Práctica 6 - Geometría Computacional

Marcos Herrero Agustín

1. Introducción

El objetivo de esta práctica es estudiar la variedad simpléctica dada por el hamiltoniano

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

Vamos a representar el espacio fásico del sistema, calcular el área de sus distribución de fases para un cierto tiempo t y obtener una animación que muestre la evolución del sistema en un intervalo de tiempo.

2. Datos y condiciones iniciales

Para la realización de la práctica se han utilizado los siguientes datos y condiciones iniciales:

- Las condiciones iniciales del sistema: $D_0 := [0,1] \times [0,1]$
- La granularidad del parámetro temporal $t = n\delta$ tal que $\delta \in [10^{-4}, 10^{-3}]$ y $n \in \mathbb{N} \cup \{0\}$.
- El valor del tiempo para el que calcular el área de la distribución de fases D_t: $t = \frac{1}{4}$.
- El intervalo de tiempo en el que realizar la animación: $t \in (0,5)$

3. Metodología

Empleando las ecuaciones de Hamilton-Jacobi se obtiene la ecuación diferencial de segundo orden:

$$q'' = -2q(q^2-1)$$

y la relación $dq = 2p$. La forma discretizada de la ecuación anterior es, discretizando el tiempo como $t = n\delta$,

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

donde $F(q) = -2q(q^2-1)$. Para calcular la componente q de una órbita del sistema se toma la condición inicial $(q_0, p_0) \in D_0$, se calcula $q_1 = q_0 + \delta dq_0$ y se itera la ecuación discretizada anterior tantas veces como se quiera.

Para el apartado i), formamos el espacio fásico con todas las órbitas que parten de D_0 hasta un tiempo $t = 10$. Nos guardamos, además, todos los puntos para reutilizarlos en apartados posteriores.

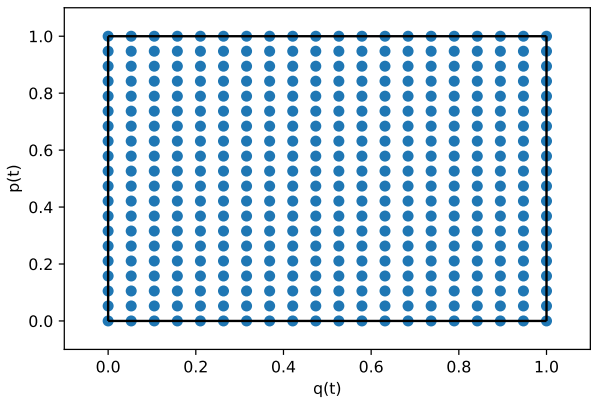

Figura 1

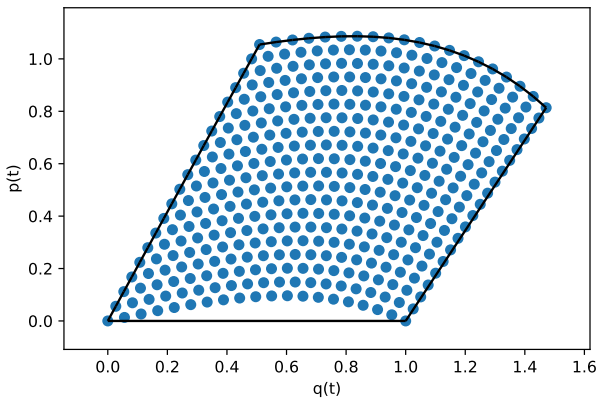

Figura 2

En el apartado ii), comenzamos calculando el área de D_0 simplemente como la de su envolvente convexa, ya que, como puede verse en la figura 1, D_0 coincide con su envolvente convexa. En cambio, calcular el área de $D_{1/4}$ no es tan directo, pues su envolvente convexa contiene puntos que no pertenecen a $D_{1/4}$, como se observa en la figura 2. Para calcular el área de $D_{1/4}$ hemos de restar al área de su envolvente el de las envolventes de las aristas inferior y derecha de D_0 transformadas por el sistema dinámico un tiempo $t = 1/4$. Estas envolventes se muestran en las figuras 3 y 4. Para estimar el error en este cálculo, repetimos la operación para valores de δ más finos hasta detectar la convergencia.

