Práctica 6: Espacio fásico

Guillermo Martín Sánchez

12/05/20

1 Introducción

El objetivo de esta práctica es el estudio del espacio fásico del oscilador no lineal, con hamiltoniano

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

cuyas ecuaciones de Hamilton-Jacobi conducen a la ecuación diferencial:

$$\begin{aligned} \dot{q} &= \frac{\partial H}{\partial p} = 2p \\ \dot{p} &= -\frac{\partial H}{\partial q} = -q(q^2-1) \end{aligned} \quad \Rightarrow \quad \ddot{q} = -2q(q^2-1) \tag{2}$$

que nos expresa la evolución del sistema $q(t)$ y $p(t) = \dot{q}/2$. Tomamos como valores iniciales $D_0 := [0,1] \times [0,1]$.

2 Material usado

2.1 Ejercicio 1

Representa gráficamente el espacio fásico D de las órbitas finales del sistema S con las condiciones iniciales D_0. Considera al menos 10 órbitas finales diferentes.

Primero discretizamos el sistema, usando una granularidad del parámetro tiempo $t = n\delta$ con cierto $\delta \in [10^{-4}, 10^{-3}]$ y discretizando la ecuación diferencial Eq 2 como:

$$\begin{aligned} q_{n+2} &= \delta^2(-2q_n(q_n^2-1)) - q_n + 2q_{n+1} \\ p_n &= \frac{q_{n+1}-q_n}{2\delta} \end{aligned} \tag{3}$$

Tenemos que determinar (q_0, q_1). Pero esto es equivalente a determinar (q_0, p_0) ya que $q_1 = q_0 + 2\delta p_0$. Para tomar (q_0, p_0) cogemos los puntos de una malla en D_0. Es decir, cogemos 12 puntos equiespaciados en $[0, 1]$ (llamemos a este conjunto C) y dibujamos las órbitas con $(q_0.p_0) \in C \times C$ usando las ecuaciones Eq 3 como evolución, con $\delta = 10^{-3}$ y hasta $n = 16/\delta$.

2.2 Ejercicio 2

Obtén el valor del área de D y una estimación de su intervalo de error, presentando los valores de forma científicamente formal. ¿Se cumple el teorema de Liouville entre D_0 y D? Razona la respuesta.

Para calcular el área usamos el Algoritmo 3.2.1. de la página 56 con métodos de integración del Trapezoide y de Simpson. Como las órbitas son periódicas, concéntricas y no se pueden intersecar (por la unicidad de solución), concluimos que la órbita que encierra más área es aquella que pertenece a una condición inicial en la frontera de D. En concreto calculamos a_{max} usando como condiciones iniciales el punto $(q_0, p_0) = (0, 1)$. Por otro lado como el punto $(0, 0)$ es un punto estable repulsor, cogemos un valor muy cercano dentro de D_0 para calcular el área que se queda sin cubrir por órbitas con dos lóbulos. En concreto calculamos a_{min} usando como condiciones iniciales $(q_0, p_0) = (0, 10^{-10})$. Sin embargo, vemos que sólo uno de los lóbulos de esta órbita se queda sin cubrir. Por ello calculamos el área como $A_\delta = a_{max} - \frac{a_{min}}{2}$.

Como hemos usado una granuladidad del tiempo para aproximar la solución exacta (en el caso continuo) sabemos que sólo cuando $\delta \to 0$ tenemos la solución exacta. Como computacionalmente no es posible trabajar en el continuo, vamos a obtener un error del valor calculado.

Primero, calculamos el área de D varias veces usando diferentes valores de δ. En concreto, dividimos el intervalo $[10^{-4}, 10^{-3}]$ en 11 puntos equiespaciados. Llamando a este array G, calculamos $A_{\delta=G(i)}$ para $i = 0, 1, .., 10$. Obtenemos el array de diferencias $\Delta A[i] = |A_{\delta=G(i)} - A_{\delta=G(i+1)}|$ con $i = 0, .., 9$ y buscamos el percentil 90. Como en este caso tenemos $|\Delta A| = 10$, basta ordenar el array de menor a mayor (obteniendo $\Delta A'$) y quedarse con el noveno elemento: $\Delta A'[8]$.

Finalmente, como conforme disminuimos el valor de δ obtenemos cada vez el resultado con menor error, y sabemos que nuestros resultados siempre son una cota superior de la solución exacta, podemos expresar el resultado como $A_{\delta=10^{-4}} - \Delta A'[8]$.

