Reporte de actividades 1

Ciro Fabián Bermúdez Márquez

24 de marzo de 2021

1. ¿Qué tengo que hacer?

1. El sistema oscila cuando los coeficientes valen 0.7, la función no lineal la vamos a dejar fija para generar 2 scrolls, esta última no se va a optimizar.
2. Verificar con Grünwald-Letnikov que el sistema genere caos cuando los exponentes fraccionarios de las derivadas son 0.9 dejando los coeficientes en 0.7 y fija la función PWL en 2 scrolls.
3. Una vez verificado lo anterior iniciamos con la programación de la heuristica para maximizar la dimensión Kaplan–Yorke. Elegimos el rango de los coeficientes a, b, c máximo hasta dos y elegimos el paso, por ejemplo de 0.1 hasta 2, serian 10 valores en cada coeficiente, y las derivadas fraccionarias deben poder tomar valores de 0.3 a 0.9 en pasos de 0.1.
4. Aprender a utilizar TISEAN 3.0.1, a este programa se le agregan parámetros y los datos temporales y arroja los valores de los exponentes de Lyapunov y la dimensión Kaplan–Yorke.
5. Primero simular con valores fraccionarios de 0.9 pero en el proceso de optimización al generar la población o las partículas para las derivadas de orden fraccionario la suma debe de ser 2.9 o mayor.

2. Definición de Grünwald-Letnikov

Comenzamos considerando que para el caso de orden entero la n-ésima derivada para una función f con $n \in \mathbb{N}$ y $j > n$ esta dada por:

$$f^{(n)}(t) = \frac{d^n f}{dt^n} = \lim_{h \to 0} \frac{1}{h^n} \sum_{j=0}^{n} (-1)^j \binom{n}{j} f(t - jh) \tag{1}$$

donde $\binom{n}{j}$ representa el coeficiente binomial dado por la expresión:

$$\binom{n}{j} = \frac{n!}{j!(n-j)!} \tag{2}$$

considerando valores negativos de n tenemos:

$$\binom{-n}{j} = \frac{-n(-n-1)(-n-2)\cdots(-n-j+1)}{j!} = (-1)^j \begin{bmatrix} n \\ j \end{bmatrix} \tag{3}$$

donde $\begin{bmatrix} n \\ j \end{bmatrix}$ esta definido como:

$$\begin{bmatrix} n \\ j \end{bmatrix} = \frac{2(n+1)\cdots(n+j-1)}{j!} \tag{4}$$

2.1. Definición de derivada de Grünwald-Letnikov

Generalizando la ecuación (1) podemos escribir la definición de derivada de orden fraccionario de orden α, ($\alpha \in \mathbb{R}$) como:

$$D_t^{\alpha} f(t) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} f(t - jh) \tag{5}$$

donde para calcular el coeficiente binomial podemos utilizar la relación entre la función Gamma de Euler y el factorial definido como:

$$\binom{\alpha}{j} = \frac{\alpha!}{j!(\alpha-j)!} = \frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)} \tag{6}$$

donde la función Gamma de Euler con $r > 0$ esta definida como:

$$\Gamma(r) = \int_0^\infty t^{r-1}e^{-t}dt \tag{7}$$

2.2. Definición de integral de Grünwald-Letnikov

Utilizando la ecuación (5) se puede definir un operador de tipo integral para la función f sobre el dominio temporal (a,t) considerando $n = \frac{t-a}{h}$ donde $a \in \mathbb{R}$ como:

$${}_aD_f^\alpha = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\left[\frac{t-a}{h}\right]} (-1)^j \binom{\alpha}{j} f(t-jh) \tag{8}$$

2.3. Método numérico para la definición de GL

Utilizando como base la ecuación (5) esta se puede discretizar para los puntos kh, $(k = 1, 2, \ldots)$ de la siguiente manera:

$$({}_{\frac{L_m}{h}})D_{t_k}^\alpha f(t) \approx \frac{1}{h^\alpha} \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} f(t_{k-j}) \tag{9}$$

donde L_m es el tamaño de memoria (memory length), $t_k = kh$, h es el paso de tiempo del cálculo y $(-1)^j\binom{\alpha}{j}$ son coeficientes binomiales $C_j^{(\alpha)}$ $(j = 0, 1, \ldots)$. Para su cálculo utilizamos la siguiente expresión:

$$C_0^{(\alpha)} = 1, \qquad C_j^{(\alpha)} = \left(1 - \frac{1+\alpha}{j}\right) C_{j-1}^{(\alpha)} \tag{10}$$

