Memoria Practica 1 – Geometría computacional

Jorge del Valle Vázquez

**1 Introducción**

Comprender la función logística, ejemplo de sistema no lineal definido por:

Para ello buscamos obtener conjuntos atractores partiendo del cálculo de las órbitas y periodos, y analizamos los intervalos de error.

**2 Material usado**

Hacemos uso del entrono *Spyder* donde creamos el archivo ‘.py’ que incorpora librerías de tratado de listas y representación gráfica. Las variables de datos relevantes son , r, f, la función logística, N0, que indica el número de veces que se aplica f, N, los últimos datos con los que se realiza el cálculo del periodo, y , para determinar cuando dos valores son cercanos en el calculo del periodo.

**3 Resultados**

1. Tomamos = 0.5 y consideramos dos valores de r al azar para los que obtenemos dos conjuntos atractores distintos. Para analizarlo en su totalidad, dibujamos la órbita, obtenemos el conjunto atractor usando la función de la plantilla.

Posteriormente procedemos a evaluar el la estabilidad y bifurcación según las sugerencias propuestas, manteniendo x (estabilidad) o r(bifurcación) fijas y variando la otra en un intervalo centrado en tal variable.

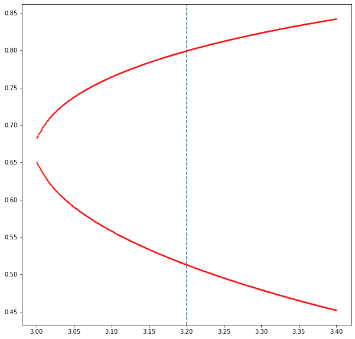
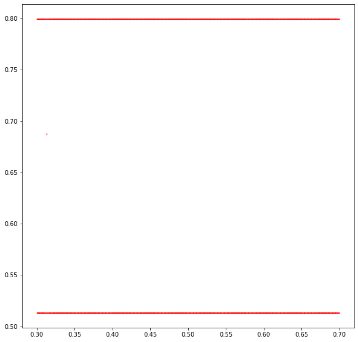
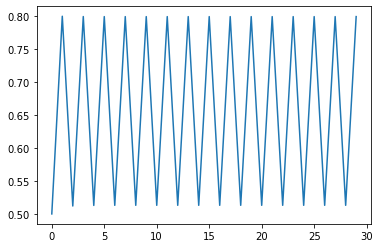
Para terminar, evaluamos el error. Seguimos una serie de iteraciones en las que calculamos los siguientes periodo valores y comparamos con el conjunto atractor inicial hasta encontrar error nulo o error invariantes en dos iteraciones consecutivas.

Así para el primer conjunto: (x=0.5) (r=3.20) (N=30) obtenemos:

Conjunto atractor: [0.51304451 0.79945549]

Atractor error: [0.51304451, 0.79945549] y error 0.0

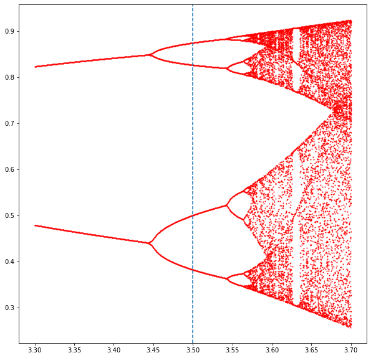
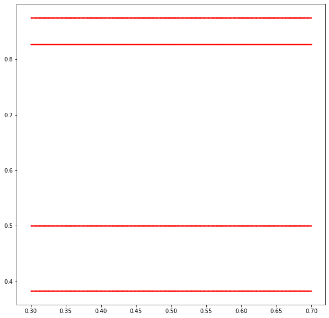
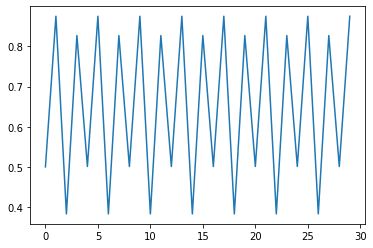
Por orden, orbita, estabilidad y bifurcación.