Por otro lado se nos pregunta si se cumple el Teorema de Liouville. Para ver si se cumple para la interpretación de distribución de fases hemos usado el Algoritmo de la Lazada (`https://en.wikipedia.org/wiki/Shoelace_formula`) para calcular el área de D_0 a partir de su malla $C \times C$ y ver si coincide con el área de D_n para $n = 1000, 2000$ a partir de su malla (q_n, p_n) con $(q_0, p_0) \in C \times C$

3 Resultados y discusión

3.1 Ejercicio 1

En la Figura 1 vemos el espacio fásico con condiciones iniciales en D_0. Es interesante ver que cerca de $(0, 1)$ se comporta similar a un oscilador lineal pero en general hace órbitas con dos lóbulos cuando el hamiltoniano (o energía) es suficientemente grande.

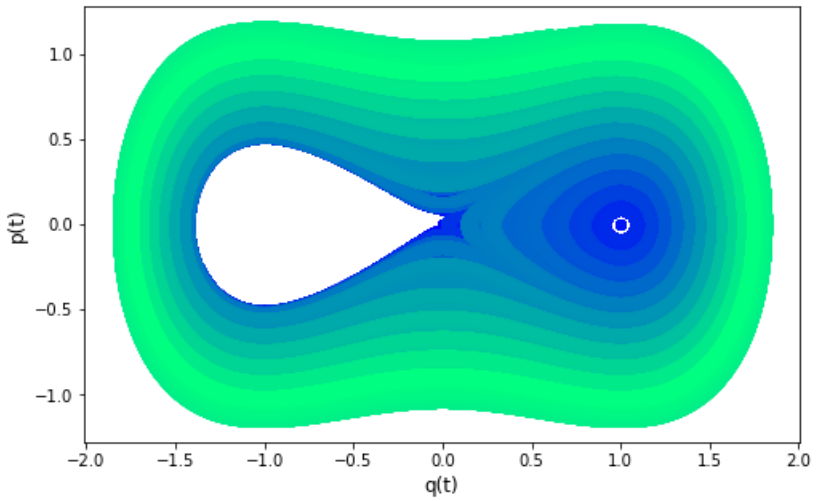

Figure 1: Espacio fásico del oscilador no lineal con condiciones iniciales en D_0

3.2 Ejercicio 2

Tanto con el método del Trapezoide como el de Simpson obtenemos un área $A = 6.21 - 0.05$.

Respecto al Teorema de Liouville tenemos dos puntos de vista:

- Respecto al espacio fásico: Si consideramos el espacio fásico con condiciones iniciales D_0 (lo que hemos llamado D) hemos obtenido que su área es $6.21 - 0.05$. Debido al Teorema de Liouville, sabemos que este área no cambia con el tiempo. En el oscilador no lineal, el hamiltoniano no depende del tiempo (Eq 1)

- Respecto a la distribución de fases: Hemos calculado el área de D_0 (Fig 2) y hemos obtenido, lógicamente, que su área es 1. Después, hemos calculado D_{1000} y D_{2000} (Fig 3) y también hemos obtenido que su área es 1. Por tanto, al menos en estos ejemplos, el Teorema de Liouville respecto a la conservación del área de la distribución de fases, se cumple.

