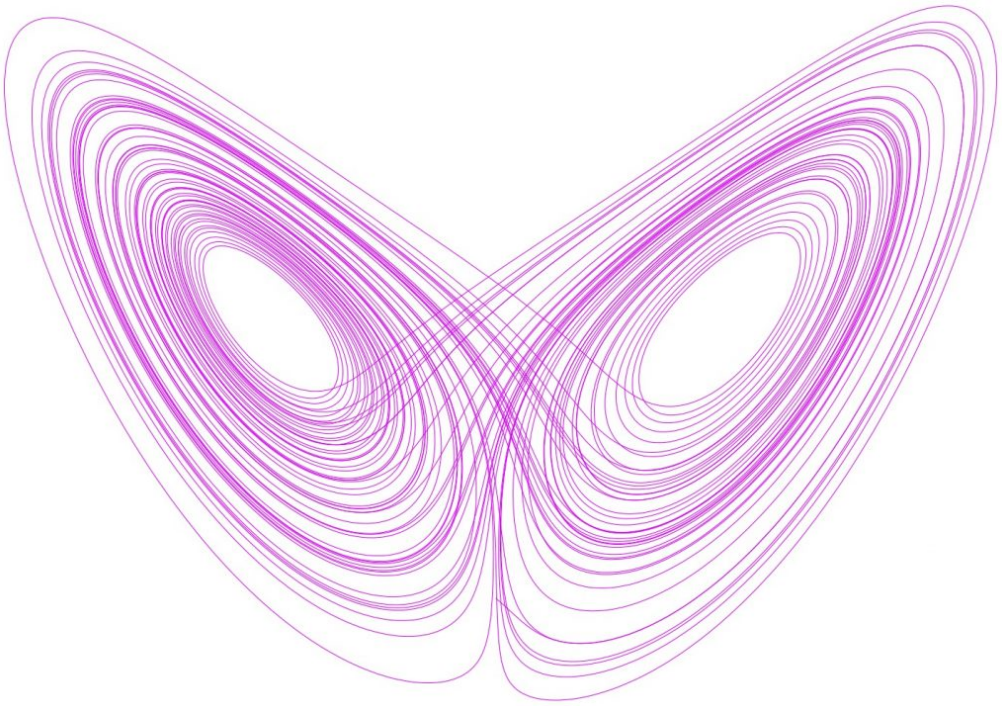
GEOMETRÍA COMPUTACIONAL

ESPACIO FÁSICO



LUCAS DE TORRE

Índice

1. Introducción

Partimos del hamiltoniano de un oscilador no lineal, $H(q,p)$, y, por las ecuaciones de Hamilton-Jacobi, tenemos también $\ddot{q}$:

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

que describe la evolución de $q(t)$ y $p(t) = \dot{q}/2$.

Suponiendo que tenemos un conjunto de condiciones iniciales $D_0 := [0,1] \times [0,1]$ y una granuralidad del parámetro temporal $t = n\delta$, tal que $\delta \in (10^{-4}, 10^{-3})$ y $n \in \mathbb{N} \cup \{0\}$, buscamos representar gráficamente el espacio fásico D de las órbitas finales del sistema S con las condiciones iniciales D_0 considerando al menos 10 órbitas.

Además, obtendremos el área de D y una estimación de su error y verificaremos que se cumple el teorema de Liouville entre D_0 y D.

2. Material

Primero, definimos, en la plantilla proporcionada, $F(q) = -2q(q^2-1)$, que es la ecuación diferencial de nuestro sistema S. Además, definimos $\delta = 10^{-3,5}$ y n como la parte entera de $16/\delta$.

Después, para la representación gráfica, tenemos en cuenta las órbitas generadas por distintos valores iniciales q_0 (entre 0 y 1) y $\dot{q}_0$ (entre 0 y 2, ya que $\dot{q} = 2p$). Tomamos 121 combinaciones posibles de valores iniciales (tomamos 11 valores equiespaciados en [0,1] para los valores iniciales q_0 y otros 11 valores equiespaciados en [0,2] para los valores $\dot{q}_0$ y hacemos todas las combinaciones posibles) y hallamos sus órbitas finales haciendo uso de la función *simplectica*.