En el segundo conjunto: (x=0.5) (r=3.50) (N=30) tenemos:

Conjunto atractor: [0.38281968 0.50088421 0.82694071 0.87499726]

Atractor error: [0.38281968, 0.50088421, 0.82694071, 0.87499726] y error 0.0



1. En el segundo apartado consideramos valores de r ∈ (3.544, 4) variando desde el extremo izquierdo a paso 0.0005 y guardando todos los valores de r para los uqe el conjunto atractor tiene 8 elementos y luego procedemos como en el primero seleccionando un r aleatorio entre los posibles (hemos quitado los últimos decimales pues surgen por trabajar con números pequeños).

Para el valor de r 3.5625 se siguen las siguientes gráficas, conjunto atractor, errores y posibles valores de r.

Conjunto atractor: [0.34762602 0.37438873 0.49224989 0.55286622 0.80791149 0.8344152 0.88066839 0.89041102]

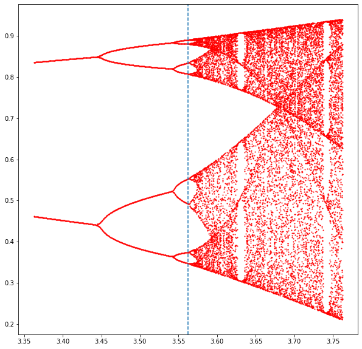
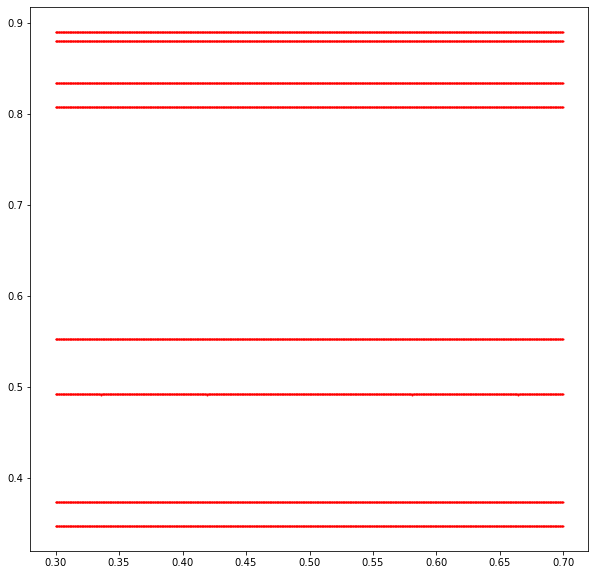
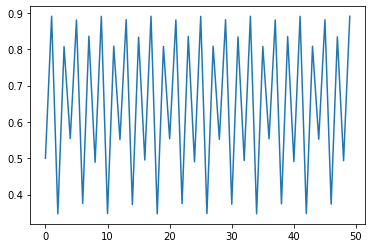
Atractor error: [0.34763092, 0.37437682, 0.49221807, 0.55285455, 0.8079168 ,

0.83440453, 0.88067279, 0.89040926]

Error: 3.181601936819156e-05

Valores de r:

[3.5440,3.5445, 3.5450, 3.5455,3.5460, 3.5465, 3.5470, 3.5475, 3.5480, 3.5485, 3.5490, 3.5495, 3.5500, 3.5505, 3.5510, 3.5515,3.5520, 3.5525, 3.5530, 3.5535, 3.5540, 3.5545, 3.5550, 3.5555, 3.5560, 3.5565, 3.5570, 3.5575, 3.5580, 3.5585, 3.5590, 3.5595, 3.5600, 3.5605, 3.5610, 3.5615, 3.5620, 3.5625, 3.5630, 3.5635, 3.5640, 3.5965, 3.6660, 3.8880, 3.8995, 3.9610]



**4 Conclusión**

La práctica cumple con el objetivo de comprender y visualizar los conceptos teóricos trabajados en clase y permiten al alumno manipular los datos a voluntad y “jugar” con el problema. Además, proporciona una primera toma de contacto agradable con el lenguaje *Python* .