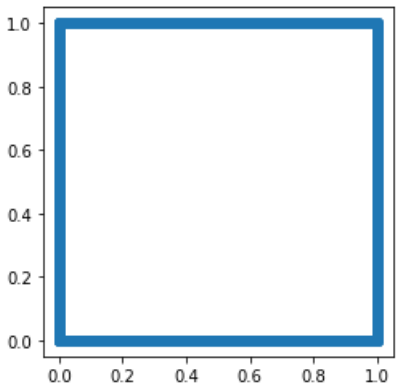

Figure 2: La frontera de D_0

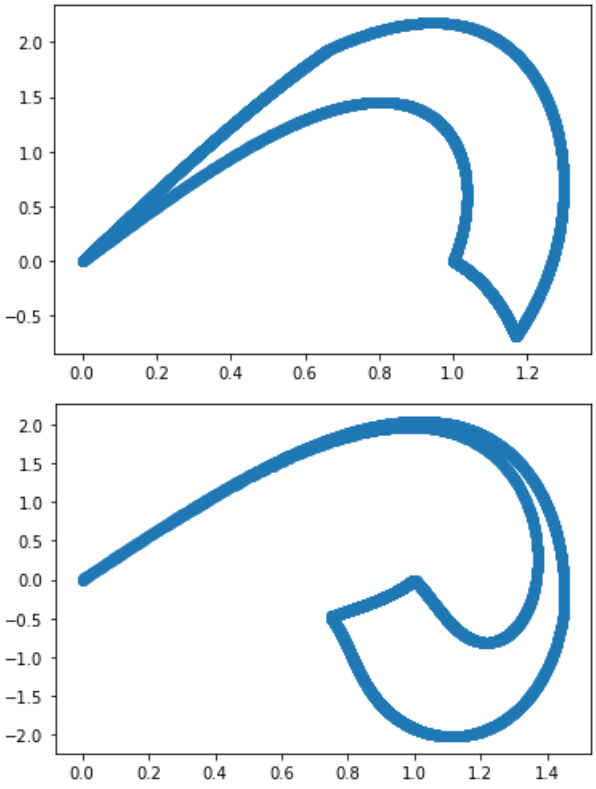

Figure 3: Arriba, la frontera de D_{1000}. Abajo, la frontera de D_{2000}

4 Conclusión

Hemos estudiado en profundidad el espacio fásico del oscilador no lineal. Lo hemos podido representar y calcular su área gracias a la discretización del sistema continuo y al uso de herramientas numéricas como la integración numérica (método del Trapezoide y de Simpson) y algoritmos de geometría computacional (Algoritmo de la Lazada). Hemos entendido las dos interpretaciones del Teorema de Liouville y hemos comprobado que se cumplen.

