MEMORIA DE LA PRÁCTICA 1

Mario Jiménez Gutiérrez

February 2022

1 Introducción

En primer lugar, en esta práctica estudiamos el sistema dinámico no lineal definido por la función logística:

$$x_{n+1} = r * (1 - x_n) * x_n$$

teniendo que $r > 0$ y $x_n \in [0, 1]$. El objetivo será obtener conjuntos atractores, así como sus conseguientes intervalos de errores. Se calculan órbitas, períodos y errores de ellas para cada r y x_0

2 Material usado

Se ha usado el entorno de desarrollo Spyder. Se crea el archivo JimenezMario-pract1.py que contiene la solución. Se emplea la librería numpy para operar y además la librería matplotlib.pyplot para representar gráficos.
Por otro lado, tenemos pocos datos de partida, se especifican $\epsilon = 0.001$, aunque en algunas funciones se modifica para ver cómo varía la solución. Los valores por defecto de N es de 200(número de puntos de la órbita) y el $N0$ vale 50.

3 Resultados

Para el apartado 1 escojo el valor de $x0 = 0.5$ y modifico el valor de r de manera aleatoria. Primero, se dibuja la correspondiente órbita con los N, N0 y ϵ dados, 200, 50 y 0.001. Posteriormente, con dichos valores se ejecuta la función atrac con los valores de x_0 y r ya fijados. En atrac después llamamos a las funciones orbita que calculan la órbita de los valores dados y con los M últimos elementos de dicha órbita calculamos el periodo; con dicho periodo obtenemos los últimos periodo elementos. De esta manera hemos obtenido el conjunto de atractores.
Es importante notar que los valores de N, M y ϵ influirán en el resultado obtenido.
Posteriormente llamamos a $errores_aux$, dicha función va calculando órbitas del último elemento que al principio es el último elemento de V0, el conjunto de

atractores, con dichas órbitas (cogiendo solo los últimos periodo elementos) vemos las que tienen mayor diferencia con respecto a las orbitas originales y lo añadimos al array de errores. Iteramos este proceso hasta que el último error sera 0 o el último error sea igual al penúltimo calculado.
Vemos como en ambos casos el error va disminuyendo.

En concreto, los resultados obtenidos son:
Para $r = 3.18$ y $x_0 = 0.5$ son:(Conjunto atractor y error):
(array([0.52084749, 0.79361791]), 0.0)

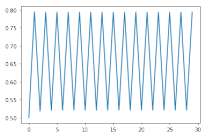

Figure 1: conjunto1

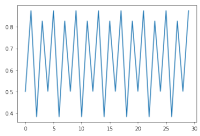

Figure 2: conjunto2

Para $r = 3.18$ y $x_0 = 0.5$:
(array([0.38281968, 0.50088421, 0.82694071, 0.87499726]), 0.0)

Posteriormente, analizamos la estabilidad, calculando el conjunto atractor para valores cerca de x0, y las bifurcaciones , modificando el r para valores que estén cerca. En nuestro caso, los valores están hasta $\delta = 0.2$
Se muestran ambos para el 2^{o} conjunto de atractores.

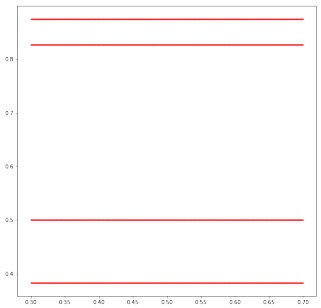

Figure 3: Estabilidad conjunto2

Para calcular el apartado 2 tomamos los elementos desde 3.544 hasta 4 con paso 0.005 y calculamos el conjunto atractor para cada uno de ellos con