Con estas órbitas finales del sistema S, representamos gráficamente el espacio fásico D.

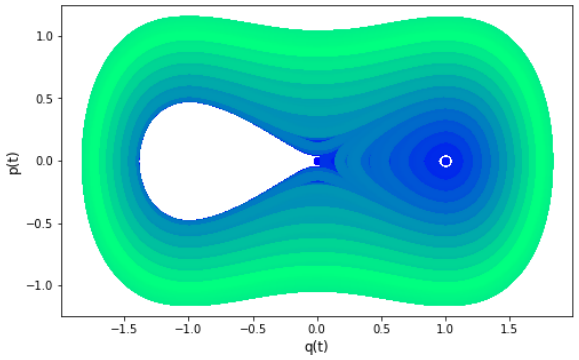
Ahora, para hallar el área del conjunto D, repetimos el proceso anterior para hallar el espacio fásico. Para cada par de datos iniciales, hallamos el área haciendo uso de la *Regla de Simpson*. Como lo hacemos 121 veces, nos quedamos con los valores a_{max} y a_{inf}. Tenemos en cuenta que estamos hallando la mitad del valor de las áreas, porque consideramos solo la "mitad superior" del espacio fásico (que es simétrico) para integrar más fácilmente. Además, a_{inf} es el doble del área que engloba a la región interna que queda sin ser cubierta de órbitas, porque la región sin cubrir se sitúa a la izquierda del origen, pero las órbitas presentan simetría respecto al origen. Así, hallamos el área de D como area $D = 2a_{max} - a_{inf}$.

Con intención de analizar el error, repetimos el proceso explicado en el párrafo anterior 11 veces, haciendo que δ tome 11 valores equiespaciados en $(10^{-3}, 10^{-4})$ y consideramos como resultado el área obtenida con el δ más pequeño. Ahora, ordenando las 11 áreas resultantes en función del tamaño del δ usado para su cálculo (siendo la primera área la hallada con el valor de δ más cercano a 10^{-3} y la última la hallada con el δ más cercano a 10^{-4}), hallamos el valor absoluto de las 10 diferencias posibles al restar un área menos el área hallada con el menor δ (la "mejor" área). Por último, ordenamos esas diferencias y nos quedamos con el noveno valor, que se corresponde con el error del área de D calculado con cuantil de orden 0,9.

Por último, para comprobar que se cumple el teorema de Liouville, realizamos diferentes gráficas correspondientes al área en distintos tiempos. Para su cálculo, no utilizamos instantes de tiempo concretos, sino pequeños intervalos que nos permiten observar mejor el área en cada momento. Para ello, variamos mínimamente la función simplectica para que, en lugar de dibujar las órbitas desde el comienzo, lo haga desde un instante determinado.