En el apartado iii), combinamos algunos de los puntos del espacio fásico que calculamos para el apartado i) en una animación. En concreto, para 25 tiempos t equiespaciados en el intervalo $(0,5)$ formamos un fotograma con los puntos de cada D_t.

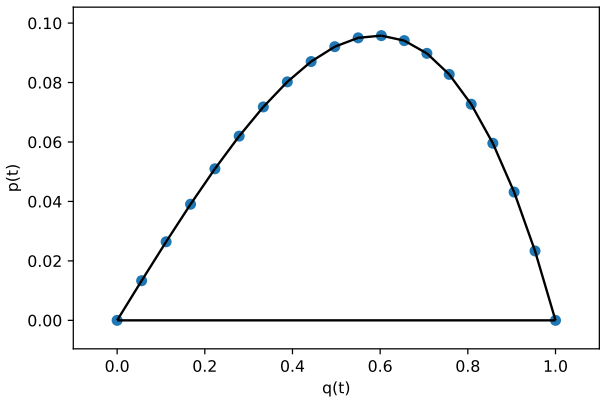

Figura 3

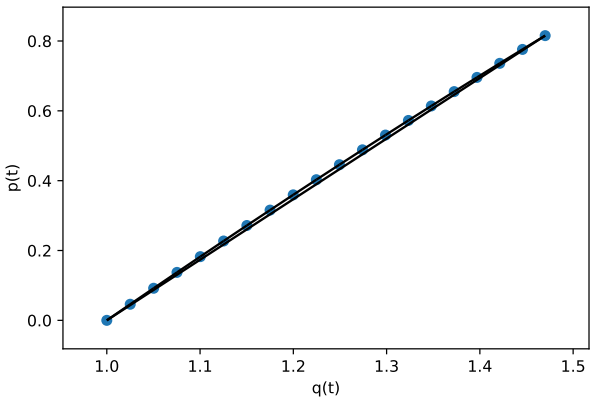

Figura 4

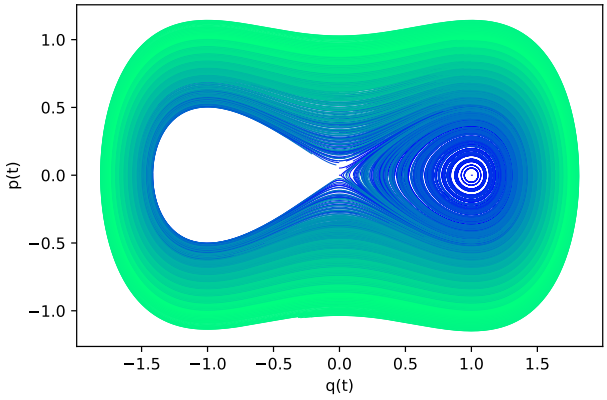

Figura 5

4. Resultados y discusión

4.1. Apartado *i)*

En la figura 5 se muestra el espacio fásico del sistema, formado por todos los puntos alcanzados por las curvas que parten de las condiciones iniciales D_0 y se prolongan durante un tiempo $t = 10$.

4.2. Apartado *ii)*

Como D_0 lo hemos construido explícitamente, sabemos de forma exacta que su área es 1. Esto puede comprobarse considerando el área de la envolvente convexa de D_0, que da 1 salvo error de coma flotante.

El área de $D_{1/4}$, por su parte, se estima, por el método que hemos descrito en la sección 3, como $0.99999 \pm 5 \cdot 10^{-5}$. Por tanto, D_0 y $D_{1/4}$ tienen la misma área, así que se verifica el teorema de Liouville entre D_0 y $D_{1/4}$.

En cambio, no tiene sentido considerar el teorema de Liouville entre D_0 y el espacio fásico completo $D_{(0,+\infty)}$. Puede verse claramente en la figura 5 que el área del espacio fásico será mucho mayor que la de D_0 o $D_{1/4}$ ya que, de hecho, los contiene a ambos. En este caso, no se cumple el teorema de Liouville.