Entonces, la solución numérica general de la ecuación diferencial fraccionaria:

$${}_aD_t^\alpha y(t) = f(y(t), t) \tag{11}$$

puede expresarse como:

$$y(t_k) = f(y(t_{k-1}), t_{k-1})h^\alpha - \sum_{j=1}^{k} C_j^{(\alpha)} y(t_{k-j}) \tag{12}$$

Para el término de la memoria expresada por la sumatoria, el principio de memoria corta puede utilizarse. Para esto el índice superior de la sumatoria en la ecuación (12) se cambiará por ν con las siguientes consideraciones: se utiliza $\nu = k$ para $k < \left(\frac{L_m}{h}\right)$ y $\nu = \left(\frac{L_m}{h}\right)$ para $k \geq (\frac{L_m}{h})$. Sin el principio de memoria corta se utiliza $\nu = k$ para toda k. En la sección 8 se muestran los Códigos 1 y 2 para hacer la simulación de los osciladores caóticos utilizando el método numérico de GL y la función de saturación SNLF.

3. Caos

El caos se refiere a un tipo de comportamiento dinámico complejo que posee algunas características muy especiales, tales como extrema sensibilidad a pequeñas variaciones de la condición inicial, trayectorias encerradas en el espacio de fase con al menos un exponente de Lyapunov positivo y un espectro en potencia continuo.

En otras palabras, el caos es un comportamiento impredecible de un sistema determinista que presenta una sensibilidad muy grande a las condiciones iniciales.

4. Exponente de Lyapunov

La principales formas de caracterización de los sistemas caóticos son la dimension fractal, la entropía de Kolmogorov-Sinai y el espectro de Lyapunov. Entre ellos, el exponente de Lyapunov proporciona una forma de determinar si el comportamiento de un sistema es caótico. Los exponentes de Lyapunov dan la descripción de la presencia de un flujo determinista no periódico, por lo tanto, son una medida asintótica que caracteriza la tasa media de crecimiento (o disminución) de pequeñas perturbaciones a las soluciones de un sistema dinámico.

Los exponentes de Lyapunov proporcionan medidas cuantitativas de la sensibilidad de respuesta de un sistema dinámico a pequeños cambios en las condiciones iniciales. El número de exponentes de Lyapunov es igual al numero de variables de estado, y si al menos uno es positivo, esto es un indicador de caos. Un valor alto del exponente positivo de Lyapunov indica un gran incremento en el grado de impredecibilidad del sistema, por lo tanto, el sistema presenta un comportamiento dinámico más complejo.

5. Oscilador caótico de múltiples enrollamientos basado en series de funciones saturadas

Este oscilador caótico también conocido como SNLF (Funciones No Lineales Saturadas) se describe por las tres ecuaciones diferenciales como se muestra en (13), donde a, b, c y d_1 son constantes positivas que pueden tener valores en el intervalo $[0, 1]$. El sistema dinámico esta controlado por una aproximación PWL (Piecewise-linear) que describe la serie de funciones saturadas f, que se obtiene de la siguiente manera: Sea f_0 la función saturada descrita por (14) donde $\frac{1}{m}$ es la pendiente del segmento del medio y $m > 0$; el radial superior $\{f_0(x_1, m) = 1 \mid x_1 > m\}$ y el radial inferior $\{f_0(x_1, m) = -1 \mid x_1 < -m\}$ se llaman *mesetas saturadas*, y el segmento $\{f_0(x_1, m) = \frac{x_1}{m} \mid |x_1| < -m\}$ entre dos mesetas saturadas se llama *pendiente saturada*.

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= x_3 \\ \dot{x_3} &= -ax_1 - bx_2 - cx_3 + d_1 f(x_1; m) \end{aligned} \tag{13}$$

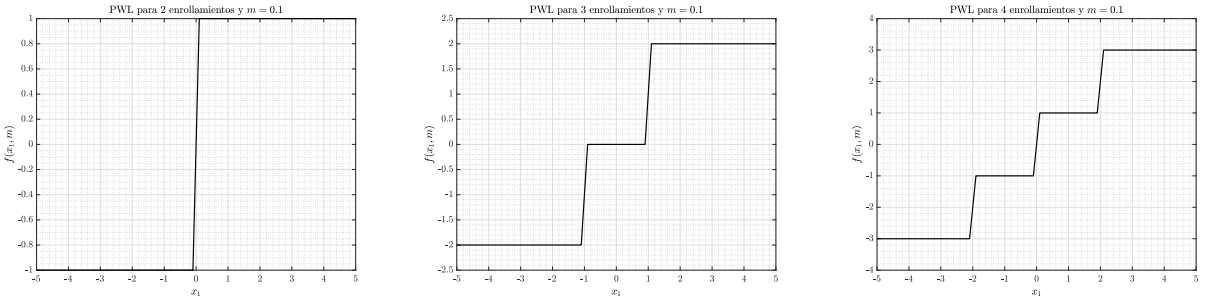
$$f_0(x_1, m) = \begin{cases} 1 & \text{si} \quad x_1 > m \\ \frac{x_1}{m} & \text{si} \quad |x_1| \leq m \\ -1 & \text{si} \quad x_1 < -m \end{cases} \tag{14}$$

