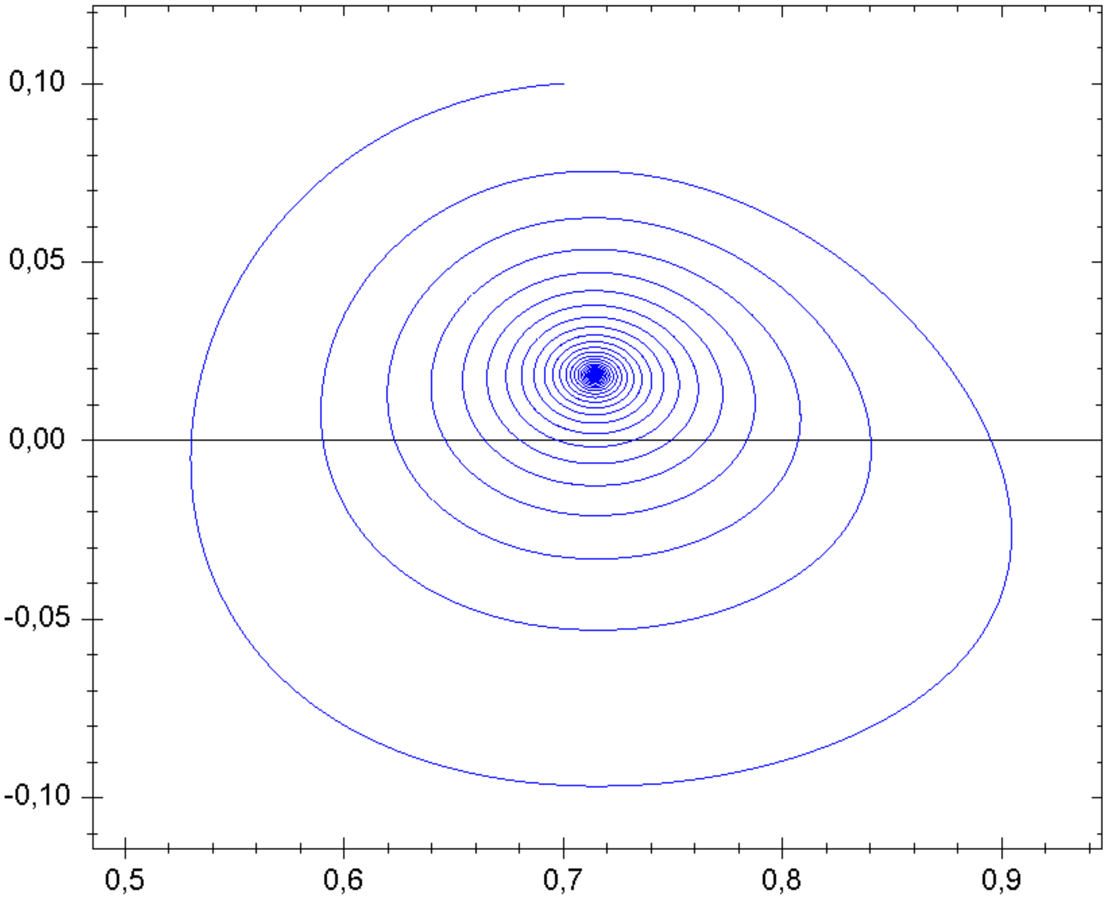
GEOMETRÍA COMPUTACIONAL

PRÁCTICA 6



DAVID SEIJAS PÉREZ

1. Introducción

En esta práctica queremos representar gráficamente el espacio fásico de las órbitas de un sistema S dado por su Hamiltoniano:

$$H(p,q) = p^2 + \frac{1}{4} \cdot (q^2 - 1)^2 \quad \Rightarrow \quad \ddot{q} = -2q \cdot (q^2 - 1)$$

Además, nos familiarizaremos con diferentes conceptos como el *Teorema de Liouville* y las formas simplécticas, necesarios para el entendimiento y elaboración de esta práctica.

2. Material Usado

Para el desarrollo de esta práctica he utilizado diversas librerías dadas por Python. Primero, como siempre, hemos usado *matplotlib.pyplot* para la representación de gráficos y *np* para el manejo de arrays. Además de estas, hemos usado la librería *animation* de *matplotlib* para realizar la animación de la proyección y las librerías *ConvexHull*, *convex_hull_plot_2d* para mostrar y calcular el área de la envoltura convexa del espacio fásico de D_t con $t = 1/4$.

Además, he usado la plantilla *GCOM2022-practica6_plantilla* para reutilizar diversas funciones del código como las que calculan la derivada, la órbita o la simpléctica, además de reutilizar código para la realización de los 3 apartados y el cáculo del diagrama de fases específico.