5 Código

```
# -*- coding: utf-8 -*-
"""
Coursework 6: Phase space
"""
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps
from numpy import trapz
os.getcwd()

# Compute the derivative dq from an array of consecutive points q with initial
# derivative dq0 with timestep d
def deriv(q,dq0,d=0.001):
   dq = (q[1:len(q)]-q[0:(len(q)-1)])/d
   dq = np.insert(dq,0,dq0)
   return dq

# Differential equation
def F(q):
    ddq = -2*q*(q**2-1)
    return ddq

# Compute the orbit of the differential equation F with initial values q0 and dq0,
# n points and timestep d
def orb(n,q0,dq0, F, 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
```

```
# Plot an orbit of the differential equation F with initial values q0 and dq0,
# n points and timestep d with color col and linestyle marker
def simplectica(q0,dq0,F,col=0,d = 10**(-4),n = int(16/10**(-4)),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))

# Compute the period of the array q with timestep d, starting the period in a
# maximum (max=True) or a minimum (min=False)
def periods(q,d,max=True):
    epsilon = 5*d
    dq = deriv(q,dq0=None,d=d)
    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

# Compute and aproxmation of the area inside the orbit with initial values q0, dq0 and
# timestep d. Use method of integration funcarea and plot=True to plot the orbit
def area(q0,dq0,d,funcarea, plot=False):
    n = int(64/d)
    q = orb(n,q0=q0,dq0=dq0,F=F,d=d)
    dq = deriv(q,dq0=dq0,d=d)
    p = dq/2

    if plot:
        # Plot orbit
        fig, ax = plt.subplots(figsize=(5,5))
        plt.rcParams["legend.markerscale"] = 6
        ax.set_xlabel("q(t)", fontsize=12)
        ax.set_ylabel("p(t)", fontsize=12)
        plt.plot(q, p, '-')
        plt.show()

    # Compute period of orbit between minima
    T, W = periods(q,d,max=False)

    if plot:
        # Plot one period of the orbit
        plt.plot(q[W[0]:W[1]],p[W[0]:W[1]])
        plt.show()
```

```
    # Take half of the period obit
    mitad = np.arange(W[0],W[0]+np.int((W[1]-W[0])/2),1)

    # Integrate with numeric method
    area = funcarea(p[mitad],q[mitad])
    return 2*area

# Compute with Shoelace formula area of polygon with vertices (x,y)
def areapol(x,y):
    return 0.5*np.abs(np.dot(x,np.roll(y,1))-np.dot(y,np.roll(x,1)))

# Compute the 90th-percentile of array areas
def error(areas):
    errors = np.zeros(10)
    for i in range(1,len(areas)):
        errors[i-1] = np.abs(areas[i-1] - areas[i])
    errors = np.sort(errors)
    return errors[8]


#%%
"""
Plot numerical solution
"""

# Example of a solution
q0 = 0.
dq0 = 1.
fig, ax = plt.subplots(figsize=(12,5))
plt.ylim(-1.8, 1.8)
plt.rcParams["legend.markerscale"] = 6
ax.set_xlabel("t = n $\delta$", fontsize=12)
ax.set_ylabel("q(t)", fontsize=12)
iseq = np.array([1,1.1,1.5,1.8,3])
for i in iseq:
    d = 10**(-i)
    n = int(32/d)
    t = np.arange(n+1)*d
    q = orb(n,q0=q0,dq0=dq0,F=F,d=d)
    plt.plot(t, q, 'ro', markersize=0.5/i,label='$\delta$ ='+str(np.around(d,3)),
             c=plt.get_cmap("winter")(i/np.max(iseq)))
    ax.legend(loc=3, frameon=False, fontsize=12)


# Orbit of the solution in phase space (q, p)
```

```
dq = deriv(q,dq0=dq0,d=d)
p = dq/2
fig, ax = plt.subplots(figsize=(5,5))
plt.rcParams["legend.markerscale"] = 6
ax.set_xlabel("q(t)", fontsize=12)
ax.set_ylabel("p(t)", fontsize=12)
plt.plot(q, p, '-')
plt.show()

#%%
"""
Exercise 1: Plot phase space
"""


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

# 12 orbits in D0 = [0,1]x[0,1] (p0 = dq0/2)
seq_q0 = np.linspace(0.,1.,num=12)
seq_dq0 = np.linspace(0.,2,num=12)
for i in range(len(seq_q0)):
    for j in range(len(seq_dq0)):
        q0 = seq_q0[i]
        dq0 = seq_dq0[j]
        ax = fig.add_subplot(1,1, 1)
        col = (1+i+j*(len(seq_q0)))/(len(seq_q0)*len(seq_dq0))
        simplectica(q0=q0,dq0=dq0,F=F,col=col,marker='ro',d= 10**(-3),n=int(16/d))
ax.set_xlabel("q(t)", fontsize=12)
ax.set_ylabel("p(t)", fontsize=12)
plt.show()


#%%
"""
Exercise 2: Area of the phase space
"""

# Values of timestep d to vary
deltavec = np.linspace(10**-4,10**-3,11)

# Compute the area of the orbits with intial values in D0

# with trapezoid integration method
i = 0
areastrap = np.zeros(11)
```

```
for d in deltavec:
    areamax = area(0.,2.,d,trapz)
    areamin = area(0.,10**(-10),d,trapz)
    areastrap[i] = areamax - areamin/2
    i += 1

print("The area with the trapezoid rule is",
      "{:.3g} - {:.1g}".format(areastrap[0],error(areastrap)))

# with Simpson integration method
i = 0
areassimps = np.zeros(11)
for d in deltavec:
    areamax = area(0.,2.,d,simps)
    areamin = area(0.,10**(-10),d,simps)
    areassimps[i] = areamax - areamin/2
    i += 1

print("The area with the Simpson rule is",
      "{:.3g} - {:.1g}".format(areassimps[0],error(areassimps)))


# Check Liouville theorem for D0 = [0,1]x[0,1]
points = 1000
x = np.concatenate((np.zeros(points),np.linspace(0,1,points),
                    np.ones(points), np.linspace(1,0,points)))
y = np.concatenate((np.linspace(0,1,points),np.ones(points),
                    np.linspace(1,0,points), np.zeros(points)))

# Print area of D0
print("The area of D0 is {:.2g}".format(areapol(x,y)))

plt.figure()
plt.axis('scaled')
plt.scatter(x,y)

# Evolve D0 n steps
n = 2000
x2 = np.zeros_like(x)
y2 = np.zeros_like(y)
for i in range(0,len(x)):
    q0 = x[i]
    dq0 = y[i]/2
    q = orb(n,q0,dq0,F)
    dq = deriv(q,dq0)
    x2[i] = q[n]
```

```
    y2[i] = dq[n]*2

plt.figure()
plt.scatter(x2,y2)

# Print area of Dn
print("The area of Dn is {:.2g}".format(areapol(x2,y2)))
```