MEMORIA PRÁCTICA 3 - GCOM: Discretización de sistemas dinámicos continuos y Teorema de Liouville

Nombre: Gabriel Alba Serrano
Subgrupo: U1

1. Introducción

La discretización de un sistema dinámico continuo desde el punto de vista computacional consiste en discretizar el tiempo como $t := n\delta$, y representar la variable posición q como una sucesión $q_n := q(n\delta) = q(t)$ para cada tiempo discretizado t. Aproximando la derivada con el método de diferencias finitas, obtenemos la siguiente fórmula iterativa:

$$q_{n+2} := \delta^2 F(q_n) - q_n + 2q_{n+1} \tag{1}$$

En esta práctica se pide discretizar el sistema de un oscilador no lineal con el objetivo de comprobar si se cumple el Teorema de Liouville.

2. Material usado

2.1. Método

Utilizando el software de python3 y sus librerías matplotlib y scipy.spatial, he graficado el espacio fásico de un oscilador no lineal y su diagrama de fases para un tiempo concreto, además de calcular el área total de la envoltura convexa de dicho diagrama de fases. Finalmente he creado una animación gif con la evolución del diagrama de fases D_t para $t \in (0, 5)$

2.2. Datos

- La plantilla "GCOM2023-practica3-plantilla" que se ofrece en el campus virtual de la asignatura.
- La ecuación de Hamilton-Jacobi de un oscilador no lineal.

$$\ddot{q} = -\frac{8}{a} q(q-b)^2 \tag{2}$$

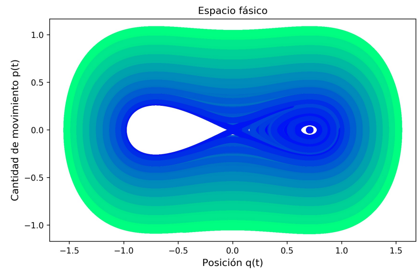
- Un conjunto de condiciones iniciales $D_0 := [0,1] \times [0,1]$, y una granularidad del parámetro temporal $t = n\delta$, con $\delta \in [10^{-4}, 10^{-3}]$ y $n \in N \cup \{0\}$.

3. Resultados

1. Para representar gráficamente el espacio fásico $D_{(0,\infty)}$ de las órbitas finales del sistema, he definido un conjunto de condiciones iniciales D_0, que representan, respectivamente, los puntos iniciales de las variables de posición y de movimiento lineal:

$$(q_{0_i}, dq_{0_j}) := \{(\frac{i}{20}, \frac{j}{20})\}_{i=0,j=0}^{20,20}$$

   Por cada punto, he dibujado una órbita final diferente.



2. Sea $A_t(\delta)$ el área del diagrama de fases del oscilador no lineal para $t \in \mathbf{R}$. He definido los siguientes conjuntos de condiciones iniciales:

$$X_{i,j} := \{(q_{0_i}, dq_{0_j})\}$$
$$Y_{i,j} := \{(q_{0_i}, dq_{0_j}) : y = 0\}$$
$$Z_{i,j} := \{(q_{0_i}, dq_{0_j}) : x = 1\}$$

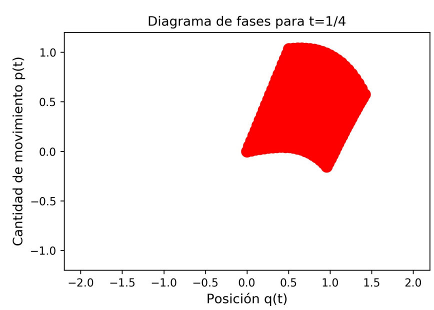
Sean A_X, A_Y, A_Z el área de las envolventes convexas de los diagramas de fases obtenidos al utilizar como condiciones iniciales los conjuntos X, Y, Z, respectivamente, tal que:

$$A_t(\delta) = A_X - A_Y - A_Z$$

Se pide calcular el área para $t = \frac{1}{4}$ y $\delta \in [10^{-4}, 10^{-3}]$, entonces el intervalo de error del cálculo de dicha área es $\xi := A_{\frac{1}{4}}(10^{-3}) - A_{\frac{1}{4}}(10^{-4})$. Tras varios cálculos he obtenido los siguientes resultados:

$$A_{\frac{1}{4}}(10^{-3}) = 1.00055 \ ; \ A_{\frac{1}{4}}(10^{-4}) = 0.99998 \ ; \ \xi = 5.626 \times 10^{-4}$$