3. Metodología

Antes de nada, y gracias a la plantilla, definimos las condiciones iniciales D_0 y el número de órbitas a considerar en cada uno de los apartados.

Para el primer apartado desarrollamos la función *apartado1()* donde consideramos $\delta = 10^{-4}$, $t = 32$ y 12 órbitas finales y, aprovechando el código aportado por la plantilla y las funciones dadas *orb* y *deriv* calculamos y mostramos el espacio fásico $D_{(0,\infty)}$.

Para el segundo apartado, en la función *apartado2()*, consideramos $t = 1/4$ y calculamos el diagrama de fases específico para ese t con dos valores distintos de δ: 10^{-3} y 10^{-4}. Para cada uno de los valores calculamos el área de la envoltura convexa de este espacio fásico. Sin embargo, para calcular el área real del espacio fásico hay que restarle el área de la envoltura convexa de la parte sobrante por abajo y por la derecha. Para calcular estas áreas nos tenemos que quedar con la última fila y última columna, respectivamente, de los vectores q y p. Descomentando las líneas de código de la función *apartado2* podríamos guardar y observar la imagen de la envoltura convexa de estas partes sobrantes de abajo y derecha y ver como, visualmente, parece coincidir el área con la calculada con *ConvexHull*.

En el último apartado apartado, lo que he hecho es implementar las funciones *animation* e *init*, modificadas de la plantilla para ajustarlas al espacio fásico D_t con $t = 0.1$. Con esto obtenemos el GIF a 5 fps del diagrama de fases para $\delta = 10^{-4}$.

4. Resultados

En el primer apartado, para $\delta = 10^{-4}$ y considerando 12 órbitas finales, he obtenido la representación, mostrada en la figura a), del espacio fásico $D_{(0,\infty)}$.

En el segundo apartado hemos obtenido la representación de la envoltura convexa del diagrama de fases para las condiciones especificadas, mostrada en la figura b). Además, hemos obtenido los siguientes valores de las áreas:

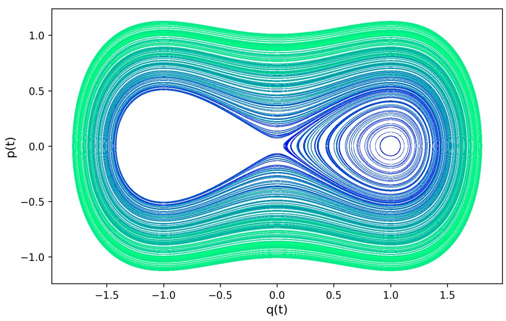
Para $\delta = 10^{-4}$:

Área total: 1.0664
Área parte derecha: 0.0043
Área parte inferior: 0.0622
Área de D_t:: 0.9999

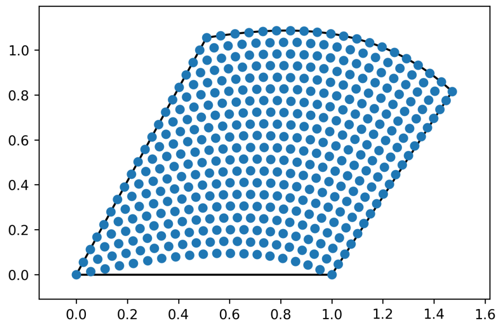
Para $\delta = 10^{-3}$:

Área total: 1.0663
Área parte derecha: 0.0041
Área parte inferior: 0.0621
Área de D_t: 1.0001

Por tanto, el valor del área de D_t para $t = 1/4$ es 1.000 ± 0.0001



(a) Espacio fásico $D_{(0,\infty)}$



(b) Envoltura Convexa

Para el tercer y último apartado adjunto el archivo *evolucion.gif* donde se ve la animación del diagrama de fases para $t = 0.1$.

5. Conclusión

El Teorema de Louiville sabemos de antemano que se verifica entre D_0 y D_t para $t = 1/4$ (en el apartado 2) al ser ambos en instantes concretos, pero no se cumple entre D_0 y $D_(0,\infty)$ al ser el segundo intervalo de tiempos pues solo se cumple para instantes de tiempo.

Gracias a esta práctica podemos verificar que esto es así de forma práctica con el segundo apartado. Como observamos el área para $t = 1/4$ es 1, sin tener en cuenta la estimación del error, que es exactamente el área de $D_0 = [0,1] \times [0,1]$. Sin embargo, se puede observar también en esta práctica (gracias al apartado 1) que el área del espacio fásico $D(0,\infty)$ es claramente superior de 1. Por tanto, gracias a esta práctica podemos comprobar y entender de forma práctica y mucho más visual este Teorema.

