)

        ## Characteristic polynomia of the used schemes

        def Equation(w):
            if Scheme == Euler:
                poly = r - 1 - w
            elif Scheme == Euler_inverso:
                poly = r - 1/(1-w)
            elif Scheme == Crank_Nicolson:
                poly = r - (1 + w/2)/(1 - w/2)
            elif Scheme == RK4:
                poly = r - 1 - w - (w**2)/2 - (w**3)/6 - (w**4)/(4*3*2)
            elif Scheme == leapfrog:
                poly = r**2 - 1

            return poly

        ## Finding the solution

        w = findroot(Equation,x0[0]+x0[1]*1j)

        ## Converting the solution from "mpf" to float

        S = array([float(str(w.real)),float(str(w.imag))])

        ## Renewing the initial condition with the solution obtained

        x0[0] = S[0]
        x0[1] = S[1]

        ## Crank-Nicolson adjustment in order to avoid "findroot" tolerance issues

        if Scheme == Crank_Nicolson:
           x0 = zeros(2)

        ## Real and imaginary part storage

        R[i] = S[0]
        I[i] = S[1]

```

```
    return [R,I]
```

Algoritmo 4.3: Regiones de estabilidad

Como se puede observar en el código mostrado el objetivo es calcular las ω para las cuales se cumple que $\|r\| = 1$ para distintos valores de θ. Tanto la parte real como la parte imaginaria de las distintas ω se van almacenando en unos vectores para su posterior representación gráfica.

4.4. Archivo principal para el Hito 4

De manera similar a hitos anteriores, de cara a automatizar el proceso se ha realizado un bucle en el que se iban recorriendo los distintos esquemas numéricos de los cuales se ha solicitado un análisis. Por otro lado también se ha hecho un bucle para utilizar distintos pasos de tiempo.

```
## HITO 4 ##

import matplotlib.pyplot as plt
from numpy import array, linspace, zeros, size

from Numeric.Esquemas_numéricos import (RK4, Crank_Nicolson, Euler,
                                        Euler_inverso, leapfrog)
from Physics.Oscilators import OscilatorX
from Mathematics.EDOS import Cauchy_Problem
from Numeric.Error import Characteristic_Polynomia

## Temporal variables ##

T = 20                          # Integration duration [s]
dt = 0.001                      # Integration step [s]
n = int(T/dt)                   # Number of steps
t  = linspace(0,T,n)            # Time array

## Initial conditions for the oscilator ##

U0 = array([0,1])

methods = [Euler,RK4, Crank_Nicolson, Euler_inverso, leapfrog]
lista = ['Euler','RK4','Crank Nicolson','Euler inverso', 'LeapFrog']
figuras = ['euler','rk4','cranknicolson','Eulerinverso','leapfrog']
markers = ['r--','b--','g--']

for j in range (size(methods)):

    T = 20                                              # Integration duration [s]
    dt = array([0.1, 0.01, 0.001])                      # Integration step [s]


    for i in range(size(dt)):

        n = int(T/dt[i])                    # Number of steps
        t  = linspace(0,T,n)                # Time array

        ### FIRST PART: OSCILATOR INTEGRATION ###

        U = Cauchy_Problem(OscilatorX,t,U0,methods[j])
        print(U[len(t)-1,:])

        plt.title(f'Oscilador integrado con {lista[j]}')
        plt.xlabel("Tiempo (s)")
        plt.ylabel("X",rotation = 0)
        plt.grid()
        plt.plot(t,U[:,0], markers[i], label = 'dt =' + str(dt[i]) + ' s')
        plt.legend(loc ='lower left')
```