Cabe destacar que en este sistema dinámico se cumple el Teorema de Liouville entre D_0 y $D_{\frac{1}{4}}$, es decir, el 2n-volumen de la distribcuión de fases es invariante en el tiempo, o equivalentemente, que el área del diagrama de fases es la misma para cualquier tiempo $t \in \mathbf{R}$. Como el conjunto de condiciones iniciales es $D_0 := [0,1] \times [0,1]$, entonces $A_0(\delta) = 1$, que no coincide exactamente con el valor de $A_{\frac{1}{4}}(\delta)$, para $\delta \in [10^{-4}, 10^{-3}]$, porque he calculado el área del sistema discretizado, en vez de continuo.



Finalmente se observa que entre D_0 y $D_{(0,\infty)}$ no se cumple el Teorema de Liouville, ya que $D_{(0,\infty)} = \cup_{t\in(0,\infty)} D_t$, entonces el área de $D_{(0,\infty)}$ sería mayor que $A_0(\delta) = 1$. Esto ocurre porque las variables $q(t)$ y $p(t)$ que representan, respectivamente, la posición y la cantidad de movimiento del sistema no son invariantes respecto al tiempo, es decir, cuando t varía el diagrama de fases es diferente, este resultado se puede observar con detalla en la animación GIF realizada en el último apartado.

3. Adjunto la animación GIF del diagrama de fases para $t \in (0,5)$ como 'animacionPractica3.gif'

4. Conclusión

La discretización del sistema dinámico de un oscilador no lineal, nos permite calcular el área del diagrama de fases de dicho sistema. El algoritmo utilizado para calcular la envolvente convexa del diagrama de fases $(p(t), q(t))$ para $t = \frac{1}{4}$ devuelve una aproximación, con un error $\xi = 2.873 \times 10^{-4}$, respecto al intervalo donde está definido el parámetro de granularidad temporal $\delta \in [10^{-4}, 10^{-3}]$.
El Teorema de Liouville afirma que el área del diagrama de fases es invariante respecto al tiempo, es decir, en este caso tendríamos que $A_t = 1$, para todo $t \in \mathbf{R}$. Comparando este resultado teórico con la aproximación numérica obtenida anteriormente, queda comprobado que se cumple el Teorema de Liouville entre D_0 y D_t. En cambio, no se cumple entre D_0 y $D_{(0,\infty)}$ esto ocurriría si el propio diagrama de fases fuese invariante respecto al tiempo.

5. Anexo con el script/código utilizado

```

"""
GCOM - PRACTICA 3: Discretizaci n de sistemas continuos y Teorema de Liouville
NOMBRE: Gabriel Alba Serrano
SUBGRUPO: U1
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull, convex_hull_plot_2d
from matplotlib import animation

# En primer lugar voy a definir las funciones que voy a utilizar para resolver
# cada apartado.

#q = variable de posici n
#dq0 = valor inicial de la derivada de q
#d = granularidad del par  metro temporal
def deriv(q,dq0,d):
   dq = (q[1:len(q)]-q[0:(len(q)-1)])/d
   dq = np.insert(dq,0,dq0)
   return dq

#Ecuacion de Hamilton-Jacobi del oscilador no lineal
def F(q):
    a = 3
    b = 1/2
    ddq = - (8/a)*q*(q**2 -b)
    return ddq

#Resolucion de la ecuacion dinamica ddq = F(q), obteniendo la orbita q(t)
#Los valores iniciales son la posici  n q0 := q(0) y la derivada dq0 := dq(0)
def orb(n, q0, dq0, F, d):
    q = np.empty([n+1])
    q[0] = q0
    q[1] = q0 + dq0*d
    for i in np.arange(2,n+1):
        q[i] = - q[i-2] + d**2*F(q[i-2]) + 2*q[i-1]
    return q

#Funcion que va a graficar cada diagrama de fases (q,p) respecto a unas
#determinadas condiones iniciales
def simplectica(q0, dq0, F, col=0, d = 10**(-4), marker='-'):
    n = int(16/d)
    q = orb(n,q0=q0,dq0=dq0,F=F,d=d)
    dq = deriv(q,dq0=dq0,d=d)
    p = dq/2
    plt.plot(q, p, marker, c=plt.get_cmap("winter")(col))

######################
##### APARTADO 1 #####
######################