6. Anexo: Código

```
"""
DAVID SEIJAS PEREZ
PRACTICA 6
"""

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


#q = variable de posici n, dq0 = \dot{q}(0) = valor inicial de la derivada
#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


#Ecuaci n del sistema din mico continuo
def F(q):
    ddq = -2*q*(q**2-1)
    return ddq


#Resoluci n de la ecuaci n din mica \ddot{q} = F(q), obteniendo la rbita q(t)
#Los valores iniciales son la posici n q0 := q(0) y la derivada dq0 := \dot{q}(0)
def orb(n, q0, dq0, F, args=None, d=0.001):
    q = np.empty([n+1])
    q[0] = q0
    q[1] = q0 + dq0*d
    for i in np.arange(2, n+1):
        args = q[i-2]
        q[i] = - q[i-2] + d**2*F(args) + 2*q[i-1]
    return q


## Pintamos el espacio de fases
```

```python
def simplectica(q0, dq0, F, d, n, col=0, marker='-'):
    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))


def diagrama_fases(t, d):
    q2 = np.array([])
    p2 = np.array([])

    #Condiciones iniciales
    seq_q0 = np.linspace(0.,1.,num=20)
    seq_dq0 = np.linspace(0.,2.,num=20)
    n = int(t/d) #t = n*delta

    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=d)
            dq = deriv(q, dq0=dq0, d=d)
            p = dq/2
            q2 = np.append(q2, q[-1])
            p2 = np.append(p2, p[-1])

    return (q2,p2)


def animate(t):
    ax = plt.axes()
    (q,p) = diagrama_fases(t, d=10**(-4))
    plt.xlim(-2.5, 2.5)
    plt.ylim(-1.5, 1.5)
    #plt.plot(q, p, marker=".", markeredgecolor="black",
        markerfacecolor="blue")
    ax.scatter(q, p, c=q, cmap="winter", marker=".")
    return ax,


def init():
    return animate(0.1),


def apartado1():
    fig = plt.figure(figsize=(8, 5))
    fig.subplots_adjust(hspace=0.4, wspace=0.2)
    ax = fig.add_subplot(1, 1, 1)

    #Condiciones iniciales
    seq_q0 = np.linspace(0.,1.,num=12)
    seq_dq0 = np.linspace(0.,2.,num=12)
    d = 10**(-4)
    n = int(32/d)
```

```
    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))
            simplectica(q0=q0, dq0=dq0, F=F, col=col, marker=',', d=d,
                n=n)

    ax.set_xlabel("q(t)", fontsize=12)
    ax.set_ylabel("p(t)", fontsize=12)
    fig.savefig('Simplectic.png', dpi=250)
    plt.show()


def apartado2(t, d):
    fig = plt.figure(figsize=(8,5))
    fig.subplots_adjust(hspace=0.4, wspace=0.2)
    ax = fig.add_subplot(1,1,1)

    (q,p) = diagrama_fases(t, d=d)
    print("Calculando  rea  de envoltura convexa para delta = ", d)

    ax.set_xlabel("q(t)", fontsize=12)
    ax.set_ylabel("p(t)", fontsize=12)
    plt.plot(q, p, marker="o", markersize= 10, markeredgecolor="black"
        )
    plt.show()

    X = np.array([q,p]).T
    hull = ConvexHull(X)
    fig = convex_hull_plot_2d(hull)
    fig.savefig('Convexa.png', dpi=250)
    X_area = hull.volume
    print(" rea  total:", X_area)

    X_der = np.array([q[-20:], p[-20:]]).T
    hull_der = ConvexHull(X_der)
    #fig = convex_hull_plot_2d(hull_right)
    #fig.savefig('Convexa_derecha.png', dpi=250)
    X_der_area = hull_der.volume
    print(" rea  parte derecha:", X_der_area)

    X_inf = np.array([q[::20], p[::20]]).T
    hull_inf = ConvexHull(X_inf)
    #fig = convex_hull_plot_2d(hull_bottom)
    #fig.savefig('Convexa_inferior.png', dpi=250)
    X_inf_area = hull_inf.volume
    print(" rea  parte inferior:", X_inf_area)

    area = X_area - X_inf_area - X_der_area
    print(" rea  de D_t para t =", t, ":", area)
```

```
def apartado3():
    fig = plt.figure(figsize=(8,5))
    ani = animation.FuncAnimation(fig, animate, np.arange(0.2, 4.9,
        0.1), init_func=init)
    ani.save("evolucion.gif", fps = 5)

apartado1()
apartado2(0.25, 10**(-4))
apartado2(0.25, 10**(-3))
apartado3()
```