Íñigo Iriondo Delgado
Geometría Computacional

Memoria 1. ATRACTOR LOGÍSTICO

(1) Introducción

El objetivo de esta práctica es estudiar los conjuntos atractores en sistemas dinámicos discretos con la función logística f(x) = rx(1 - x) con x0 = 0.5. Primero encontraremos dos conjuntos atractores distintos para r $\in$ (3, 3.544) y después estimaremos un valor de r $\in$ (3.544, 4) con conjunto atractor de solo 8 elementos.

(2) Material usado

El código de Python tiene 4 funciones:

- funcion_logistica(r, x): devuelve r * x * (1 - x)
- atractores(r, x0 (=0.5), iter_inic (=1000), iter_fin (=1000)): calcula el conjunto atractor aplicando la función logística iterativamente (primero iter_inic veces para que el sistema se estabilice y luego hace iter_fin iteraciones más para obtener los valores atractores)
- error(r, x0 (=0.5), iter_inic (=1000), iter_fin (=1000), delta_r =(0.001), delta_x (=0.001)): calcula los errores asociados con variaciones en r (delta_r) y x0 (delta_x) (primero toma las diferencias mínimas entre los conjuntos atractores para distinguir los distintos puntos y después devuelve la máxima de estas)
- r_con_atractor_de_n_elementos(r_min (=3.544), r_max (=4), n_elem (=8), tol=0.0001): encuentra un valor de r en (r_min, r_max) que nos da un conjunto atractor de n_elem elementos, haciendo búsqueda binaria

(3) Resultados

Si ejecutamos el programa obtenemos:
Apartado i
r: 3.1813, conjunto atractor: [0.5203, 0.794]
Intervalo de error: ±3e-05 en r, ±0.0 en x
r: 3.3627, conjunto atractor: [0.4617, 0.8357]
Intervalo de error: ±2e-05 en r, ±0.0 en x
Apartado ii
r: 3.5511, conjunto atractor: [0.354, 0.3709, 0.5044, 0.542, 0.8121, 0.8286, 0.8815, 0.8877]
Intervalo de error: ±1e-05 en r, ±0.0 en x

(4) Conclusión

Cuando elegimos un valor de r entre 3 y 3.544, obtenemos conjuntos atractores con menos elementos, lo cual indica un comportamiento periódico. Sin embargo, para r mayor, el sistema tiene más atractores ya que es más caótico y sensible a las condiciones iniciales. Obtenemos errores muy pequeños en todos los casos, lo cual muestra la precisión del algoritmo.

(5) Anexo con el script

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

def atractores(r, x0, iter_inic, iter_fin):
    x = x0
    for _ in range(iter_inic):
        x = funcion_logistica(r, x)
    atract = set()
    for _ in range(iter_fin):
        x = funcion_logistica(r, x)
        atract.add(x)
    return atract

def error(r, x0, iter_inic, iter_fin, delta_r, delta_x):
    atract1 = atractores(r, x0, iter_inic, iter_fin)
    atract2 = atractores(r + delta_r, x0, iter_inic, iter_fin)
    atract3 = atractores(r - delta_r, x0, iter_inic, iter_fin)
    atract4 = atractores(r, x0 + delta_x, iter_inic, iter_fin)
    atract5 = atractores(r, x0 - delta_x, iter_inic, iter_fin)
    error_r_pos = min(min(abs(x - y) for y in atract2) for x in atract1)
    error_r_neg = min(min(abs(x - y) for y in atract3) for x in atract1)
    error_x_pos = min(min(abs(x - y) for y in atract4) for x in atract1)
    error_x_neg = min(min(abs(x - y) for y in atract5) for x in atract1)
    return (max(error_r_pos, error_r_neg), max(error_x_pos, error_x_neg))

def r_con_atractor_de_n_elementos(r_min, r_max, n_elem, tol=0.0001):
    while r_max - r_min > tol:
        r_med = (r_min + r_max) / 2
        atract_med = atractores(r_med, x0, 10000, 10000)
        num_elem = len(atract_med)
        if num_elem < n_elem:
            r_min = r_med
        elif num_elem > n_elem:
```

```
            r_max = r_med
        else:
            return r_med, atract_med

# Apartado i
x0 = 0.5
r1, r2 = 3 + (0.544 / 3), 3 + (0.544 * 2 / 3)
atract1 = atractores(r1, x0, 10000, 10000)
atract2 = atractores(r2, x0, 10000, 10000)
error_1 = error(r1, x0, 10000, 10000, 0.0001, 0.0001)
error_2 = error(r2, x0, 10000, 10000, 0.0001, 0.0001)
print("Apartado i")
print(f"r: {round(r1, 4)}, conjunto atractor: {sorted({round(x, 4) for x in atract1})}")
print(f"Intervalo de error: ±{round(error_1[0], 5)} en r, ±{round(error_1[1], 5)} en x")
print(f"r: {round(r2, 4)}, conjunto atractor: {sorted({round(x, 4) for x in atract2})}")
print(f"Intervalo de error: ±{round(error_2[0], 5)} en r, ±{round(error_2[1], 5)} en x")

# Apartado ii
r3, atract3 = r_con_atractor_de_n_elementos(3.544, 4, 8)
error_3 = error(r3, x0, 10000, 10000, 0.0001, 0.0001)
print("Apartado ii")
print(f"r: {round(r3, 4)}, conjunto atractor: {sorted({round(x, 4) for x in atract3})}")
print(f"Intervalo de error: ±{round(error_3[0], 5)} en r, ±{round(error_3[1], 5)} en x")
```