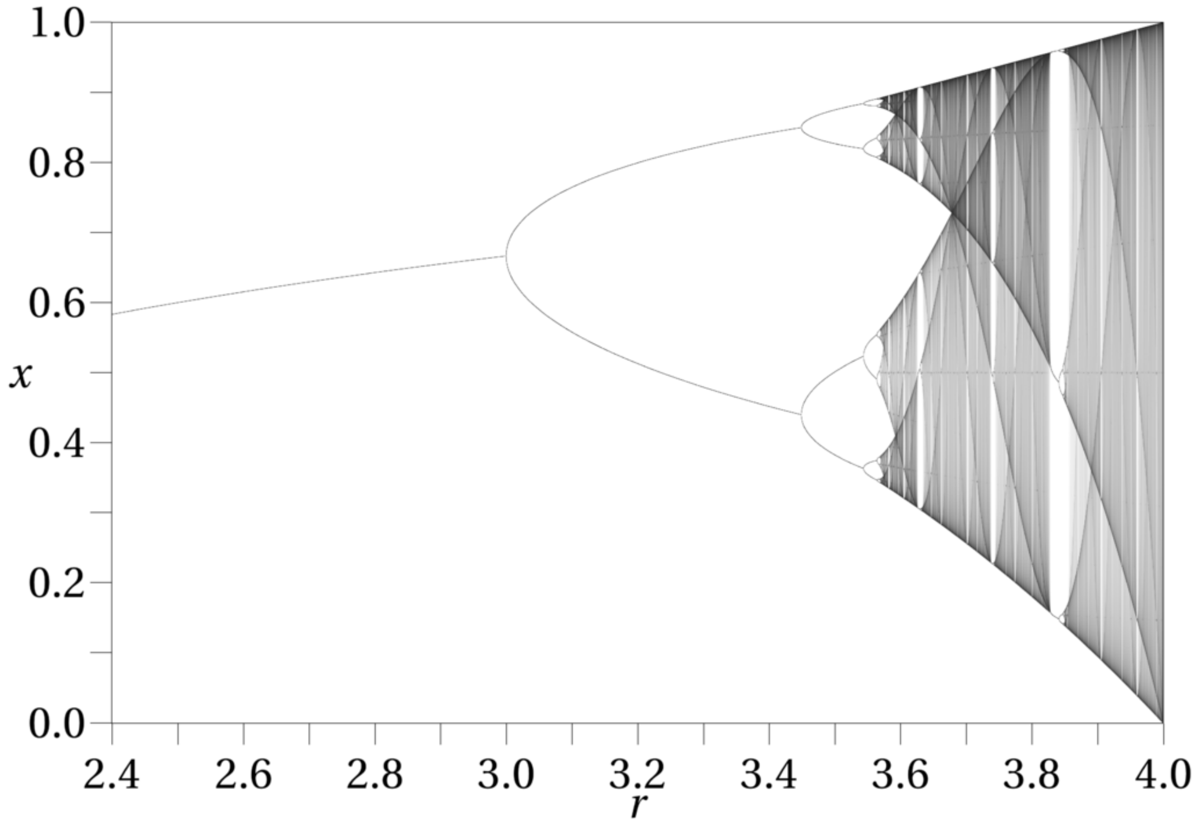
GEOMETRÍA COMPUTACIONAL

PRÁCTICA 1



DAVID SEIJAS PÉREZ

1. Introducción

En esta práctica vamos a estudiar el comportamiento de un sistema dinámico discreto $x_{n+1} = f(x_n)$ definido por la función logística

$$f(x) = rx(1 - x)$$

El objetivo es tratar de observar conjuntos atractores de este, con sus respectivos intervalos de error, para $x \in [0, 1]$ y $r \in (3, 4)$. Para conseguir esto, estudiaremos las órbitas que representan al sistema consiguiendo familiarizarnos con la programación en Python de estas.

2. Material Usado

Para el desarrollo de la práctica he utilizado el entorno *Spyder* de programación en Python, así como el archivo *plantilla1.py* del cual he utilizado, aunque ligeramente cambiadas, las funciones definidas, así como las formas para mostrar los resultados en gráficos. Además, he empleado las librerías *matplotlib.pyplot* y *random* para la representación de las órbitas y la elección "aleatoria" de mis datos r y x_0 respectivamente.