```
50
51      plt.savefig('Plots/Hito 4/ osc' + figuras[j] +'.png')
52      plt.close()
53
54      ### SECOND PART: SCHEME'S ABSOLUTE STABILITY REGION ###
55
56      [A,B] = Characteristic_Polynomia(methods[j])
57      X = linspace(-3,3,10)
58      Y = linspace(-3,3,10)
59      zer = zeros(10)
60      fig = plt.figure()
61      ax = fig.add_subplot()
62      plt.title(f'Región de estabilidad absoluta de {lista[j]} ')
63      plt.plot(X,zer,'k-')                          ## X axis
64      plt.plot(zer,Y,'k-')                          ## Y axis
65      plt.grid()
66      plt.plot(A,B,'r',linewidth=4)                 ## Stability region
67
68      if methods[j] != Crank_Nicolson:
69          ax.set_aspect('equal', adjustable = 'box')
70
71      plt.xlabel("Re")
72      plt.ylabel("Im",rotation = 0)
73      plt.savefig('Plots/Hito 4/ ASR' + lista[j] + '.png')
```

Algoritmo 4.4: Archivo principal Hito 4

5. Resultados

Con el objetivo de llevar a cabo un buen análisis de los resultados obtenidos, y de forma parecida al hito anterior, se realizarán varias simulaciones variando el valor del paso de integración Δt y para un tiempo de 20 segundos. Se mostrarán los resultados de la integración obtenidos junto con la región de estabilidad absoluta del esquema empleado.

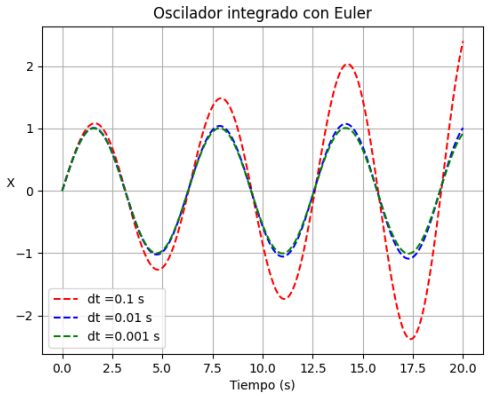
Los valores del paso de integración elegidos para este hito son:

Tabla 5.1: Valores del paso de tiempo empleados

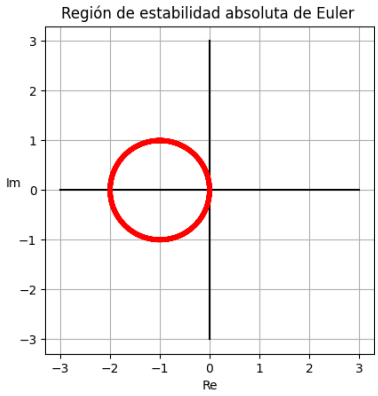
$\Delta t(s)$		
0.1	0.01	0.001

5.1. Euler

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



(a) Resultados obtenidos para distintos dt



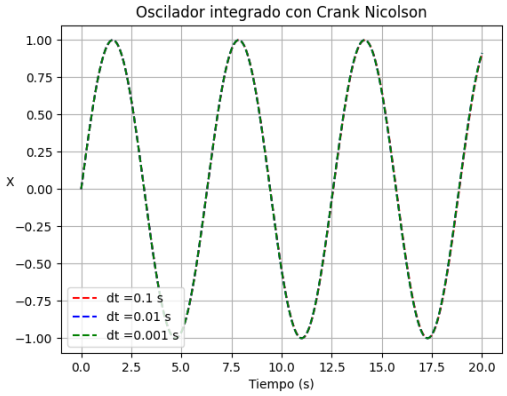
(b) Frontera de la REA del Euler

Figura 5.1: Resultados obtenidos para Euler

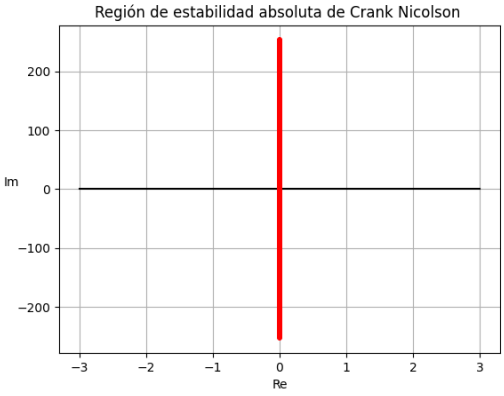
Como se puede observar en la figura 5.1, la frontera de la región de estabilidad absoluta del esquema es una circunferencia centrada en el $(-1, 0)$. Observando el comportamiento de los resultados obtenidos a medida que se va disminuyendo Δt (es decir, a medida que se va descendiendo en el eje de los imaginarios) se puede deducir que la región de estabilidad absoluta está dentro del círculo, ya que los resultados son más estables a medida que se aproximan a la frontera.

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 distintos dt



(b) Frontera de la REA del Crank Nicolson

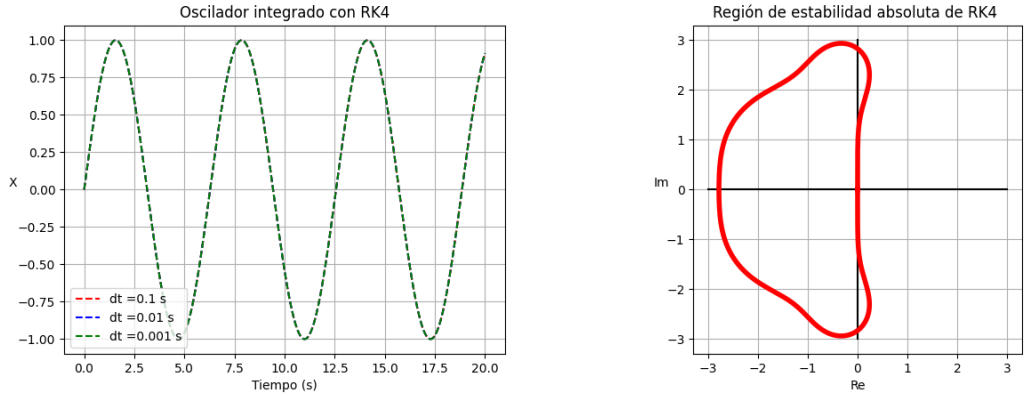
