TRABAJO AMPLIACIÓN DE MATEMÁTICAS

FRACTAL DE FEIGENBAUM

RAÚL PÉRULA MARTÍNEZ

I.T.INFORMÁTICA DE SISTEMAS

UNIVERSIDAD DE CÓRDOBA

FRACTAL DE FEIGENBAUM

Introducción

Primero trataré de explicar quien fue Feigenbaum, comentaré su aportación de la constante de Feigenbaum y después me centraré en comentar de que trata su fractal y como lo he realizado.

Mitchell Jay Feigenbaum (1944 -):

Nacido el 19 de diciembre de 1944 en Filadelfia, USA. Entró en el MIT con la intención de investigar en la rama de ingeniería eléctrica para realizar su doctorado pero después empezó a estudiar la relatividad general leyendo el curso de física teórica de Lev Landau y Evgenii Lifshitz.

Ha realizado estudios principalmente de la Teoría de la Relatividad General, Espacios de Banach, Análisis computacional, **Teoría del Caos**, Ecuación Logística, **Geometría Fractal**…

Feigenbaum es considerado tanto físico como matemático. Y aun hoy sigue brindándonos aportaciones en la rama del caos siendo uno de los pioneros creadores y estudiantes de esta teoría.

La constante de Feigenbaum:

Si llamamos r_k al valor de r para que el periodo 2^k llegue a ser inestable y determinamos el límite, si k tiende a infinito, de $(r_{k+1} - r_k) / (r_{k+2} - r_{k+1})$ obtenemos un valor de 4,66920160...

Sorprendentemente, ese valor es el mismo para todas las funciones de este tipo las cuales tienden al caos. De hecho, el valor es una de las constantes universales, y aproximadamente es igual a **4.669.**

Fractal de Feigenbaum:

El fractal de Feigenbaum es una bifurcación fractal. Está producido por la ecuación

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

donde r está variando sobre una dimensión. Al principio parece tender a tener un orden pero poco a poco va entrando en un entorno caótico.

Para su realización podríamos iterar r en el intervalo [0,4] aunque en el dibujo nos hemos centrado en la parte donde se ve mejor la parte caótica, el intervalo [2.5,4]. Hemos utilizado el método del punto fijo en el cual cada x toma el valor de la función antes descrita en un punto anterior en el intervalo [0,1], esto es, por ejemplo, para x_0 tomaremos el valor inicial, que es 0, y para el siguiente valor, x_1, tomaremos el valor de la función evaluándola en el punto anterior.

$$x_1 = r*x_0*(1-x_0)$$

Y así sucesivamente para x_2, x_3, … hasta llegar a hacer todas las iteraciones. Para que el dibujo tenga una consistencia y pueda contemplarse bien y claramente, harían falta como mínimo 150 iteraciones.

El código en lenguaje C sería:

```
#include<stdio.h>
#include<math.h>

//Prototipos de funciones
double funcion(double, double);

int
main(void)
{
  int i, j = 0;
  double b1 = 1., b2 = 4.0001;     //Fin de intervalos
  double r = 2.5;                  //Valor inicial de r
  double x = 0.1;                  //Valor inicial de x
  double aux;                      //Variable auxiliar
  double h = 0.001;                //Valor de h, representa el tamaño del intervalo

  //Se itera desde 2.5 hasta el fin de intervalo
  while(r <= b2){
      //Efectuamos 1050 iteraciones de las cuales nos quedamos las 50 ultimas
     for(i=0;i<1050;i++){
          //Almacenamos en la variable auxiliar el valor de la función
          aux = funcion(x,r);
          //Almacenamos el valor auxiliar en la variable x para que posteriormente se
produzca la iteracion de punto fijo
          x = aux;
           if(i > 999){
               printf("%lf %lf\n", r, x); //Imprimimos los valores de los puntos
            }
      }
      r += h;
      x = 0.1;
   }

  return 0;
}

//Función
double funcion(double x, double r)
{
  return(r*x*(1-x));
}
```

Este código nos mostraría por pantalla los puntos a representar, aunque no es el más efectivo ya que muestra los puntos a partir de la iteración 1000 hasta la iteración 1050, mostrando para cada r todos los puntos, en iteración de punto fijo, de x, así repetiría al principio, sobretodo, la misma representación.

Para representarlo he utilizado el programa de representación gráfica GNU-PLOT, para ello he tenido que crear un archivo *.plt para que el programa cargue los puntos y pueda representarlos.

Las líneas de código del programa *.plt son:

```
set pointsize 0.01
plot [2.5:4]"datos.dat" title ""
```

Por último, mostraré la representación que he obtenido: