Memoria Practica 5 – DEFORMACION DE VARIEDADES DIFERENCIABLES

Jorge del Valle Vázquez

1 Introducción

En esta práctica estudiamos el hamiltoniano de un oscilador no lineal, *H(q,p)* que describe una variedad simpléctica a partir de la ecuación diferencial

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

$$\ddot{q} = -2q(q^2-1)$$

que se obtiene con las ecuaciones de Hamilton-Jacobi. Las condiciones iniciales pertenecen a $D_0 \coloneqq [0,1] \times [0,1]$. Se discretiza el sistema con una granularidad sobre el tiempo

$t = n\delta \; con \;\; \delta \, \epsilon \, [10^{-4}, 10^{-3}]$.

2 Material usado

Se hace uso de la plantilla en Python proporcionada por el docente, que aporta las indicaciones necesarias para trabajar con los osciladores simples. Las graficas se realizan por medio de la librería **matplotlib**, la envoltura convexa con **scipy.spatial** y también **animation** para las animaciones. Entre otras herramientas auxiliares estría **numpy** para tratamiento de vectores y cálculos asociados.

Entre los cambios sobre la plantilla, detallamos la ecuación de F(q) por la de $\ddot{q} = -2q(q^2-1)$. Como p vive en [0,1] y tenemos que $\dot{q} = 2p$, esta toma valores en [0,2].

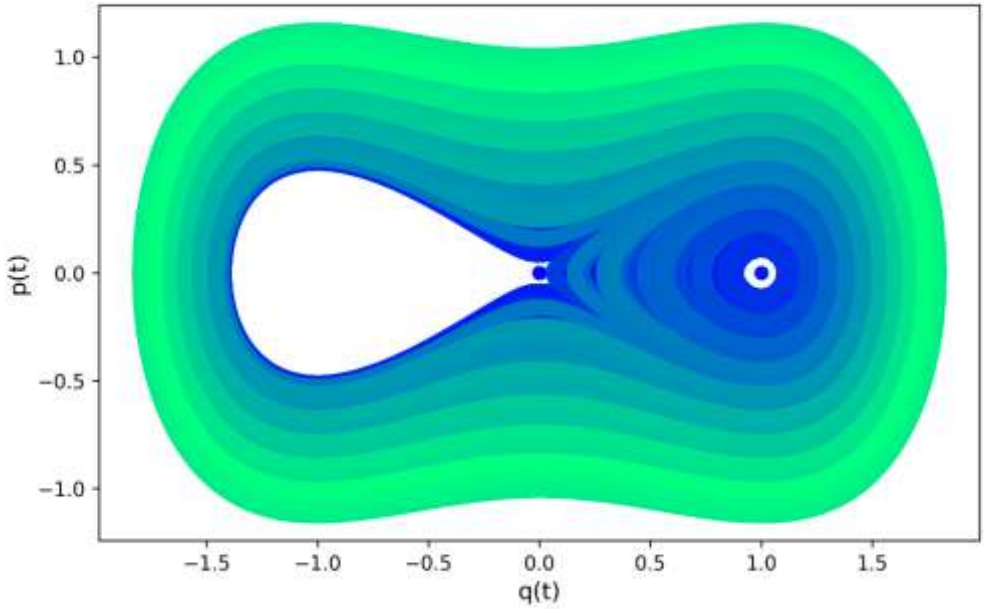
Para cada valor de q y $\dot{q}$ graficamos la órbita con la función de simpléctica proporcionada.

En segundo lugar, tenemos que estimar el error en el cálculo del área para un tiempo específico t(1/4). Agrupando parte del código de la plantilla definimos una función que devuelve el diagrama de fases (q, p) para ese tiempo determinado. Para el cálculo del área debemos tener en consideración las áreas que encierra la envoltura convexa y que no deseamos contabilizar, las partes inferiores y derecha.

Para finalizar se ilustra una animación del diagrama de fases sobre la variable temporal t.

3 Resultados

El espacio fásico obtenido en el primer apartado es el siguiente ($\delta = 10^{-4}$):

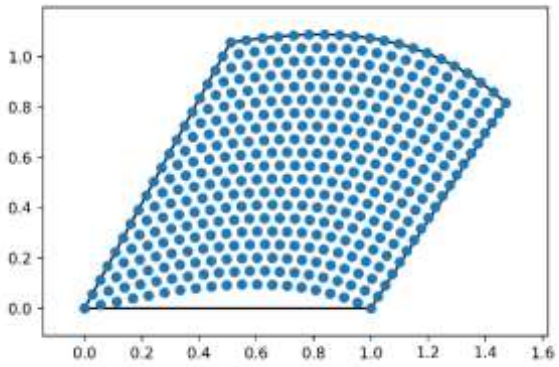


En el segundo ejercicio consideramos $\delta = 10^{-3}\ para$ obtener así un área 1.0001680221922904 que procede de lo expuesto antes sobre las envolturas convexas.

$$1.0663279223468758 - 0.0620808498746883 - 0.004079050279897095$$

$$Área = 1 \pm \delta$$

Por ello, se cumple el teorema de Liouville entre D_0 y D_t; pero a simple vista es posible descartar que se cumpla entre D_0 y $D_{(0,\infty)}$.



La animación se aporta como documentación extra en el archivo *evolucion.mp4*.

4 Conclusión

Este primer acercamiento al estudio de los espacios fásicos sirve para primero visualizar de forma clara como se construyen los diagramas de fases en función de sus condiciones iniciales, y segundo, ofrece un método ad-hoc para el cálculo de la áreas que pone de manifiesto las curvaturas del diagrama y como estás generan variaciones no deseadas en el cálculo de las áreas por medio de la envolvente.

