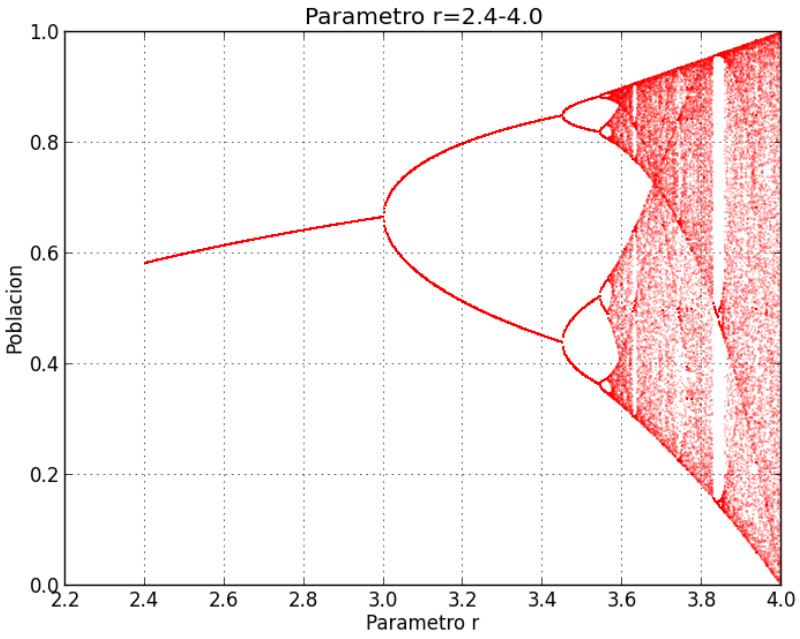
GEOMETRÍA COMPUTACIONAL

ATRACTOR LOGÍSTICO



LUCAS DE TORRE

Índice

1. Introducción

Nuestro objetivo es estudiar el sistema dinámico discreto $x_{n+1} = f(x_n)$ dominado por la función logística $f(x) = rx(1-x)$. Concretamente, por un lado buscamos dos conjuntos atractores diferentes (variando el valor de r y de x_0) y por el otro el valor de r tal que el conjunto atractor tenga 8 elementos.

2. Material

Para encontrar los dos conjuntos atractores diferentes, utilizamos $r \in (3,3,5)$ y $x_0 \in [0,1]$. Primero, generamos valores aleatorios de r y de x_0 válidos. Con estos valores realizamos un gran número de iteraciones de manera que $x_n = rx_{n-1}(1-x_{n-1}) \forall n \in Z^+$ para que la órbita se estabilice. Seguidamente, tomamos los últimos elementos calculados y comprobamos si hay ciclos. En concreto, realizamos 200 iteraciones, tomamos los últimos 25 elementos, y observamos si hay algún ciclo en esos 25 elementos. En caso de que sí, habremos encontrado el número de elementos de la órbita dada por la función logística con los valores de r y de x_0 iniciales, que se corresponde con el número de elementos que forman el ciclo de manera que no haya repetidos. Esos elementos que forman el ciclo son los elementos del conjunto atractor. Una vez calculado el conjunto atractor, lo refinamos para minimizar su error. Para ello, realizamos más iteraciones (10 iteraciones en nuestro caso) y escogemos el último conjunto calcuado como nuestro conjunto atractor V_0. En estas iteraciones, hacemos la diferencia en valor absoluto de los elementos de una iteración menos los de la anterior, cogemos el máximo en de estas dferencias en cada iteración y ordenamos estos máximos. En nuestro caso, tomamos el noveno mayor de los 10, lo que nos da el error de nuestro conjunto V_0 con cuantil de orden 0,9.

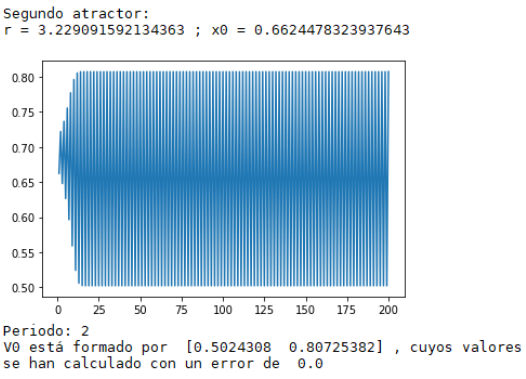


Figura 1: Órbita de 2 elementos

Por otro lado, para encontrar el intervalo I tal que si $r \in I$ entonces el conjunto V_0 tiene 8 elementos, comenzamos con un valor de r en el intervalo $(3,4)$. Procedemos mediante búsqueda binaria en dicho intervalo para encontar un valor de r que verifique que el conjunto V_0 tiene 8 elementos: utilizamos r como el punto medio del intervalo y si con dicha r encontramos un conjunto atractor con menos de 8 elementos, miramos en el intervalo $(r,4)$, mientras que si encontramos un conjunto atractor con más de 8 elementos, miramos en el intervalo $(3,r)$, y así sucesivamente hasta encontrar un valor de r tal que el conjunto atractor tiene 8 elementos. Una vez conseguida dicha r, utilizamos dos búsquedas binarias para encontrar los extremos del intervalo I buscado. Con una de ellas encontraremos el extremo inferior del intervalo y con la otra el extremo superior. Para hallar estos extremos, realizamos 20 iteraciones, por lo que, si el cálculo de los elementos del conjunto atractor fuera sin errores (aunque ya hemos visto antes

