Íñigo Iriondo Delgado
Geometría Computacional

Memoria Práctica 3

(1) Introducción

Dado el hamiltoniano de un oscilador no lineal H(q,p) = p^2 + 1/3 (q^2 − 1/2)^2, tenemos que las ecuaciones de Hamilton–Jacobi del sistema son q'' = − 8/3 q (q^2 − 1/2), p(t) = q'/2. Para el conjunto de condiciones iniciales D_0 := [0, 1] × [0, 1] y la granularidad de t = n δ con δ ∈ [10−4 , 10−3], n natural, representaremos gráficamente el espacio físico D_(0,∞) de las órbitas finales del sistema, cálcularemos el área de D_t para distintos valores de t para ver si se cumple el teorema de Liouville y haremos una animación GIF de D_t para t ∈ (0, 5).

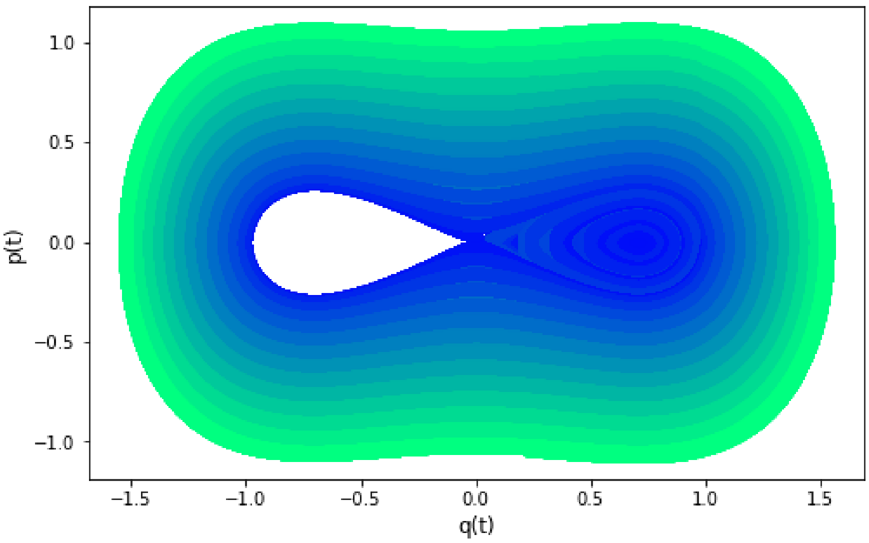
(2) Material usado

Programamos en Python usando la plantilla del campus virtual. Importamos *os*, *numpy*, *matplotlib.pyplot*, *ConvexHull*, *convex_hull_plot_2d* y *animation* y definimos las siguientes funciones: *deriv* (devuelve el vector derivada), *F* (ecuación diferencial del oscilador no lineal), *orb* (obtiene la órbita para nuestro sistema), *periodos* (calcula el periodo de un vector de datos), *simplectica* (dibuja una órbita del espacio físico), *area* (cálcula el área del espacio físico) y *teoremaDeLiouville* (imprime los áreas de los espacios físicos para distinta granularidad del parámetro temporal).

(3) Resultados

Si ejecutamos el programa obtenemos:

Apartado 1



Apartado 2

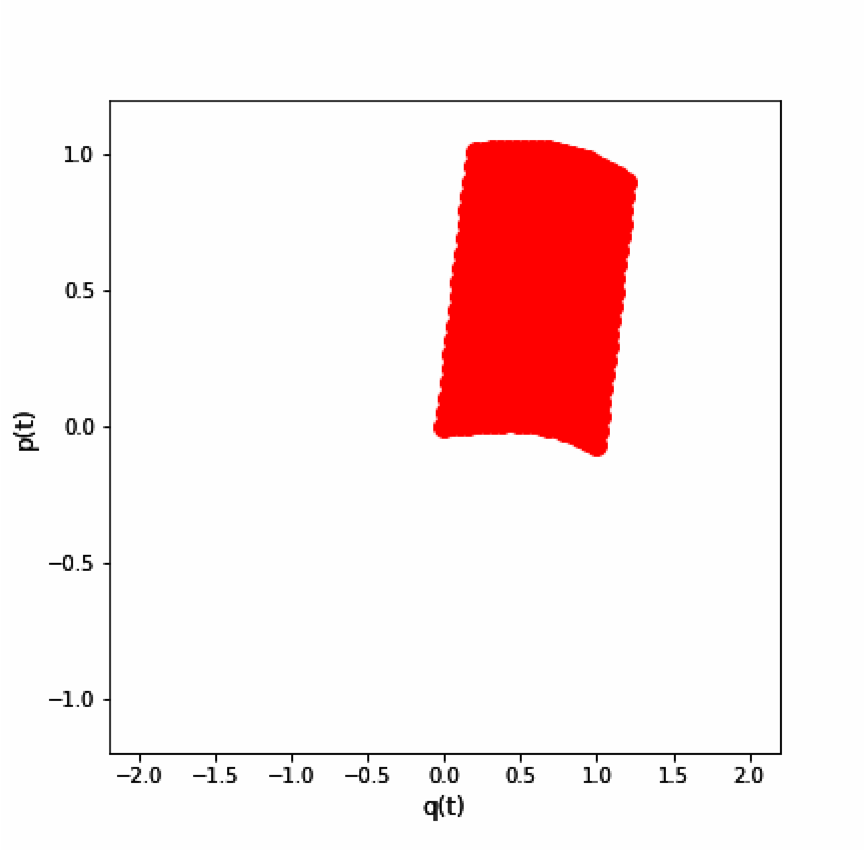
areas D_{1/4}

Area: 1.084
Area: 1.084

Area: 1.084
Area: 1.084
Area: 1.084
El area calculada es 1.084 con un error de 0.0
areas D_0