3. Metodología

Para la realización de esta primera práctica he utilizado los datos proporcionados en la plantilla: $N0 = 200$, $N = 50$ y $epsilon = 0.001$.

Para el primer apartado, usando la plantilla dada, he obtenido la órbita y el periodo para unos r y x0 obtenidos de forma pseudoaleatoria. Una vez hallada esta, he modificado la función *atrac* para que me devolviese, además de los atractores de la órbita, los puntos anteriores en cada curva que tiende a la asíntota definida por cada atractor. Con cada uno de estos puntos y los atractores, soy capaz de hallar la estimación del error de cada atractor, calculando la diferencia en valor absoluta de los dos puntos. Esto se basa en que un atractor en una iteración k, será como poco el valor de ese mismo atractor en la iteración anterior. Además, por arriba podemos estimar el error como esta diferencia también.

Para el segundo apartado, he usado la función *atrc* dada. He ido recorriendo distintos rs, empezando por $r = 3.4000$ con paso constante. He visto para qué r empezamos a tener 8 atractores y he hallado el punto medio entre ese r y el anterior (para la que teníamos 4 atractores). Ese punto, con la mitad del paso como error, lo he considerado como el extremo izquierdo del intervalo en el que tenemos 8 atractores. Para el extremos derecho realizo lo mismo, pero en el cambio de 8 a 16 atractores.

4. Resultados

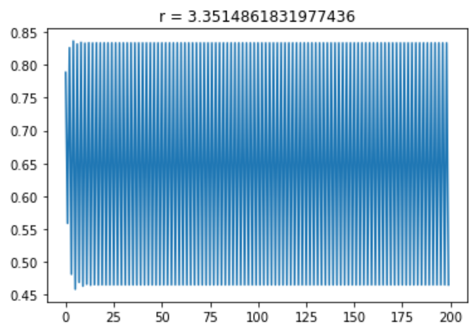
Para el apartado 1, según los distintos valores de r y x0 hemos obtenido distintas órbitas (gráficos insertados) y conjuntos de atractores:

a) Para $r = 3.527115490135466$ y $x0 = 0.0702934890852216$ he obtenido el siguiente conjunto de atractores:

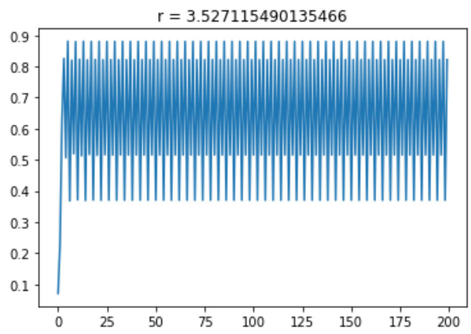
$(3700113035275 \pm 5) * 10^{-13}$ $(8809136539662 \pm 2) * 10^{-13}$
$(515662211175 \pm 2) * 10^{-12}$ $(8221809862006 \pm 8) * 10^{-13}$

b) Para $r = 3.3514861831977436$ y $x0 = 0.7887070542623948$:

$(8336914266006074 \pm 3) * 10^{-16}$ $(4646836659220392 \pm 7) * 10^{-16}$

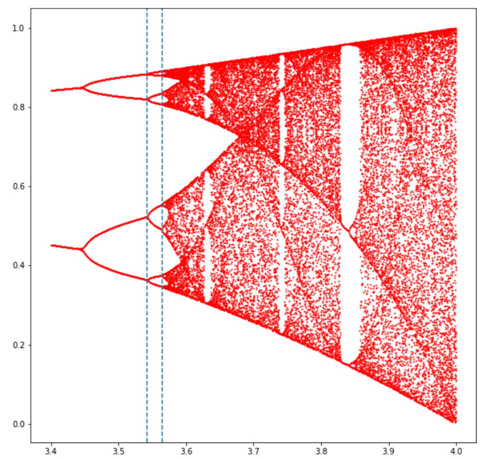


(a)



(b)