que esto no es así), estaríamos calculando cada extremo del intervalo con un error menor de 2^{-20} (ya que el intervalo inicial tiene longitud 1), por lo que el error en el cálculo del intervalo sería, a lo sumo, 2^{-19}.

3. Resultados

Buscando los dos conjuntos atractores se observa que, en función del valor de r (siempre contenido en el intervalo $(3,\ 3,5)$), encontramos conjuntos atractores con dos o elementos (si en dos casos distintos la r es distinta, aunque en ambos casos el conjunto atractor tuviera el mismo número de elementos, estos conjuntos serían diferentes). En concreto, cuando $r < 3,445$ (aproximadamente) el conjunto tiene dos elementos, y cuando $r > 3,445$ el conjunto atractor tiene cuatro elementos.

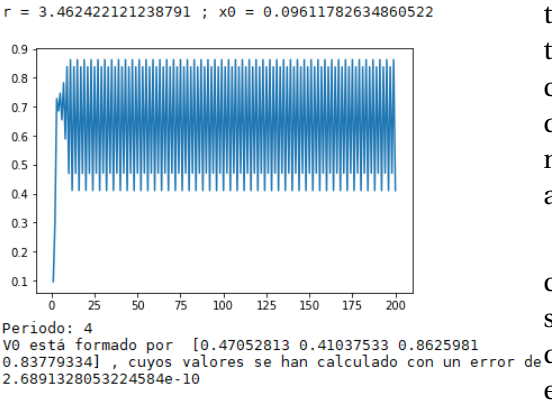


Figura 2: Órbita de 4 elementos

Estudiando el intervalo de r para el cual el conjunto atractor tiene 8 elementos, hemos observado que este es (3,546, 3,561), aunque hay que tener en cuenta el error con el que se calcula este intervalo.

4. Conclusiones

Nos hemos encontrado unos resultados muy interesantes ya que hemos visto que, en función del valor inicial de r, puede que no determinemos el conjunto atractor. Esto se produce alrededor del valor $r = 3,445$, ya que es cercano a la bifurcación que provoca que para valores menores de r el conjunto atractor tenga dos elementos y para valores mayores de r tenga 4 elementos. Una manera de reducir la cantidad de fallos es aumentando el número de iteraciones: siendo estas 2000 (en lugar de 200), el número de veces que no se encuentra el conjunto atractor es notablemente inferior.

Además, fijándonos en ambos estudios, vemos que el intervalo de r tal que el conjunto atractor tiene 2 elementos es mayor que el intervalo para el cual tiene 4 elementos y este es mayor que el intervalo para el que tiene 8. Aunque habría que comprobarlo, esto nos indica que el número de bifurcaciones crece enormemente cuando crece r, por lo que también crece enormemente el número de elementos que forman el conjunto atractor.

5. Anexo A: Código

```
"""
Created on Mon Feb  3 10:14:07 2020
@author: Jorge Sainero y Lucas de Torre
"""

import numpy as np
import random as rand
import matplotlib.pyplot as plt

PINTAR_GRAFICA = True
EPSILON = 1e-6

#Dado un punto x y el par metro r devuelve el valor de la funci n en x
def logistica(x, r):
    return r*x*(1-x)

#Aplica n veces al funci n f en x0
def fn(x0, f, n, r):
    x=x0
    for j in range(n):
        x=f(x, r)
    return x

#Aplica n veces la funci n f en x0 y guarda todos los valores para
    despu s calcular el error
def fError(x0, f, n, r):
    x=x0
    aux=np.zeros(n)
    for j in range(n):
        x=f(x, r)
        aux[j]=x
    return x,aux


#Esta funci n afina V0 y calcula su error realizando las diferencias entre
     sucesivas iteraciones
def afinar(v0,f, r, iteraciones, cuantil):
    error=np.zeros(iteraciones)
    aux=np.zeros((v0.size,iteraciones))
    v0.sort()
    v0ini=v0
    for i in range(v0.size):
        v0[i],aux[i]=fError(v0[i],f,iteraciones, r)
    for i in range(iteraciones):
        aux[:,i].sort()

    for i in range(iteraciones-1):
        for j in range(v0.size):
            aux[j][iteraciones-1-i]=np.abs(aux[j][iteraciones-1-i]-aux[j][
    iteraciones-2-i])