$$f_h(x_1, m, h) = \begin{cases} 2 & \text{si} \quad x_1 > h + m \\ \frac{x_1 - h}{m} & \text{si} \quad |x_1 - h| \leq m \\ 0 & \text{si} \quad x_1 < h - m \end{cases} \tag{15}$$

$$f_{-h}(x_1, m, -h) = \begin{cases} 0 & \text{si} \quad x_1 > h + m \\ \frac{x_1 - h}{m} & \text{si} \quad |x_1 - h| \leq m \\ -2 & \text{si} \quad x_1 < h - m \end{cases} \tag{16}$$

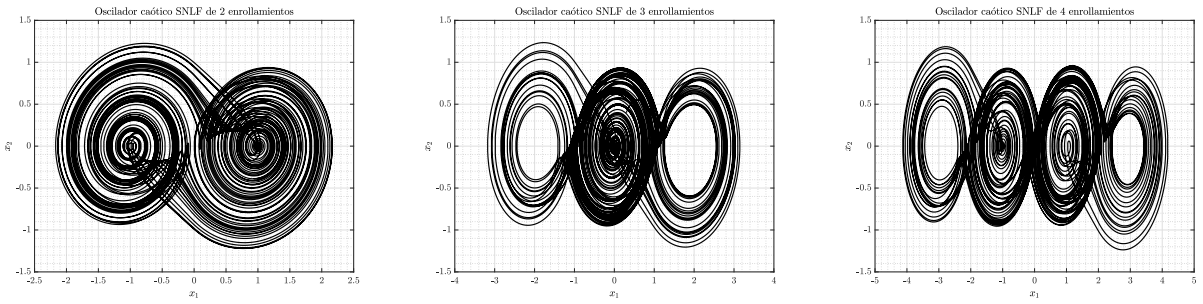
La serie de funciones saturadas para generar s enrollamientos puede definirse como se muestra en (17), para $s > 2$.

$$f(x, m) = \sum_{i=0}^{s-2} f_{2i-s+2}(x, m, 2i - s + 2) \tag{17}$$



(a) (b) (c)

Figura 1: Diferentes PWL con $m = 0.1$, (a) 2 enrollamientos, (b) 3 enrollamientos, (c) 4 enrollamientos..



(a) (b) (c)

Figura 2: Osciladores caóticos con $a = b = c = 0.7$ y $\alpha = 1$, (a) 2 enrollamientos, (b) 3 enrollamientos, (c) 4 enrollamientos.

6. Optimización

Consisten en determinar el valor máximo o mínimo de una función con respecto a una variable o varias variables, encontrando la mejor solución de todas las soluciones posibles. En general el problema se puede definir como:

$$\begin{gathered}\text{Minimizar: } f(\mathbf{x}) \\ \mathbf{x} \in \Psi \subseteq \mathbb{R}^n\end{gathered}$$

$$\text{Sujeto a:} \quad g(\mathbf{x}) \leq 0 \quad \text{y} \quad h(\mathbf{x}) = 0$$

donde $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ es un vector que contiene todas las variables de la función, $f(\mathbf{x})$ es llamada función objetivo la cual mide la calidad de la solución y Ψ es el conjunto de puntos que cumplen las restricciones del problema, conocida como zona factible. De mismo modo se pueden agregar restricciones en la forma de desigualdad como $g(\mathbf{x})$ o igualdad como $y(\mathbf{x})$.

Para problemas no lineales, multimodales o de más de tres variables se justifica utilizar heurísticas para la optimización.

7. Evolución diferencial (ED)

La evolución diferencial usa números reales en su presentación de las variables, en general realiza un mejor trabajo que el algoritmo genético y se pude usar una condición automática de paro.

El algoritmo requiere las siguientes variables para su funcionamiento:

- Tamaño de población μ
- Numero de generaciones g
- Valor del constante de recombinación R
- Valor del constante de diferencias F
- Valor del umbral s.