4.3. Apartado *iii)*

Junto a esta memoria se adjunta un archivo "animación.gif" que constituye la animación pedida, que muestra la evolución del sistema en el intervalo de tiempo $(0, 5)$.

5. Conclusiones

En esta práctica hemos visto de manera práctica cómo se puede estudiar un sistema dinámico descrito mediante su hamiltoniano a través de su ecuación diferencial discretizada. Las técnicas que se han utilizado para esta práctica serían aplicables para el estudio de otros sistemas.

Asimismo, hemos comprobado empíricamente la validez del teorema de Liouville en este caso concreto.

Apéndice A Código utilizado

```
"""
Práctica 6
Geometría Computacional
Autor: Marcos Herrero
"""

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

'''
Funciones auxiliares

'''

#Funcion que define el sistema
def FSistema(q):
    return -2*q*(q*q-1)

#Devuelve los primeros n elementos de la órbita q(t) solución de dq = F(q)
# con condiciones iniciales q0 y dq0 y granularidad de tiempo delta
def orb(n,F,q0,dq0,delta):
    q = np.empty(n)
    q[0] = q0
    q[1] = q0 + delta*dq0

    for i in range(2,n):
        q[i] = delta**2*F(q[i-2])-q[i-2]+2*q[i-1]

    return q

def obtenDt(D0,t,delta):
    n = int(t/delta)
    Dt = []

    for q0,p0 in D0:
        dq0 = 2*p0

        #Calculamos la componente q de la órbita (calculamos un elemento más para poder
        q = orb(n+1,FSistema, q0, dq0,delta)

        #Calculamos su derivada discreta
        dq = (q[1:n+1]-q[0:n])/delta

        #A partir de la derivada se obtiene (por Hamilton-Jacobi) la componente p de la
        p = dq/2

        #Eliminamos el último elemento de q
        q = q[0:n]

        #Añadimos a Dt el punto obtenido
        Dt.append([q[-1],p[-1]])

    return Dt

#Obtener el área de la distribución de fases Dt para t= 1/4 y el delta dado
def obtenareaDt(q0s,p0s,delta):
    t = 1/4
    m1 = len(q0s)
```

```
    m2 = len(p0s)

    #Calculamos el área de la envolvente convexa de Dt
    D0 = [[q0s[i],p0s[j]] for i in range(m1) for j in range(m2)]
    Dt = obtenDt(D0,t,delta)
    DtHull = ConvexHull(Dt)
    areaDtHull = DtHull.volume

    #Calculamos el área resultado de transformar el segmento p = 0
    aristainf_0 = [[q0s[i],p0s[0]] for i in range(m1)]
    aristainf_t = obtenDt(aristainf_0,t,delta)
    hullaristainf_t = ConvexHull(aristainf_t)
    areaaristainfHull = hullaristainf_t.volume

    #Calculamos el área resultado de transformar la recta q = 1
    aristader_0 = [[q0s[m1-1],p0s[i]] for i in range(m2)]
    aristader_t = obtenDt(aristader_0,t,delta)
    hullaristader_t = ConvexHull(aristader_t)
    areaaristaderHull = hullaristader_t.volume

    #El área real de Dt es la de su cobertura convexa menos el excedente debido a
    #a la curvatura de las aristas p=0 y q=1
    return areaDtHull - areaaristainfHull - areaaristaderHull

#Realiza la animación pedida
def fAnimation(t,D,m1,m2):
    ax = plt.axes()
    ax.set_xlim(-2,2)
    ax.set_ylim(-2,2)

    for q,p in D[t]:
        ax.plot(q,p,marker ='o',markerfacecolor = 'tab:blue',markeredgecolor = 'tab:blu

'''
Apartado i): Representar gráficamente el espacio fásico de las órbitas finales del
del sistema (D(0,inf)). Nos guardamos además los puntos que calculamos para usarlos
en los siguientes apartados
'''
delta = 10**(-3)  #granularidad del tiempo