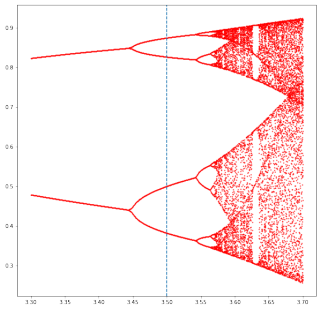

Figure 4: Bifurcaciones conjunto2

$x_0 = 0.5$ fijado, los valores de N, N0 ya fijados y vemos cuáles tienen tamaño 8. Nos sale este array: Valores de r con conjunto atractor de 8 elementos [3.544, 3.5445, 3.5450000000000004, 3.5455000000000005, 3.5460000000000007, 3.546500000000001, 3.547000000000001, 3.547500000000001, 3.5480000000000014, 3.5485000000000015, 3.5490000000000017, 3.549500000000002, 3.550000000000002, 3.550500000000002, 3.5510000000000024, 3.5515000000000025, 3.5520000000000027, 3.552500000000003, 3.553000000000003, 3.553500000000003, 3.5540000000000034, 3.5545000000000035, 3.5550000000000037, 3.555500000000004, 3.556000000000004, 3.556500000000004, 3.5570000000000044, 3.5575000000000045, 3.5580000000000047, 3.558500000000005, 3.559000000000005, 3.559500000000005, 3.5600000000000054, 3.5605000000000055, 3.5610000000000057, 3.561500000000006, 3.562000000000006, 3.562500000000006, 3.5630000000000064, 3.5635000000000066, 3.5640000000000067, 3.5965000000000176, 3.666000000000041, 3.888000000000115, 3.8995000000001188, 3.9610000000001393]
Cogemos el último r, con dicho r tenemos este conjunto atractor y error(como analizado anteriormente (array([0.03824848, 0.03898555, 0.1457075 , 0.14840156, 0.49305269, 0.50058538, 0.99005882, 0.99024864]), 1.446620601086579e-13) Los gráficos anteriores de su órbita, estabilidad y bifurcación de dicho conjunto son los siguientes:

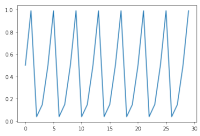

Figure 5: conjunto calculado

4 Conclusión

Esta práctica ha sido útil para experimentar con los conceptos teóricos vistos en clase sobre sistemas dinámicos no lineal y conjuntos atractores. Hemos no-

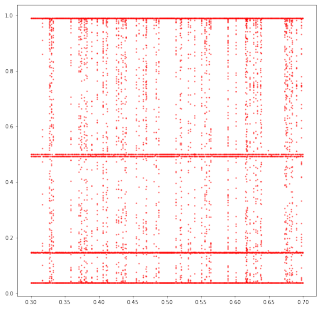

Figure 6: Estabilidad conjunto calculado

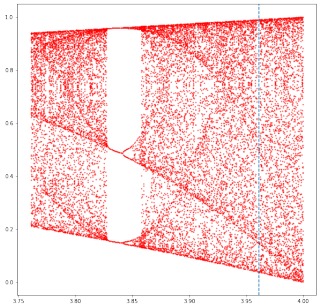

Figure 7: Bifurcaciones conjunto calculado

tado que cambio en los parámetros producen grandes cambios en los resultados. Además de resaltar la importancia de ser conscientes de los intervalos de errores