```

```
horiz = 12
# Tomo delta como el punto medio del intervalo donde est  definido
d = (10**(-3)-10**(-4))/2

fig = plt.figure(figsize=(8,5))
fig.subplots_adjust(hspace=0.4, wspace=0.2)
ax = fig.add_subplot(1,1,1)

#Condiciones iniciales D0 = [0,1] x [0,1] :
#Defino un conjunto de condiciones iniciales que pertenece a D0
#Como p = dq/2, entonces dq0 := p0*2
#Las sucesiones seq_q0 y seq_dq0 tienen un total de 11 puntos, por lo tanto,
#voy a graficar 11*11=121 orbitas finales diferentes

seq_q0 = np.linspace(0.,1.,num=11)
seq_dq0 = np.linspace(0.,1.,num=11)*2
for i in range(len(seq_q0)):
    for j in range(len(seq_dq0)):
        q0 = seq_q0[i]
        dq0 = seq_dq0[j]
        col = (1+i+j*(len(seq_q0)))/(len(seq_q0)*len(seq_dq0))
        #ax = fig.add_subplot(len(seq_q0), len(seq_dq0), 1+i+j*(len(seq_q0)))
        simplectica(q0=q0, dq0=dq0, F=F, col=col, marker='ro', d=d)
ax.set_xlabel("Posici n q(t)", fontsize=12)
ax.set_ylabel("Cantidad de movimiento p(t)", fontsize=12)
plt.title('Espacio f sico')
#fig.savefig('Espacio fasico.jpg', dpi=250)
plt.show()

######################
##### APARTADO 2 #####
######################

#Para t=1/4 , es decir, para horiz=1/4, calculamos el area de la envoltura
#convexa del conjunto de puntos p y q. Dichos puntos son una sucesion de puntos
#calculados a partir de unas condiciones iniciales y la siguiente formula:
#    q[i] = - q[i-2] + d**2*F(q[i-2]) + 2*q[i-1]

#Calculamos dicha  rea  para delta = 10**(-3) y para delta = 10**(-4), con el
#objetivo de estimar el intervalo de error.
t = 0.25

def area_t(delta):
    n = int(t/delta)
    # Primero calculamos el diagrama de fases para las condiciones iniciales
    # definidas por seq_q0 y seq_dq0
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    seq_q0 = np.linspace(0.,1.,num=21)
    seq_dq0 = np.linspace(0.,1.,num=21)*2
    qX = np.array([])
    pX = np.array([])
    for i in range(len(seq_q0)):
        for j in range(len(seq_dq0)):
            q0 = seq_q0[i]
            dq0 = seq_dq0[j]
            q = orb(n,q0=q0,dq0=dq0,F=F,d=delta)
            dq = deriv(q,dq0=dq0,d=delta)
            p = dq/2
            qX = np.append(qX,q[-1])
            pX = np.append(pX,p[-1])
            plt.xlim(-2.2, 2.2)
            plt.ylim(-1.2, 1.2)
            plt.rcParams["legend.markerscale"] = 6
            plt.plot(q[-1], p[-1], marker="o", markersize= 10,
                     markeredgecolor="red",markerfacecolor="red")
    ax.set_xlabel("Posici n q(t)", fontsize=12)
    ax.set_ylabel("Cantidad de movimiento p(t)", fontsize=12)
    plt.title('Diagrama de fases para t=1/4')
    plt.show()
    #fig.savefig('Diagrama de fases.jpg', dpi=250)
```

```python

    #Calculamos el area de la envolvente convexa del diagrama de fases que
    # hemos obtenido