```

```
49      for j in range(v0.size):
50              aux[j][0]=np.abs(aux[j][0]-v0ini[j])
51
52      for i in range(iteraciones):
53          error[i]=max(aux[:,i])
54      error.sort()
55
56      return v0,error[max(round(iteraciones*cuantil-1),0)]
57
58 #Dado un punto x0 y el par metro r, calcula la  rbita  y devuelve su
       cardinal
59 def atractor(x, r):
60      k = 25
61      n = 200
62      orb =  np.zeros(n)
63      #Calculamos los n primeros t rminos de la sucesi n x_n+1=f(x_n)
64      for i in range(n):
65          orb[i] = fn(x,logistica,i, r)
66      if PINTAR_GRAFICA:
67          abscisas = np.linspace(1,n,200)
68          plt.plot(abscisas,orb[0:n])
69          plt.show()
70      #Tomamos los k  ltimos  t rminos de la sucesi n
71      ult=orb[-1*np.arange(k,0,-1)]
72      periodo = -1
73      #Calculamos el periodo de la  rbita  para saber cu ntos elementos
       tiene y cu les son
74      for i in range(1,k,1):
75          if abs(ult[k - 1] - ult[k - i - 1]) < EPSILON:
76              periodo = i
77              break
78      if periodo != -1:
79          #Si lo encontramos, tomamos esos elementos en v0
80          error = 0
81          v0 = orb[-1*np.arange(periodo,0,-1)]
82          #Afinamos V0 calculando su error con cuantil de orden 0,9
83          v0,error = afinar(v0,logistica, r, 10,0.9)
84          return periodo,v0,error
85      return periodo,0,0
86
87
88 def apartado1():
89      print("APARTADO UNO:\n")
90      global PINTAR_GRAFICA
91      PINTAR_GRAFICA = True
92      r1 = rand.uniform(3.0001 ,3.4999)
93      r2 = rand.uniform(3.0001 ,3.4999)
94      x01 = rand.random()
95      x02 = rand.random()
96
97      print("Primer atractor:")
98      print("r =",str(r1),"; x0 =",str(x01))
99      per1,v01,error1=atractor(x01, r1)
```

```
100     if per1 !=-1:
101         print("Periodo: "+str(per1))
102         print("V0 est  formado por ",v01,", cuyos valores se han calculado
        con un error de ",error1)
103     else:
104         print ("No se ha encontrado un periodo")
105     print("\n\n")
106 
107     print("Segundo atractor:")
108     print("r =",str(r2),"; x0 =",str(x02))
109     per2,v02,error2=atractor(x02, r2)
110     if per2 != -1:
111         print("Periodo: "+str(per2))
112         print("V0 est  formado por ",v02,", cuyos valores se han calculado
        con un error de ",error2)
113     else:
114         print ("No se ha encontrado un periodo")
115     print("\n\n\n")
116 
117 def apartado2():
118     print("APARTADO DOS:\n")
119     global PINTAR_GRAFICA
120     PINTAR_GRAFICA = False
121     x0 = rand.random()
122     a = 3
123     b = 4
124     cardinalV0 = -1
125     #Iteramos hasta encontar un r tal que V0 tenga 8 elementos
126     while cardinalV0 != 8:
127         r = (a+b)/2
128         cardinalV0,aux,aux = atractor(x0, r)
129         if cardinalV0 > 8 or cardinalV0 == -1:
130             b = r
131         elif cardinalV0 < 8:
132             a = r
133     #Buscamos los extremos del intervalo de los valores de r tal que el
       cardinal de la  rbita  es 8
134     a1 = a #Extremo izquierdo del intervalo izquierdo
135     b1 = r #Extremo derecho del intervalo izquierdo
136     a2 = r #Extremo izquierdo del intervalo derecho
137     b2 = b #Extremo derecho del intervalo derecho
138     #Iteramos 20 veces para reducir el error
139     for i in range(20):
140         r1 = (a1 + b1)/2
141         cardinalV01,aux,aux = atractor(x0,r1)
142         r2 = (a2 + b2)/2
143         cardinalV02,aux,aux = atractor(x0,r2)
144         if cardinalV01 < 8:
145             a1 = r1
146         else:
147             b1 = r1
148         if cardinalV02 > 8 or cardinalV02 == -1:
149             b2 = r2
```

```
150         else:
151             a2 = r2
152     #Devolvemos b1 y a2 para asegurarnos que en esos puntos el periodo vale
         8
153     print ("Para r perteneciente al intervalo ",[b1,a2]," v0 tiene 8
        elementos.")
154 
155 apartado1()
156 apartado2()
```