BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA
MAESTRÍA EN INGENIERÍA ELECTRÓNICA, OPCIÓN INSTRUMENTACIÓN ELECTRÓNICA

Tesis para obtener el grado de
MAESTRO EN INGENIERÍA ELECTRÓNICA, OPCIÓN INSTRUMENTACIÓN ELECTRÓNICA

"INSTRUMENTACIÓN ELECTRÓNICA DE CAOS FRACCIONAL PARA APLICACIONES DE CIFRADO DE DATOS EN IoT"

Presenta:
Ing. Samuel Giovanni Hernández Orbe*

Asesores:
Dr. Jesús Manuel Muñoz Pacheco
Dr. Germán Ardul Muñoz Hernández
Dr. Christos Volos

*Becario CONACYT

Puebla, Pue., Diciembre 2019

Dedicatoria

A veces los miedos nos hacen caminar más lento, es hasta que los afrontas que te das cuenta que el querer es poder y esto es lo que te permite avanzar.

Con dedicatoria especial para mi padres, amigos y seres queridos que estuvieron en todo momento conmigo, apoyándome, animándome y haciéndome ver que podía lograrlo.

Agradecimientos

Agradezco a la Facultad de Ciencias de la Electrónica por forjar las bases, los conocimientos, las habilidades y experiencias durante estos años los cuales fueron fundamentales para llevar a cabo este trabajo de tesis. Hago un extenso agradecimiento al Consejo Nacional de Ciencia y Tecnología (CONACYT) por haberme otorgado una beca que me permitió realizar mis estudios de Maestría en la Benemérita Universidad Autónoma de Puebla (BUAP).

Quiero agradecer a mis asesores Dr. Jesús Manuel Muñoz Pacheco, Dr. Germán Ardul Muñoz Hernández y Dr. Christos Volos por sus valiosos consejos, observaciones, enseñanzas, conocimientos compartidos y apoyo que me brindaron para el desarrollo, solución y conclusión de este trabajo. De igual manera quiero agradecer a los miembros de jurado Dra. María Monserrat Morín Castillo, Dra. Olga Guadalupe Félix Beltrán y Dr. José Eligio Moisés Gutiérrez Arias por sus observaciones, consejos, recomendaciones y enseñanzas para llevar a cabo y mejorar mi trabajo. También quiero agradecer al Dr. Esteban Tlelo Cuautle y al Instituto Nacional de Astrofísica, Óptica y Electrónica, por recibirme para realizar mis estancias, además de los conocimientos, enseñanzas y apoyo brindado.

Por último, pero no menos importante, quiero agradecer a mis padres Fidencio Hernández Miranda, Verónica Orbe Esquivel, a mis hermanos Daniel Hernández Orbe, Miguel Ángel Hernández Orbe, amigos y seres queridos que siempre estuvieron apoyándome, animándome, motivándome en todo momento tanto de manera académica, profesional y personal. También quiero agradecerles a mis compañeros y amigos de la Maestría en Ingeniería Electrónica: Tomás Orozco, Oscar Ramírez, Jonathan Morones, Alejandro Hernández, Edgar Daniel, Donato Maldonado, Juan Díaz, Leonardo Villalpando, Marcelino Luis y Viridiana Feernández quienes fueron parte fundamental durante este tiempo y este trabajo. También me gustaría agradecer a Anahy Ortíz, Karen Villanueva, Anahí Ramírez, Guadalupe Ramírez quienes estuvieron conmigo en todo momento apoyándome.

El autor de esta tesis también agradece al proyecto No. 258880 (Proyecto Apoyado por el Fondo Sectorial de Investigación para la Educación) por el apoyo brindado.

Resumen

El presente trabajo describe el diseño de generadores de números aleatorios basados en sistemas caóticos con atractores ocultos y orden fraccionario con los cuales se busca que sean utilizados en aplicaciones de cifrado de datos, principalmente cajas de sustitución (S-box) las cuales son la base para los métodos criptográficos.

