Práctica 1 - Geometría Computacional

Marcos Herrero Agustín

1. Introducción

El propósito de este trabajo es analizar la convergencia del sistema dinámico discreto no lineal definido por la sucesión $x_{n+1} = f(x_n)$ con $f(x) = rx(1-x)$. Este sistema, conocido como sistema logístico, modela el crecimiento de una población cuando los recursos son limitados. Es paradigmático de cómo, incluso con un sistema dinámico no lineal muy sencillo, aparecen comportamientos caóticos.

2. Datos y condiciones iniciales

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

- El sistema a analizar: $x_{n-1} = rx_n(1-x_n)$
- Para el apartado i), el rango $(3, 3.544)$ para r y el rango $[0, 1]$ para x_0
- Para el apartado ii), el rango $(3.544, 4)$ para r

3. Metodología

Se ha utilizado el siguiente algoritmo para el cálculo de los conjuntos atractores:

1. Calcular la órbita mediante aplicaciones sucesivas de f. El número de elementos a incorporar es un tiempo transitorio a partir del cual la amplitud del nuevo conjunto disminuye menor que el ϵ escogido inicialmente (para esta práctica, escojo siempre $\epsilon = 0.001$).
2. Buscar el periodo de la órbita, con el que se obtiene el valor aproximado de los elementos del conjunto límite.
3. Calcular los errores de los elementos del conjunto límite.
4. Comprobar la estabilidad del conjunto límite, para ver si es un conjunto atractor.

Para el apartado i), se han escogido valores de r y x_0 en los intervalos proporcionados ($r = 3.4$ y $x_0 = 0.5$ para el primer conjunto, y $r = 3.3$ y $x_0 = 0.2$ para el segundo conjunto) y se ha aplicado sobre ellos el algoritmo anterior.

Para el apartado ii), se ha escogido $x_0 = 0.5$ y se han recorrido los r en el intervalo $(3.544, 4)$ (a paso 0.001) comprobando para cada uno de ellos, con el algoritmo anterior, si genera un conjunto límite de tamaño 8. Se proporciona como ejemplo el primer conjunto atractor de cardinal 8 encontrado.

4. Resultados

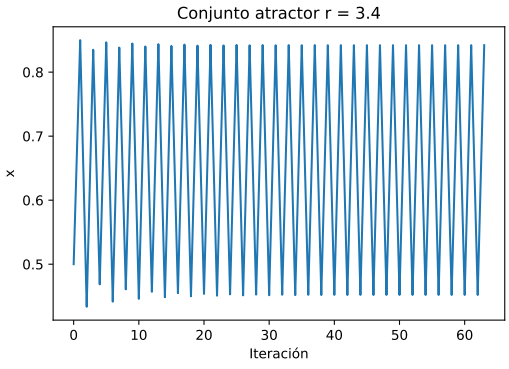
4.1. Apartado i)

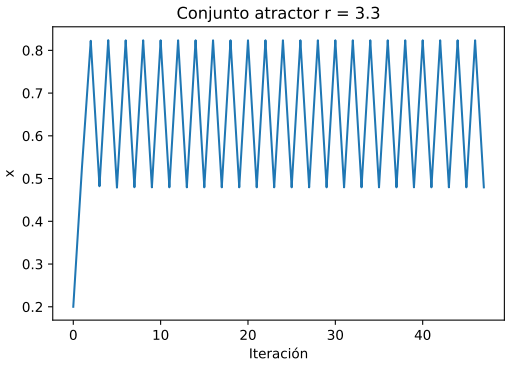
Para $r = 3.4$ y $x_0 = 0.5$, se ha encontrado un conjunto atractor de cardinal 2, cuyos elementos son:

$$0.45196 \pm 10^{-5}, 0.842153 \pm 4 * 10^{-6}$$

Para $r = 3.3$ y $x_0 = 0.2$, se ha encontrado un conjunto atractor de cardinal 2, cuyos elementos son:

$$0.47942701982424 \pm 2 * 10^{-14}, 0.823603283206067 \pm 8 * 10^{-15}$$

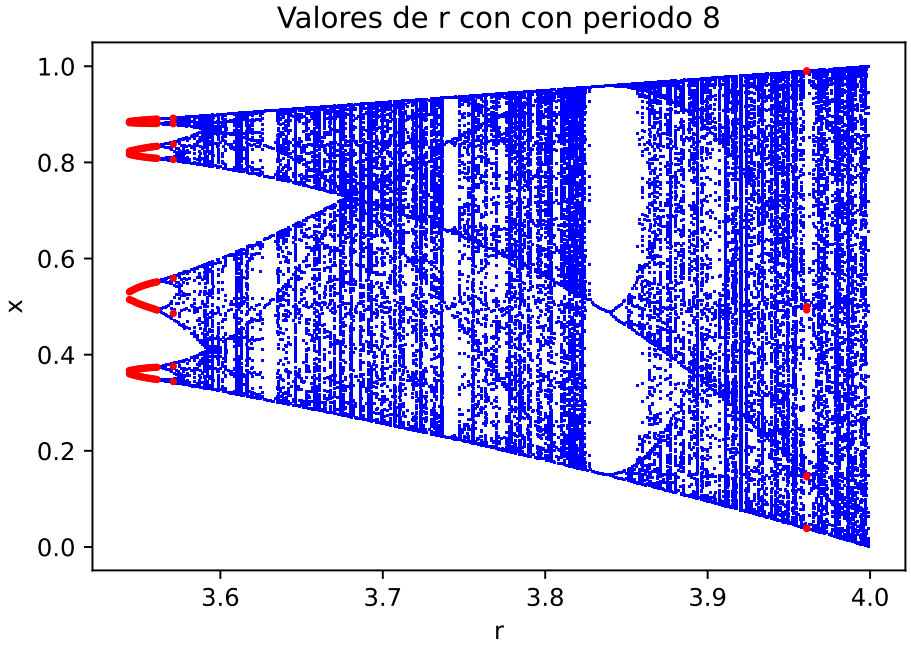




4.2. Apartado *ii)*

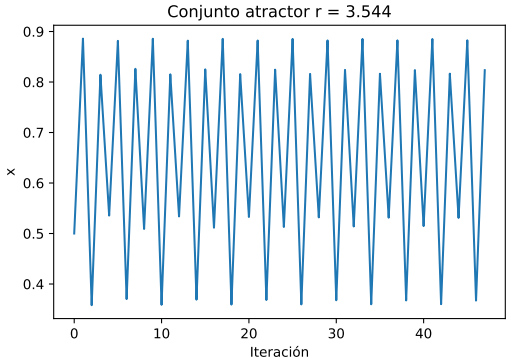
Los valores de r en el intervalo $(3.544, 4)$ para los que el conjunto límite tiene 8 elementos son:

- Todos los del intervalo $(3.544 \pm 0.001, 3.561 \pm 0.001)$
- 3.567 ± 0.001
- 3.571 ± 0.001
- 3.961 ± 0.001



Como ejemplo de conjunto atractor de 8 elementos damos el correspondiente a $r = 3.544$. Está formado por los siguientes 8 elementos:

$$0.3600 \pm 0.0002, 0.3671 \pm 0.0003, 0.5147 \pm 0.0008, 0.5308 \pm 0.0005$$
$$0.8166 \pm 0.0002, 0.8234 \pm 0.0003, 0.8826 \pm 0.0001, 0.88524 \pm 8 * 10^{-5}$$



5. Discusión y conclusiones

En todas las ejecuciones del algoritmo de búsqueda de atractores se ha requerido un tiempo transitorio bastante pequeño (entre 40 y 70 iteraciones) para lograr reducir la diferencia de amplitudes por debajo del valor de ϵ. Este tiempo es muy inferior a la capacidad computacional. En el apartado *i)*, se ha podido comprobar que ambos conjuntos atractores calculados son estables. Esto no ha sido así en el apartado *ii)*, ya que los conjuntos de 8 elementos correspondientes a algunas de las r's ecnontradas no son estables. Por último, cabe observar que los errores obtenidos para los elementos son todos inferiores al valor de $\epsilon = 0.001$ fijado inicialmente, como se requería.