5 Anexo: Código

```
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull, convex_hull_plot_2d
from matplotlib import animation
os.getcwd()
#q = variable de posiciÃ³n, dq0 = \dot{q}(0) = valor inicial de la derivada
#d = granularidad del parÃ¡metro temporal
def deriv(q,dq0,d):
   #dq = np.empty([len(q)])
   dq = (q[1:len(q)]-q[0:(len(q)-1)])/d
   dq = np.insert(dq,0,dq0)
   return dq
# Ecuación de un sistema dinámico continuo
# Ejemplo de oscilador simple
def F(q):
    ddq = - 2*q*(q*q - 1)
    return ddq

#ResoluciÃ³n de la ecuaciÃ³n dinÃ¡mica \ddot{q} = F(q), obteniendo la Ã³rbita
q(t)
#Los valores iniciales son la posiciÃ³n q0 := q(0) y la derivada dq0 :=
\dot{q}(0)
def orb(n,q0,dq0,F, d=10**(-3), args=None):
    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 #np.array(q),

## Pintamos el espacio de fases
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))

######################################################################
#  APARTADO 1
######################################################################
#D0 := [0, 1] × [0, 1]
d = 10**(-4)
t=32
n = int(t/d)
seq_q0 = np.linspace(0.,1.,num=10)
seq_dq0 = np.linspace(0.,2,num=10)

fig = plt.figure(figsize=(8,5))
fig.subplots_adjust(hspace=0.4, wspace=0.2)
ax = fig.add_subplot(1,1, 1)
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,d=d,n=n,col=col,marker='ro')
ax.set_xlabel("q(t)", fontsize=12)
ax.set_ylabel("p(t)", fontsize=12)
fig.savefig('Simplectica.png', dpi=250)
plt.show()

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



#Ejemplo de diagrama de fases (q, p) para un tiempo determinado
def diagrama_fases_tiempo_plot(t,d,s):
    seq_q0 = np.linspace(0.,1.,num=20)
    seq_dq0 = np.linspace(0.,2,num=20)
    q2 = np.array([])
    p2 = np.array([])
    n = int(t/d)
    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)
            dq = deriv(q,dq0=dq0,d=d)
            p = dq/2
            q2 = np.append(q2,q[-1])
            p2 = np.append(p2,p[-1])
```

```
            plt.plot(q[-1], p[-1], marker=".", markersize=10)
    plt.savefig(s, dpi=250)
    plt.show()
    return (q2,p2)

#Ejemplo de diagrama de fases (q, p) para un tiempo determinado
def diagrama_fases_tiempo(t,d):
    seq_q0 = np.linspace(0.,1.,num=20)
    seq_dq0 = np.linspace(0.,2,num=20)
    q2 = np.array([])
    p2 = np.array([])
    n = int(t/d)
    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)
            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 area(q, p,s):
    X = np.array([q, p]).T
    hull = ConvexHull(X)
    fig = convex_hull_plot_2d(hull)
    fig.savefig(s+'EnvCon_1.png', dpi=250)
    X_area = hull.volume
    print("Área total: ", X_area)

    Y = np.array([q[::20], p[::20]]).T
    hull_Y = ConvexHull(Y)
    fig2 = convex_hull_plot_2d(hull_Y)
    fig2.savefig(s+'EnvCon_2.png', dpi=250)
    X_area -= hull_Y.volume
    print("Área inferior: ", hull_Y.volume)

    Z = np.array([q[-20:], p[-20:]]).T
    hull_Z = ConvexHull(Z)
    fig3 = convex_hull_plot_2d(hull_Z)
    fig3.savefig(s+'EnvCon_3.png', dpi=250)
    X_area -= hull_Z.volume
    print("Área derecha: ", hull_Z.volume)
    return X_area

t=1/4
d = 10 **(-3)
(q,p) = diagrama_fases_tiempo_plot(t,d,s='diag_fases_t1.png')
```

```
print("Area: ", area(q,p,s='A'))
fig = plt.figure(figsize=(8,5))
fig.subplots_adjust(hspace=0.4, wspace=0.2)
ax = fig.add_subplot(1,1,1)
plt.plot(q, p, marker="o", markersize= 10)
fig.savefig('ap2_A.png', dpi=250)
plt.show()

d = 10 **(-4)
(q,p) = diagrama_fases_tiempo_plot(t,d,s='diag_fases_t2.png')
print("Area: ", area(q,p,s='B'))
fig = plt.figure(figsize=(8,5))
fig.subplots_adjust(hspace=0.4, wspace=0.2)
ax = fig.add_subplot(1,1,1)
plt.plot(q, p, marker="o", markersize= 10)
fig.savefig('ap2_B.png', dpi=250)
plt.show()



######################################################################
#  APARTADO 3
######################################################################
def animate(t):
    ax = plt.axes()
    (q,p) = diagrama_fases_tiempo(t,d=10 **(-3.5))
    plt.xlim(-2.5, 2.5)
    plt.ylim(-1.5, 1.5)
    # plt.plot(q, p, marker="o", markersize= 10,
markeredgecolor="red",markerfacecolor="red")
    ax.scatter(q, p, c=q, cmap="plasma", marker=".")
    return ax,

def init():
    return animate(0.1),

fig = plt.figure(figsize=(6,6))
ani = animation.FuncAnimation(fig, animate, np.arange(0.1, 5,0.1),
init_func=init)
ani.save("evolucion.mp4", fps = 10)
```