Para ello a cabo este trabajo se realizaron diferentes actividades, entre ellas se realizó un análisis de la dinámica no lineal de diversos sistemas con atractores ocultos los cuales pertenecen a las familias de: sistemas con punto de equilibrio estable, sistemas con puntos de equilibrio infinitos, sistemas sin puntos de equilibrio y sistemas con mega estabilidad; también se llevaron los sistemas de orden entero a orden fraccionario por medio de a derivada de Grünwald-Letnikov, se buscaron formas para hacer la generación de números aleatorios y se implementaron metodologías para la generación de S-box.

Al utilizar los sistemas con atractores ocultos nos permite aumentar la complejidad en el cifrado de datos, la idea de utilizar orden fraccionario brinda un grado de libertad también ya que se toma como un parámetro más que debe ser averiguado.

La organización del presente trabajo esta dividida en varios capítulos iniciando con una breve introducción al tema que se esta por desarrollar, la justificación por la que se va a realizar, el planteamiento del problema que se desea resolver y los objetivos generales, así como específicos que se plantearon cumplir al llevar a cabo este tema de tesis.

En el capítulo uno corresponde al marco teórico, en este capítulo se presentará la información acerca del cálculo de orden fraccionario (derivada fraccionaria), el método a utilizar que es el la derivada de Grünwald-Letnikov, se habla también de la estabilidad en sistemas de orden fraccionario, se muestra información acerca de los sistemas no lineales y sobre caos, los sistemas caóticos con atractores ocultos y los generadores de números aleatorios (RNG).

Dentro del capítulo dos se muestra lo referente al análisis de la dinámica no lineal para conocer el comportamiento de los sistemas caóticos con atractores ocultos entre lo que se analiza están los puntos de equilibrio (estabilidad), los diagramas de fase para observar el atractor, las series de tiempo, el diagrama de bifurcación el cual nos permitirá ver el comportamiento del sistema al cambiar el orden fraccionario, el mapa de Poincaré y el exponente de Lyapunov, estos métodos nos permitirán observar y corroborar que el sistema es caótico.

En el capítulo tres se presenta lo referente sobre la instrumentación electrónica en sistemas embebidos ARM (Advanced RISC Machine) en la cual se utilizó una

tarjeta Raspberry Pi 3 modelo B+ en la cual se implementaron los códigos, las configuraciones y con ayuda de algunos dispositivos electrónicos poder observar los atractores en un osciloscopio, además se retomarán lo referente a generadores de números aleatorios para obtener un buen generador que apruebe las pruebas del estándar NIST y también se hablará acerca de las cajas de sustitución (S-box), los métodos que se utilizaron para generarlas y los códigos utilizados para ello.

Por último se mostrarán las conclusiones a las que se llegó y un apéndice en el cual se muestra la primera página de los artículos publicados, así como las constancias obtenidas y las referencias bibliográficas utilizadas para este trabajo.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Caos se refiere a un tipo de comportamiento dinámico complejo que posee algunas características muy especiales como lo son: extrema sensibilidad a pequeñas variaciones de las condiciones iniciales, trayectorias cerradas en el espacio de fase con un exponente de Lyapunov positivo, un espectro de potencia continua y/o una dimensión topológica fraccionaria, etc. [3, 4].

Este fenómeno se presenta en sistemas o procesos dinámicos importantes tales como la turbulencia en fluidos, dispositivos láser retroalimentados, vibraciones mecánicas debidas a fricción y procesos biológicos, entre otros [5].

De acuerdo con [6], existen tres identificadores básicos de caos:

- Es un sistema determinista, su condición actual es consecuencia de los estados previos del sistema.
- Es un sistema que exhibe comportamiento que es "difícil de distinguir del comportamiento aleatorio".
- Es un sistema que es sensible a las condiciones iniciales.