Para un problema de n variables el tamaño de población debe ser igual a $10n$. Es importante resaltar que si R, F las cuales se encuentran en el rango de $[0, 1]$ son iguales a 1 se está usando una búsqueda aleatoria. El parámetro s está definido en el espacio de la función. Si se tiene un umbral u significa $0.1u$ en las variables. Si se tiene una precisión 1×10^{-5}, equivale a 1×10^{-6} en las variables.

Las especificaciones a tener en cuenta para esta heurística son $F = 0.2$, $R = 0.4$, $\mu = 100$, $g = 30, 50, 100, 150$, espacio de búsqueda x_i en el rango **por definir**.

Vamos a utilizar evolución diferencial y PSO para la optimización

8. Códigos

```
function  R = sat_fun_k(x,m,s,k)
%   x vector de entrada
%   m
%   s numero de enrollamientos

    [r,c] = size(x);
     R = zeros(r,c);
     for i = 0:s-2
         sub = 2*i-s+2;

             if sub == 0
                 f = @fo;
             elseif sub >0
                 f = @fh;
             else
                 f = @f_h;
             end

          for j = 1:r*c
             R(j) = R(j) + f(x(j),m,sub,k);
          end

     end

end

function R = fo(x,m,h,k)
    if x < -m
        R = -1*k;
    elseif -m <= x && x <= m
        R = (x/m)*k;
    elseif x > m
        R = 1*k;
    end
end

function R = fh(x,m,h,k)
    if x > h + m
        R = 2*k;
    elseif -m+h <= x && x <= m+h
        R = ((x-h)/m +1)*k;
    elseif x < h-m
        R = 0;
    end
end

function R = f_h(x,m,h,k)
    if x > h + m
        R = 0;
    elseif -m+h <= x && x <= m+h
        R = ((x-h)/m -1)*k;
    elseif x < h-m
        R = -2*k;
    end
end
```

Código 1: Función de series saturadas.

```
%%
clear all;
close all;
clc;
tic                              %Medir tiempo de ejecucion
%% Variables
ini_cond = [0.3 0.4 0.5]';          % Condiciones iniciales
tn = 1000;                         % Tiempo
h = 0.01;                        % Paso de integracion
k = round(tn/h);                  % Numero de iteraciones
alpha = 0.97;                     % Orden fraccionario
lm = (tn/10);                     % Tamanio de memoria
M =  lm/h;                        % Ventana de memoria
%% Inicializacion de vectores
x = zeros(k + 1, 1);
y = zeros(k + 1, 1);
z = zeros(k + 1, 1);
Cj = zeros(k + 1, 1);
A = zeros(3,1);
%% Primer elemento de los vectores
x(1) = ini_cond(1);
y(1) = ini_cond(2);
z(1) = ini_cond(3);
Cj(1) = 1;
%% Calculo coeficientes Cj
for j=1:M                                  % Modificar k o M sin y con memoria corta
    Cj(j+1) = ( 1 - (alpha+1)/(j) ) * Cj(j);
end
for i=1:k
```

```
30      A(:) = 0;
31      %% Principio de memoria corta
32      if i < M
33          v = i;
34      else
35          v = M;
36      end
37      %% Sin memoria corta
38  %     v = i;
39      for j=1:v
40          A(1) = A(1) + Cj(j+1)*x(i+1-j);
41          A(2) = A(2) + Cj(j+1)*y(i+1-j);
42          A(3) = A(3) + Cj(j+1)*z(i+1-j);
43      end
44      x(i+1) = x_state(x(i),y(i),z(i))*h^(alpha) - A(1);
45      y(i+1) = y_state(x(i),y(i),z(i))*h^(alpha) - A(2);
46      z(i+1) = z_state(x(i),y(i),z(i))*h^(alpha) - A(3);
47  end
48
49  % plot3(x,y,z);
50  % xlabel('x(t)');
51  % ylabel('y(t)');
52  % zlabel('z(t)');
53  plot(x,y);
54  grid on;
55  toc             % Mostrar tiempo de ejecucion
56  %% Funciones oscilador caotico
57  function R = x_state(x,y,z)
58      R = y;
59  end
60
61  function R = y_state(x,y,z)
62      R = z;
63  end
64
65  function R = z_state(x,y,z)
66      a = 0.7;
67      b = 0.7;
68      c = 0.7;
69      d = 0.7;
70      m = 0.1;
71      R = -a*x -b*y -c*z+ d*sat_fun_k(x,m,2,1);
72  end
```

Código 2: Método numérico de GL.

Referencias

[1] I. Petráš, *Fractional-Order Nonlinear Systems*. Springer Berlin Heidelberg, 2011.