Figura 5.2: Resultados obtenidos para Crank Nicolson

Se puede observar en 5.2 que la variación del Δt no tiene impacto sobre los resultados obtenidos, esto es porque la frontera de la región de estabilidad absoluta del esquema de Crank Nicolson es el eje imaginario. Por lo tanto, cualquier valor de Δt estará en la frontera y por lo tanto la solución obtenida será estable. La región de estabilidad absoluta del esquema es el plano que queda

a la izquierda de la frontera.

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 distintos dt

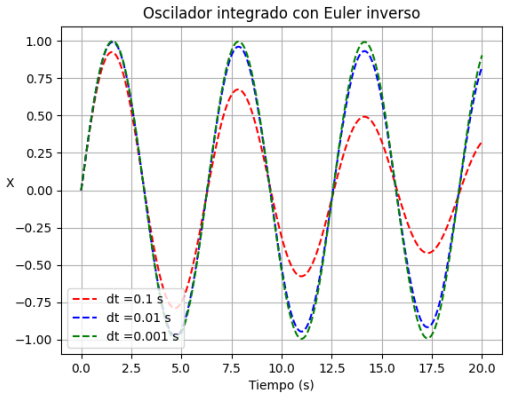
(b) Frontera de la REA del RK4

Figura 5.3: Resultados obtenidos para Runge Kutta de cuarto orden

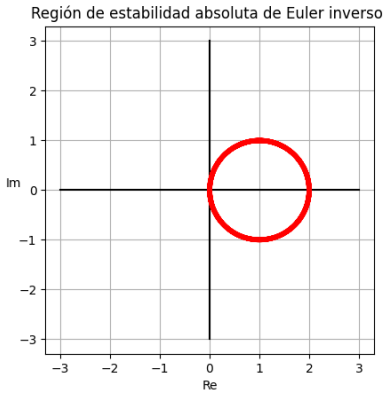
De forma parecida al esquema de Crank Nicolson, el esquema de Runge Kutta de cuarto orden no muestra cambio alguno al variar el Δt, esto es debido a que parte de su frontera comprende al eje imaginario. Sin embargo, a diferencia del anterior, existirá un Δt máximo que se saldrá de la región de estabilidad absoluta, la cual está comprendida dentro de la frontera.

5.4. Euler inverso

Los resultados obtenidos por el esquema numérico de Euler inverso se presentan en la figura 5.4.



(a) Resultados obtenidos para distintos dt



(b) Frontera de la REA del Euler inverso

Figura 5.4: Resultados obtenidos para Euler inverso

De forma parecida al método de Euler, se puede observar en la figura 5.4 que la frontera de la región de estabilidad absoluta del esquema es una circunferencia centrada en el $(1, 0)$. Análogamente, se puede deducir que la región de estabilidad absoluta está dentro del círculo, ya que los resultados son más estables a medida que se aproximan a la frontera.

5.5. Leap Frog

Los resultados obtenidos por el esquema numérico de Leap Frog se presentan en la figura 5.5.

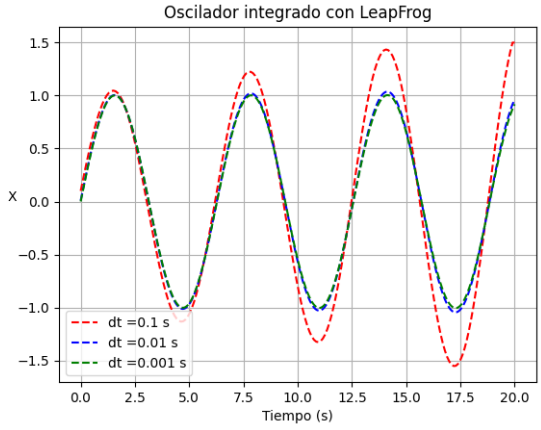


Figura 5.5: Resultados obtenidos para distintos dt

En cuanto a la región de estabilidad absoluta de este esquema, se recuerda cómo es el polinomio característico del mismo:

$$\Pi(r, \omega) = r^2 - 1 \tag{5.5.1}$$

Se puede observar que este polinomio tiene dos raíces distintas y con módulo igual a uno, por lo que se puede decir que el Leap Frog es cero-estable (estable por lo menos con $\Delta t \to 0$). La cero-estabilidad es una propiedad del esquema numérico y asegura que el sistema en diferencias es insensible a perturbaciones numéricas, es decir, que está bien planteado.