Apéndice A Código utilizado

```
"""
Práctica 1 de Geometría computacional 2021-2022
Autor: Marcos Herrero
"""

import matplotlib.pyplot as plt
import numpy as np

"""
Funciones auxiliares
"""

#Devuelve la funcion logística f de parametro r y evaluada en x
def logistica(x):
    return r*x*(1-x);

#Calcula la sucesion xn hasta encontrar un tiempo transitorio para el que
#la amplitud no varia. Se consideran  conjuntos de inicialmente tamaño m
#y mayores en cada iteracion, como en el algoritmo de la teoria.
# Devuelve la orbita obtenida.
def nuevaOrbita(x0,f,m):
    A = 0 #amplitud

    orb = np.empty(m)
    x = x0
    for i in range(m):
        orb[i] = x
        x = f(x)

    Aant = A #amplitud del conjunto anterior
    A = np.max(orb) - np.min(orb)

    while abs(A-Aant) >= epsilon :

        for i in range(m):
            orb = np.append(orb,x)
            x = f(x)

        Aant = A
        A = np.max(orb[-m:]) - np.min(orb[-m:])


    return orb

#Devuelve el periodo de la orbita para el epsilon proporcionado
#Es decir, el numero de iteraciones que separan el ultimo elemento de uno
#con el que difiere menos de epsilon
def periodo(orb, epsilon=0.001):
    N=len(orb)
    for i in np.arange(2,N-1,1):
        if abs(orb[N-1] - orb[N-i]) < epsilon :
            break
    return(i-1)

#Comprueba que el conjunto limite correspondiente V0
# generado a partir de x0, es estable
def comprobarEstabilidad(x0,per,V0,epsilon,m):

    for z in np.arange(-10*epsilon,11*epsilon,epsilon):
```

```
            x0mod = x0+z
            orbmod = nuevaOrbita(x0mod,logistica,m)
            Nmod = orbmod.size
            permod = periodo(orbmod,epsilon)

            if per != permod:
                return False

            V0mod = np.sort(orbmod[Nmod-permod:])
            dif = np.absolute(V0 - V0mod)
            maxdif = np.max(dif)

            if maxdif >= epsilon:
                return False

    return True


#Busca un conjunto atractor para r partiendo de x0.
def buscarConjuntoAtractor(r,x0,epsilon):

    #1. Calculamos la orbita, de tamaño un tiempo transitorio. Al menos tendra ta
    # m = 16 necesario para poder identificar conjuntos atractores
    # de 8 elementos y calcular sus errores

    m = 16
    orb = nuevaOrbita(x0,logistica,m)
    N = orb.size

    #2. Buscamos el periodo de la orbita obtenida y calculamos el conjunto limite

    per = periodo(orb,epsilon)

    #3. Calculamos los errores de los elementos

    V0 = per*[None]
    for i in range(per):
        V0[i] = (orb[N-per+i],abs(orb[N-per+i] - orb[N-2*per+i]))

    #4. Comprobamos la estabilidad del conjunto limite obtenido

    V0.sort()
    estable = comprobarEstabilidad(x0,per,[e[0] for e in V0],epsilon,m)

    return V0, orb, estable

"""
Apartado i): Hallar dos conjuntos atractores para r en el rango (3,3.544)
y x0 en [0,1]

"""

print("Nota: el redondeo de acuerdo a las cifras significativas se hace "
      "posteriormente a mano")


print()
print("Apartado i):")
#Primer conjunto atractor
r = 3.4
x0 = 0.5
```

```
epsilon = 0.001
print("Primer conjunto atractor: r = {}, x0 = {}".format(r,x0))
print()
V0, orb, estable = buscarConjuntoAtractor(r,x0,epsilon)

if not(estable):
    print("No se encontró un conjunto estable")

else:
    for (x,delta) in V0:
        print("{} +- {}".format(x,delta))


    plt.title("Conjunto atractor r = {}".format(r))
    plt.plot(orb)
    plt.xlabel("Iteración")
    plt.ylabel("x")
    fig = plt.gcf()
    fig.savefig("conjuntoi1.pdf",format='pdf')
    plt.show()


print()
print()

#Segundo conjunto atractor
r = 3.3
x0 = 0.2
epsilon = 0.001
print("Segundo conjunto atractor: r = {}, x0 = {}".format(r,x0))
print()
V0,orb, estable = buscarConjuntoAtractor(r,x0,epsilon)


if not(estable):
    print("No se encontró un conjunto estable")

else:
    for (x,delta) in V0:
        print("{} +- {}".format(x,delta))


    plt.title("Conjunto atractor r = {}".format(r))
    plt.plot(orb)
    plt.xlabel("Iteración")
    plt.ylabel("x")
    fig = plt.gcf()
    fig.savefig("conjuntoi2.pdf",format='pdf')
    plt.show()


print()
print()

"""
Apartado ii): Estimar los valores de r en (3.544, 4) para los que el conjunto
atractor tiene 8 elementos
"""


print("Apartado ii)")
```

```
x0 = 0.5
epsilon = 0.001
rerror = 0.001# error en la r

rs = np.arange(3.544,4,rerror)
rs8el = []
ejemplo = None
rejemplo = 0

xeje = []
yeje = []
xejeesp = []
yejeesp = []

for r in rs:
    V0,orb, _ = buscarConjuntoAtractor(r, x0, epsilon)
    conj = [V0[i][0] for i in range(len(V0))]

    for elem in conj:
        xeje.append(r)
        yeje.append(elem)


    if len(V0) == 8:
        rs8el.append(r)

        for elem in conj:
            xejeesp.append(r)
            yejeesp.append(elem)

        if ejemplo == None:
            ejemplo = V0
            rejemplo = r

            plt.title("Conjunto atractor r = {}".format(round(r,3)))
            plt.plot(orb)
            plt.xlabel("Iteración")
            plt.ylabel("x")
            fig = plt.gcf()
            fig.savefig("conjuntoii.pdf",format='pdf')
            plt.show()

#print(V0s)

plt.title("Valores de r con con periodo 8")
plt.plot(xeje,yeje,'b,',markersize=0.01)
plt.plot(xejeesp,yejeesp,'ro', markersize=2)
plt.xlabel("r")
plt.ylabel("x")
fig = plt.gcf()
fig.savefig("rs8elems.pdf",format='pdf')
plt.show()

print("Los r encontrados para los que el conjunto atractor tiene 8 elementos son:
for r in rs8el:
    print("{} +- {}".format(r,rerror))

print()

print("Un ejemplo de conjunto atractor (r = {}):".format(rejemplo))
```

```
for (x,delta) in ejemplo:
    print("{} +- {}".format(x,delta))
```