    X = ConvexHull(np.array([qX,pX]).T)
    aX = X.volume

    #Definimos dos nuevos conjuntos de condiciones iniciales:
    # Y0 = { (seq_q0, seq_dq0) | y = 0 }
    # Z0 = { (seq_q0, seq_dq0) | x = 1}

    #Calculamos las envolventes convexas de los conjuntos de puntos de las
    #condiciones iniciales Y0 & Z0
    qY = np.array([])
    pY = np.array([])
    for i in range(len(seq_q0)):
        q0 = seq_q0[i]
        dq0 = 0
        q = orb(n,q0=q0,dq0=dq0,F=F,d=delta)
        dq = deriv(q,dq0=dq0,d=delta)
        p = dq/2
        qY = np.append(qY,q[-1])
        pY = np.append(pY,p[-1])
    Y = ConvexHull(np.array([qY,pY]).T)
    aY = Y.volume

    qZ = np.array([])
    pZ = np.array([])
    for i in range(len(seq_dq0)):
        q0 = 1
        dq0 = seq_dq0[i]
        q = orb(n,q0=q0,dq0=dq0,F=F,d=delta)
        dq = deriv(q,dq0=dq0,d=delta)
        p = dq/2
        qZ = np.append(qZ,q[-1])
        pZ = np.append(pZ,p[-1])
    Z = ConvexHull(np.array([qZ,pZ]).T)
    aZ = Z.volume

    #Observando la gr fica, nos damos cuenta que el area de la envolvente
    #convexa X NO es el area real que queremos calcular, por ello tenemos que
    #restarle el  rea  de las envolventes convexas Y & Z
    #Devolvemos el valor del area real del diagrama de fases para t = 1/4
    return aX - aY - aZ

#Definimos a1 y a2 como el area del diagrama de fases para t = 1/4
# cuando delta = 10**(-3) y 10**(-4) respectivamente
a1 = area_t(10**(-3))
a2 = area_t(10**(-4))

print('El  rea  del espacio f sico para t = 1/4 y delta = 10**(-3) es:', a1)
print('El  rea  del espacio f sico para t = 1/4 y delta = 10**(-4) es:', a2)
print("Intervalo de error del valor del  rea :", format(a1 - a2,'.3E'))

#####################
##### APARTADO 3 #####
#####################

fig, ax = plt.subplots()

def animate(t):
    seq_q0 = np.linspace(0.,1.,num=16)
    seq_dq0 = np.linspace(0.,1.,num=16)*2
    d = 10**(-4)
    n = int(t/d)+1 #Sumo 1 para evitar que n sea 0 cuando t=0
    ax.clear()
    ax.set_xlim(-2,2)
    ax.set_ylim(-2,2)

    for i in range(len(seq_q0)):
        for j in range(len(seq_dq0)):
            q0 = seq_q0[i]
```

```python
            dq0 = seq_dq0[j]
            q = orb(n,q0=q0,dq0=dq0,F=F,d=d)
            dq = deriv(q,dq0=dq0,d=d)
            p = dq/2
            plt.rcParams["legend.markerscale"] = 6
            ax.set_xlabel("Posicion q(t)", fontsize=12)
            ax.set_ylabel("Cantidad de movimiento p(t)", fontsize=12)
            ax.plot(q[-1], p[-1], marker="o", markersize= 10,
                    markeredgecolor="purple",markerfacecolor="purple")
    return ax,

def init():
    return animate(0),

ani = animation.FuncAnimation(fig, animate, np.linspace(0.,5.,num=16),
                              init_func=init, interval=100)
#ani.save("animacionPractica3.gif", fps = 5)
```