Para el apartado 2, ejecutando con $r \in (3.4, 4)$ con $paso = 0.0006$ (dando un tiempo de transición igual a 1000) obtengo que el intervalo en el que el conjuntos atractor de la órbita tiene 8 elementos es: $[3.5451 \pm 0.0003, 3.5641 \pm 0.0003]$



5. Conclusión

Con esta práctica he afianzado y acabado de entender muchos conceptos sobre sistemas dinámicos no lineales. Además, me ha parecido realmente sorprendente los cambios tan grandes que puede sufrir una órbita y su conjunto de atractores según el valor de r, pudiendo variar mucho los resultados sin un gran cambio en este parámetro, como hemos visto en los resultados. Además, cabe destacar la importancia de la estimación de los errores y su correcta notación. Cuando calculamos algo computacionalmente hay que tener en cuenta siempre los posibles errores que puede tener la computadora al realizar los cálculos.

6. Anexo: Código

```python
"""
Practica 1
DAVID SEIJAS
"""

import matplotlib.pyplot as plt
import random


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


def orbita(r, x0, f, N):
    orb = [0]*N
    orb[0] = x0
    for i in range(1, N):
        orb[i] = f(orb[i-1], r)
    return orb


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


def atrac1(f, r, x0, N0, N, epsilon=0.001):
    orb = orbita(r, x0, f, N0)
    ult = orb[N0-N-1:]
    per = periodo(ult, epsilon)
    atractor = [0]*per
    anterior = [0]*per
    for i in range(per):
        atractor[i] = ult[N-1-i]
        anterior[i] = ult[N-1-i-per]
    return (atractor, anterior)
#Los primeros per elems son los atractores y los ultimos per elems son
    los puntos anteriores a esos atractores en la curva que tiende a
   la asintota

def atrac2(f, r, x0, N0, N, epsilon=0.001):
    orb = orbita(r, x0, f, N0)
    ult = orb[N0-N-1:]
    per = periodo(ult, epsilon)
    atractor = [0]*per
    for i in range(per):
        atractor[i] = ult[N-1-i]
    return atractor
```

```


#Apartado 1
def conjAtractor():
    k = 2
    for i in range(k):
        r = random.uniform(3.0, 3.544)
        x0 = random.uniform(0.0, 1.0)
        atractor, anterior = atrac1(logistica, r, x0, N0, N)
        print("Conjunto atractor para r = " + str(r) + " y x0 = " + str(x0) + ":")
        for j in range(len(atractor)):
            print(str(atractor[j]) + " +/- " + str(abs(atractor[j] - anterior[j])))
        graficos1(r, x0, N0)


def graficos1(r, x0, N0):
    orb = orbita(r, x0, logistica, N0)
    plt.title('r = ' + str(r))
    plt.plot(orb)
    plt.show()


#Apartado 2
def atractor8elems(tam = 8):
    paso = 0.0006
    error = 0.0003
    time_trans = 1000

    rs = [3.4000 + paso*i for i in range(time_trans)] #(3.4000, 4.0000)
    x0 = random.uniform(0.0, 1.0)
    inicio = 0
    fin = 0
    atractores = [0]*time_trans

    for i in range(time_trans):
        atractor = atrac2(logistica, rs[i], x0, N0, N)
        if len(atractor) == tam and inicio == 0:
            inicio = (rs[i] + rs[i-1])/2
        if len(atractor) > tam and fin == 0:
            fin = (rs[i] + rs[i-1])/2
        atractores[i] = atractor + [float("nan")]*(N-len(atractor))

    print("Conjunto atractor con 8 elems en intervalo de r:")
    print("[" + str(inicio) + " +/- " + str(error) + ", " + str(fin) + " +/- " + str(error) + "]")
    graficos2(rs, atractores, inicio, fin)


def graficos2(r, atractores, int_min, int_max):
    plt.figure(figsize=(10,10))
    plt.plot(r, atractores, 'ro', markersize=1)
```

```
    plt.xlabel = "r"
    plt.ylabel = "atractores"
    plt.axvline(x=int_min, ls="--")
    plt.axvline(x=int_max, ls="--")
    plt.show()


N0 = 200
N = 50
conjAtractor()
atractor8elems()
```