5 Anexo

```
##Mario Jimenez Gutierrez
import os
import matplotlib.pyplot as plt
import numpy as np#import math as mt

workpath = "C:/"
os.getcwd()
files = os.listdir(workpath)



def logistica(x, r):
    return r*x*(1-x);
```

```
def fn(x0,f,n, r):
    x = x0
    for j in range(n):
        x = f(x, r)
    return(x)

def orbita(x0,f,N, r):
    orb = np.empty([N])
    for i in range(N):
        orb[i] = fn(x0, f, i, r)
    return orb

def periodo(suborb, epsilon=0.001):
    N=len(suborb)
    for i in np.arange(2,N-1,1):
        if abs(suborb[N-1] - suborb[N-i]) < epsilon :
            break
    return(i-1)

def atrac(f, x0,r, N0 = 200, N = 50, epsilon=0.001):
    orb = orbita(x0,f,N0,r)
    ult = orb[-1*np.arange(N,0,-1)]
    per = periodo(ult, epsilon)
    V0 = np.sort([ult[N-1-i] for i in range(per)])
    return V0

"""Como se indica en las sugerencias, calula para valores cerca de x0 su conjunt
    atractor"""
def analisis_estabilidad(r, x0, N0 = 200, N = 50, delta = 0.2):
    x0s = np.arange(x0 - delta,x0 + delta, 0.0005)
    atractores = []

    for valor in x0s:
        atractores.append(atrac(logistica, valor, r))

    plt.figure(figsize=(10,10))

    for i in range(len(x0s)):
        for j in atractores[i]:
            plt.plot(x0s[i], j,'ro', markersize=1)

    plt.show()


def maximo(l):
    m = l[0]
```

```
    for i in range(len(l)):
        if l[i]>m:
            m = l[i]
    return m
def errores_aux(f, r, V0,  N0 = 200, N = 50, epsilon = 0.001):
    period = len(V0)
    ultimo = V0[-1]
    errores = []
    for j in range(N):
        orbit = orbita(ultimo, f, period, r)
        ultimo = orbit[-1]
        ult_aux = np.sort(orbit[-period:])
        errores.append(maximo([abs(V0[i]- ult_aux[i]) for i in range(period)]))
        if errores[-1] == 0 or (len(errores) >= 2 and errores[-1] == errores[-2]
            break
    return ( ult_aux, errores[-1] )

    """ Como se indica en las sugerencias, calula para valores cerca de r su con
    atractor"""
def analisis_bifurcaciones(x0,r,  N0= 200, N = 50, delta = 0.2):
    rss = np.arange(r - delta,r + delta, 0.0005)
    V0s = np.empty([N,len(rss)])
    V0s *=float("nan")
    for i in range(len(rss)):
        V0 = atrac(logistica,x0, rss[i], N0, N)
        V0s[range(len(V0)),i] = V0

    plt.figure(figsize=(10,10))

    for j in range(N):
        plt.plot(rss, V0s[j,], 'ro', markersize=1)
    plt.xlabel = "r"
    plt.ylabel = "V0"

    plt.axvline(x=r, ls="--")
    plt.show()



"""Apartado 1.a"""
print("Apartado 1.a\n")
plt.plot(orbita(0.5, logistica, 30, 3.18 ))
atractores = atrac(logistica, 0.5, 3.18)
print(atractores)

print(errores_aux(logistica,3.18,  atractores))
```

```
analisis_estabilidad(3.18, 0.5)
analisis_bifurcaciones(0.5, 3.18)



"""Apartado 1.b"""
print("Apartado 1.b|n")
plt.plot(orbita(0.5, logistica, 30, 3.50))
atractores = atrac(logistica, 0.5, 3.50)
print(atractores)
print(errores_aux(logistica,3.50, atractores))
analisis_estabilidad(3.50, 0.5)
analisis_bifurcaciones( 0.5,3.50)
"""Apartado 2"""
print("Apartado 2|n")
rss = np.arange(3.544,4, 0.0005)



"""Apartado 2"""
print("Apartado 2|n")

rss = np.arange(3.544,4, 0.0005)
valor_aux = 0 ##un valor que tiene tamano 8
print("Valores de r con conjunto atractor de 8 elementos")
lista = []
for i in range(len(rss)):
    r = rss[i]
    V0 = atrac(logistica, 0.5, r)
    if len(V0)==8 :
        lista.append(r)
        valor_aux = r

print(lista)
plt.plot(orbita(0.5, logistica, 30,valor_aux))
atractores = atrac(logistica, 0.5, valor_aux)
print(atractores)
print(errores_aux(logistica,valor_aux, atractores))
analisis_estabilidad(valor_aux, 0.5)
analisis_bifurcaciones( 0.5,valor_aux)
```