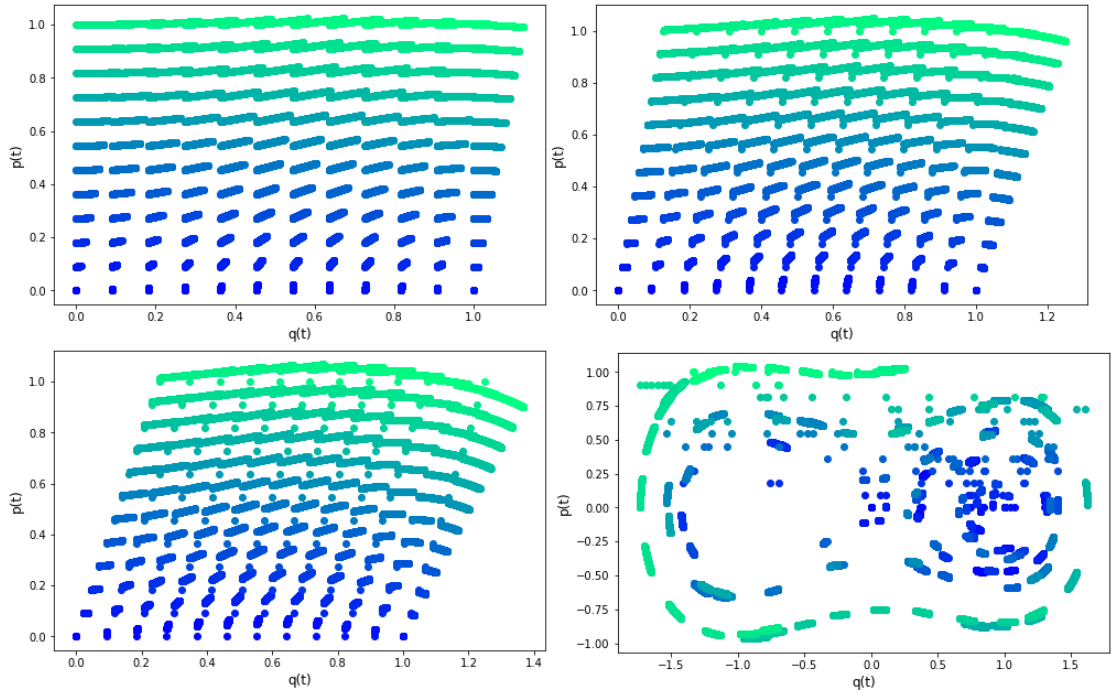
3. Resultados

En la representación gráfica del espacio D, observamos que el punto (0,0) es un punto fijo inestable (es decir, que cualquier pequeña perturbación hace que la partícula se separe de la posición de equilibrio), mientras que el punto (1,0) es un punto fijo estable (es decir, que cualquier pequeña perturbación hace que la partícula tienda a volver a la posición de equilibrio).



Por otro lado, el área de D es de 6,182 con un error de 0,119 (calculado con cuantil de orden 0,9).

A continuación vemos cuatro gráficas, las tres primeras correspondientes a instantes de tiempo iniciales y la última a un instante avanzado.



Observamos (sobre todo en las tres primeras), que el área de los conjuntos es la misma. Aunque en la última es menos evidente, también lo es. Por tanto, se verifica el teorema de Liouville.

4. Conclusiones

Resulta muy interesante lo bien que se observa como se va formando el espacio fásico, algo que se aprecia muy bien con las cuatro últimas imágenes. Llama mucho la atención como, a pesar del cambio de forma que se va produciendo, el volumen se mantiene constante.

5. Anexo A: Código

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr  8 23:53:36 2020
@author: Jorge Sainero y Lucas de Torre
"""
import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simps
#https://matplotlib.org/3.1.0/tutorials/colors/colormaps.html

os.getcwd()

#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 = 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

#Ecuaci n de un sistema din mico continuo
#Ejemplo de oscilador simple
def F(q):
    #k = 1
    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,n0=0):
    #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[n0:] #np.array(q),

def periodos(q,d,max=True):
    #Si max = True, tomamos las ondas a partir de los m ximos/picos
    #Si max == False, tomamos los las ondas a partir de los m nimos/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**(-3.5)
## Pintamos el espacio de fases
def simplectica(q0,dq0,F,col=0,d=10**(-3.5),n = int(16/d),marker='-',n0=0):
    q = orb(n,q0=q0,dq0=dq0,F=F,d=d,n0=n0)
    dq = deriv(q,dq0=dq0,d=d)
    p = dq/2
    plt.plot(q, p, marker,c=plt.get_cmap("winter")(col))

```