n = int(10/delta) #numero de puntos de cada órbita
m1 = 20
m2 = 20

ts = np.arange(n)*delta
D = [[] for i in range(n)]

#Condiciones iniciales (D0) de todo el espacio de fases
q0s = np.linspace(0,1,m1)
p0s = np.linspace(0,1,m2)

fig = plt.figure()
ax = fig.add_subplot(111)

#Representamos una curva por cada una de las condiciones iniciales anteriores
for i in range(m1):
    for j in range(m2):
        color = (1+i+j*(len(q0s)))/(len(q0s)*len(p0s))

        #Condiciones iniciales de la componente q de la orbita
        q0 = q0s[i]
        dq0 = 2*p0s[j]
```

```
        #Calculamos la componente q de la órbita (calculamos un elemento más para poder
        q = orb(n+1,FSistema, q0, dq0,delta)

        #Calculamos su derivada discreta
        dq = (q[1:n+1]-q[0:n])/delta

        #A partir de la derivada se obtiene (por Hamilton-Jacobi) la componente p de la
        p = dq/2

        #Eliminamos el último elemento de q
        q = q[0:n]


        #Representamos la curva
        ax.plot(q,p,c=plt.get_cmap('winter')(color),lw = 0.5)


        #Guardamos los puntos para reutilizarlos en los apartados posteriores
        for k in range(n):
            D[k].append([q[k],p[k]])

plt.xlabel('q(t)')
plt.ylabel('p(t)')
fig.savefig('espacioFasico.pdf',format='pdf')
plt.show()

'''
Apartado ii): Obtener el área de D_t para t=1/4 y su intervalo de error. Comprobar
además si se cumple el teorema de Liouville
'''

t = 1/4

#Área de D0: es exacta porque lo hemos construido nosotros explícitamente
delta = 10**(-3)

hullD0 = ConvexHull(D[0])
ax = plt.axes(xlabel = 'q(t)',ylabel = 'p(t)')
fig = convex_hull_plot_2d(hullD0,ax)
fig.savefig('D0.pdf',format='pdf')
plt.show()

areaD0 = hullD0.volume

print('Área de D0: {:.5f}'.format(areaD0))


#Área de Dt: requiere aproximarse
delta = 10**(-3)

hullDt = ConvexHull(D[int(t/delta)])
ax = plt.axes(xlabel = 'q(t)',ylabel = 'p(t)')
fig = convex_hull_plot_2d(hullDt,ax)
fig.savefig('Dt.pdf',format='pdf')
plt.show()

aristainf_0 = [[q0s[i],p0s[0]] for i in range(m1)]
aristainf_t = obtenDt(aristainf_0,t,delta)
hullaristainf_t = ConvexHull(aristainf_t)
ax = plt.axes(xlabel = 'q(t)',ylabel = 'p(t)')
fig = convex_hull_plot_2d(hullaristainf_t,ax)
fig.savefig("AristaInf_t.pdf",format='pdf')
```

```
plt.show()


aristader_0 = [[q0s[m1-1],p0s[i]] for i in range(m2)]
aristader_t = obtenDt(aristader_0,t,delta)
hullaristader_t = ConvexHull(aristader_t)
ax = plt.axes(xlabel = 'q(t)',ylabel = 'p(t)')
fig = convex_hull_plot_2d(hullaristader_t,ax)
fig.savefig('AristaDer_t.pdf',format='pdf')
plt.show()

areaDt = hullDt.volume - hullaristainf_t.volume - hullaristader_t.volume

areaDtant = -1
error = -1
errorant = -1

while errorant == -1 or error/errorant >= 0.5:
    delta /= 2
    areaDtant = areaDt
    areaDt = obtenareaDt(q0s,p0s,delta)

    errorant = error
    error = abs(areaDt-areaDtant)

print("Área de Dt: {:.5f} | Error: {:.5e}".format(areaDt,error))


'''
Apartado iii): Realizar una animación de D_t para t entre 0 y 5
'''

delta = 10**(-3)

fig = plt.figure(figsize = (10,10))
ani = animation.FuncAnimation(fig, fAnimation,range(0,int(5//delta),200),
                              fargs = (D,m1,m2),interval = 20)
ani.save("animación.gif", fps = 5)
```

Apéndice B Resultado de la ejecución

```
Área de D0: 1.00000
Área de Dt: 0.99999 | Error: 5.92233e-05
```