Area: 1.041
Area: 1.041
Area: 1.041
Area: 1.041
Area: 1.041
El area calculada es 1.041 con un error de 0.0
areas D_{(0, \infty}

Area: 1.183
Area: 1.182
Area: 1.182
Area: 1.182
Area: 1.182
El area calculada es 1.182 con un error de 0.001
Apartado 3



(4) Conclusión
Si aceptamos un error decimal, podemos afirmar que el teorema de Liouville se cumple para nuestros ejemplos de *D_t*, pero el error crece rápidamente cuando tomamos *t* más grande.

(5) Anexo con el script
Código compartido con David Diez Roshan

```
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull, convex_hull_plot_2d
#https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html
from matplotlib import animation

os.getcwd()

#q = variable de posición, dq0 = \dot{q}(0) = valor inicial de la derivada
#d = granularidad del parámetro 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) #dq = np.concatenate(([dq0],dq))
   return dq

#Ecuación de un sistema dinámico continuo
#Ejemplo de oscilador simple
def F(q):
    k = 1
    ddq = -(8/3)*q*(q**2 - 1/2)
    return ddq

#Resolución de la ecuación dinámica \ddot{q} = F(q), obteniendo la órbita q(t)
#Los valores iniciales son la posición q0 := q(0) y la derivada dq0 := \dot{q}(0)
def orb(n,q0,dq0,F, args=None, d=0.001):
    #q = [0.0]*(n+1)
    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),

def periodos(q,d,max=True):
    #Si max = True, tomamos las ondas a partir de los máximos/picos
    #Si max == False, tomamos las ondas a partir de los mínimos/valles
    epsilon = 5*d
    dq = deriv(q,dq0=None,d=d) #La primera derivada es irrelevante
    if max == True:
        waves = np.where((np.round(dq,int(-np.log10(epsilon))) == 0) & (q >0))
    if max != True:
```

```
        waves = np.where((np.round(dq,int(-np.log10(epsilon))) == 0) & (q <0))
    diff_waves = np.diff(waves)
    waves = waves[0][1:][diff_waves[0]>1]
    pers = diff_waves[diff_waves>1]*d
    return pers, waves

d = 10**(-4)

def simplectica(q0,dq0,F,col=0,d = 10**(-4),n = int(16/d),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))


print("Apartado 1\n")

Horiz = 12
d = 10**(-3)

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=14)
seq_dq0 = np.linspace(0.,2,num=14)
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= 10**(-3),n = int(Horiz/d))
ax.set_xlabel("q(t)", fontsize=12)
ax.set_ylabel("p(t)", fontsize=12)
#fig.savefig('Simplectic.png', dpi=250)
plt.show()


def area(d, horiz):
    seq_q0 = np.linspace(0., 1., num=20)
    seq_dq0 = np.linspace(0.,2, num=20)
    q2 = np.array([])
    p2 = np.array([])
    for i in range(len(seq_q0)):
        for j in range(len(seq_dq0)):
            q0 = seq_q0[i]
            dq0 = seq_dq0[j]
```

```
            n = int(horiz/d)
            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])
    X = np.array([q2, p2]).T
    hull = ConvexHull(X)
    print("Area:", round(hull.volume, 3))
    return hull.volume

def teoremaDeLiouville(t):
    deltas = np.linspace(3.01, 3.99, num=5)
    areas = [area(10**(-d), t) for d in deltas]
    resta_areas = [abs(areas[i] -areas[4]) for i in range(len(areas)-1)]
    sort_resta_areas = sorted(resta_areas)
    print("El area calculada es", round(areas[4],3),"con un error de", round(sort_resta_areas[3],3))

print("Apartado 2\n")

print('areas D_{1/4} \n')
teoremaDeLiouville(0.25)
print('areas D_0 \n')
teoremaDeLiouville(0.125)
print('areas D_{(0, \infty} \n')
teoremaDeLiouville(0.5)


print("Apartado 3\n")

fig= plt.figure(figsize=(8,5))
def animate(horiz):
    ax = fig.add_subplot(1,1,1)
    seq_q0 = np.linspace(0.,1.,num=20)
    seq_dq0 = np.linspace(0.,2,num=20)
    q2 = np.array([])
    p2 = np.array([])
    for i in range(len(seq_q0)):
        for j in range(len(seq_dq0)):
            q0 = seq_q0[i]
            dq0 = seq_dq0[j]
            d = 10**(-3)
            n = int(horiz/d)
            #t = np.arange(n+1)*d
            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])
        plt.xlim(-2.2, 2.2)
        plt.ylim(-1.2, 1.2)
        plt.rcParams["legend.markerscale"] = 6
        ax.set_xlabel("q(t)", fontsize=12)
        ax.set_ylabel("p(t)", fontsize=12)
        plt.plot(q[-1], p[-1], marker="o", markersize= 10,
        markeredgecolor="red",markerfacecolor="red")
    return ax

def init():
    return animate(0.1),

animate(np.arange(0.1,5.1,1)[1])
plt.show()

fig = plt.figure(figsize=(6,6))
ani = animation.FuncAnimation(fig, animate, np.arange(0.1,5.1,0.1), init_func=init,interval=20)

ani.save("apartado3.gif", fps=5)
```