```
63 def print_espacio_fasico(n=int(16/d),n0=0):
64     fig = plt.figure(figsize=(8,5))
65     fig.subplots_adjust(hspace=0.4, wspace=0.2)
66     seq_q0 = np.linspace(0.,1.,num=11)
67     seq_dq0 = np.linspace(0.,2.,num=11)
68     for i in range(len(seq_q0)):
69         for j in range(len(seq_dq0)):
70             q0 = seq_q0[i]
71             dq0 = seq_dq0[j]
72             ax = fig.add_subplot(1,1, 1)
73             col = (1+i+j*(len(seq_q0)))/(len(seq_q0)*len(seq_dq0))
74             #ax = fig.add_subplot(len(seq_q0), len(seq_dq0), 1+i+j*(len(seq_q0)))
75             simplectica(q0=q0,dq0=dq0,F=F,col=col,marker='ro',d=d,n=n,n0=n0)
76     ax.set_xlabel("q(t)", fontsize=12)
77     ax.set_ylabel("p(t)", fontsize=12)
78     #fig.savefig('Simplectic.png', dpi=250)
79     plt.show()
80
81 ######################################################################
82 #   C LCULO DE   RBITAS
83 ######################################################################
84 def apartado1():
85     print("Apartado 1\n")
86
87     ######################################################################
88     #   ESPACIO F SICO
89     ######################################################################
90
91     print_espacio_fasico()
92
93
94
95
96 ######################################################################
97 #   C LCULO DEL   REA  DEL ESPACIO F SICO
98 ######################################################################
99 #Tomamos un par (q(0), p(0)) y nos quedamos s lo en un trozo/onda de la  rbita , sin
       repeticiones
100 #Para eso, tomaremos los periodos de la  rbita , que definen las ondas
101
102 #Paso1: Buscamos las condiciones iniciales que maximizan/minimizan en  rea
103 #q0 = 0.
104 #dq0 = 2.
105 #d = 10**(-3.5)
106 def area_espacio_fasico(d):
107     areas=[]
108     for q0 in np.linspace(0.,1.,num=11):
109         for dq0 in np.linspace(0.,2.,num=11):
110
111             n = int(32/d)
112             #t = np.arange(n+1)*d
113             q = orb(n,q0=q0,dq0=dq0,F=F,d=d)
114             dq = deriv(q,dq0=dq0,d=d)
115             p = dq/2
116             #Tomaremos los periodos de la  rbita , que definen las ondas
117             T, W = periodos(q,d,max=False)
118             if len(W)>1:
119
120                 #Tomamos la mitad de la "curva cerrada" para integrar m s
      f cilmente
121                 mitad = np.arange(W[0],W[0]+np.int((W[1]-W[0])/2),1)
122
123                 # Regla de Simpson
```

```
124                 area = simps(p[mitad],q[mitad])
125                 areas.append([q0,dq0,area])
126     sort_areas=sorted(areas,key=lambda x:x[2])
127     min_a=sort_areas[0][2]
128     max_a=2*sort_areas[len(sort_areas)-1][2]
129     return max_a-min_a
130 
131 def apartado2():
132     #Definimos distintos valores de d
133     iseq=np.linspace(3.001,3.999,num=11)
134     #Hallamos el  rea  de cada espacio f sico para cada d
135     areas=[area_espacio_fasico(10**(-d)) for d in iseq]
136     #Hallamos el valor absoluto de las diferencias de cada  rea  comparada con "la
        mejor"
137     resta_areas=[abs(areas[i]-areas[10]) for i in range(len(areas)-1)]
138     sort_resta_areas=sorted(resta_areas)
139     #Cogemos el cuantil de orden 0,9
140     print("El  rea  calculada es:",round(areas[10],3),
141           "con un error de", round(abs(areas[10]-sort_resta_areas[8]),3))
142 
143     ######################################################################
144     ################## TEOREMA DE LIOUVILLE ##############################
145     ######################################################################
146 
147     for i in range(3):
148         print_espacio_fasico((i+1)*200,i*200)
149 
150     print_espacio_fasico(int(16/(10**(-3.5))),int(16/(10**(-3.5)))-200)
151 
152 apartado1()
153 apartado2()
```