**5 Anexo: Código**

# -\*- coding: utf-8 -\*-

"""

Práctica 1 : Jorge del Valle Vázquez

"""

**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)

**return**(x)

**def** orbita(x0,f,N,r):

orb = np.empty([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** 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

"""

Sugerencias

"""

**#Tomamos delta 0.0005 y analizamos el intervalo centrado en xo**

**def** estabilidad(x0, r, N0 = 200, N = 50, h = 0.2):

x0\_ = np.arange(x0 - h,x0 + h, 0.0005)

V0s = []

for x in x0\_:

V0s.append(atrac(logistica, x, r))

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

**for** i **in** range(len(x0\_)):

**for** j **in** V0s[i]:

plt.plot(x0\_[i], j,'ro', markersize=1)

#dibujamos para cada valor de x0 el conjunto de atractores

plt.show()

**#Tomamos DELTA 0.0005 y analizamos el intervalo centrado en r**

**def** bifurcaciones(x0,r, N0= 200, N = 50, h = 0.2):

rss = np.arange(r - h,r + h, 0.0005)

V0s = np.empty([N,len(rss)])\*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="--")

#dibujamos para cada valor de r el conjunto de atractores

plt.show()

**def** error(f, r, V0, N0 = 200, N = 50, epsilon = 0.001):

errores = []

V0\_ult=V0[-1]

**'''**

**a partir del ultimo de los elementos del conjunto atractor calculamos los siguientes valores ,**

**cuantos? el periodo**

**calculamos componente a componente el error entre el V0 de entrada y el conjunto atractor**

**obtenido de aplicar f "periodo" veces a tal conjunto**

**nos quedamos con el maximo de entre estos errores**

**si tal maximo es 0 eso indica que nuestro calculo inicial carecia de error**

**si tal maximo permanece invariante por dos iteraciones obtenemos así el error del calculo inicial**

**'''**

**for** j **in** range(N):

orb = orbita(V0\_ult, f, len(V0), r)

V0\_ult = orb[-1]

orb\_ult\_ord = np.sort(orb[-len(V0):])

errores.append(max([abs(V0[i]- orb\_ult\_ord[i]) for i in range(len(V0))]))

**if** errores[-1] == 0 or (len(errores) >= 2 and errores[-1] == errores[-2]):

**break**

**return** ( orb\_ult\_ord, errores[-1] )

"""

i

"""

print("**Primer conjunto: (x=0.5) (r=3.** **20) (N=30)**")

plt.plot(orbita(0.5, logistica, 30, 3. 20))

V0 = atrac(logistica, 0.5, 3. 20)

estabilidad(0.5, 3. 20)

bifurcaciones(0.5, 3. 20)

print(V0, error(logistica,3. 20, V0))

print**("Segundo conjunto: (x=0.5) (r=3.50) (N=30)"**)

plt.plot(orbita(0.5, logistica, 30, 3.50))

V0 = atrac(logistica, 0.5, 3.50)

estabilidad(0.5, 3.50)

bifurcaciones( 0.5,3.50)

print(V0, error(logistica,3.50, V0))

**"""**

**ii**

**"""**

print**("Valores de r ∈ (3.544, 4) para los cuales el conjunto atractor tiene 8 elementos**")

rss = np.arange(3.544,4, 0.0005)

r\_ = []

for i in range(len(rss)):

V0 = atrac(logistica, 0.5, rss[i])

# comprobamos que el cjto atractor tenga 8 elementos

if len(V0)==8 :

r\_.append(rss[i])

#seleccionamos uno de los r para los cuales hay 8 elem en cjto atractor

r8= np.random.choice(r\_)

plt.plot(orbita(0.5, logistica, 100,r8))

V0 = atrac(logistica, 0.5, r8)

estabilidad(0.5, r8)

bifurcaciones( 0.5,r8)

print(r\_)

print(V0, error(logistica,r8, V0))