Esto ha generado un número grande de estudios relacionados a la dinámica caótica en sistemas dinámicos y las aplicaciones relacionadas. En ese contexto, las aplicaciones abarcan un rango amplio de áreas del conocimiento desde sistemas vivos (señales del cerebro, señales del corazón, razonamiento, epilepsia, depredadores-presa), hasta sistemas no vivos (convertidores de datos, generadores de números aleatorios, comunicaciones seguras, robots autónomos) [7, 8].

Dentro de los sistemas caóticos se encuentran dos tipos: de orden entero y de orden fraccionario. Los sistemas de orden entero son aquellos que emplean derivadas de orden entero, como se aprecia en la ecuación (1.1), donde n es un número entero positivo:

$$D^n(x) = f(x) \tag{1.1}$$

El cálculo de orden fraccionario es un área de las matemáticas que trata con derivadas e integrales de orden no entero [9]. Los sistemas de orden fraccionario tienen

la forma:

$$D^{\alpha}(x) = f(x) \tag{1.2}$$

donde $\alpha = [\alpha_1, \alpha_2, ..., \alpha_n]^T$ para $0 < \alpha_i < 1, (i = 1, 2, ..., n)$ y $x \in \mathbb{R}^n$ [10].

Entre las principales ventajas que ofrecen los sistemas de orden fraccionario, destaca el hecho de que pueden representar de manera más precisa los fenómenos naturales, debido a que poseen memoria.

Dentro de los sistemas caóticos se pueden observar los sistemas con atractores auto-excitados y con atractores ocultos. Los atractores auto-excitados tienen una región de atracción la cual es excitada por los puntos de equilibrio. Los atractores ocultos, cuentan con una región de atracción la cual no contiene puntos de equilibrio cercanos (vecinos). Los atractores ocultos no se pueden encontrar fácilmente por métodos numéricos tradicionales, además, el conocimiento sobre los equilibrios no ayuda a localizarlos [11].

La implementación de estos sistemas caóticos puede emplearse para distintas aplicaciones, aunque en la actualidad, el auge está en el cifrado de información, esto debido a que en muchas ocasiones se transmiten datos a través de redes públicas, lo cual posibilita que personas ajenas puedan o intenten acceder a la información.

1.1. Justificación

En la actualidad, el envío de información de manera segura se ha vuelto uno de los puntos vitales en cuanto a seguridad, esto debido al ser enviada por medios públicos, existiendo la posibilidad de invasión de la privacidad por personas ajenas. Por consiguiente, se buscan métodos alternativos de codificación de información, por medio de su robustez y rapidez, que proporcionen una mayor seguridad y que evite que personas ajenas puedan acceder a esa información. Una de las alternativas que se han empleado en los últimos años, son los sistemas caóticos, esto debido a su comportamiento determinista y a la gran sensibilidad que presenta al cambio de las condiciones iniciales [12].

Por lo tanto, surge la importancia del estudio de los sistemas caóticos, en este caso, más enfocado a los sistemas de orden fraccionario. En aplicaciones de cifrado de datos, en caso de que una persona externa quisiera acceder a la información, tendría primeramente que identificar el sistema caótico con el cual se está trabajando, posteriormente tendría que identificar las condiciones iniciales, además de tener que identificar el orden fraccionario que se está utilizando. Ésta última característica se podría visualizar como un grado extra de libertad [13], debido a que es un nuevo parámetro que descifrar, además de que existe un rango específico en el cual el sistema sigue presentando caos.

La finalidad de utilizar atractores ocultos es la complejidad que presentan para encontrarlos, a diferencia de los atractores auto-excitados, estos no presentan puntos de equilibrio que permitan visualizar en que región se ubican. Además, se necesitan procesos de análisis especiales (más complicados) en comparación con los sistemas con atractores auto-excitados [14].

Este tema surge debido a las investigaciones realizadas acerca del estudio de los atractores ocultos. Por el momento, no se ha encontrado documentación al respecto de cifrado de información con sistemas caóticos de orden fraccionario con atractores ocultos, por lo que, al llevar a cabo la implementación de un generador caótico con dichas características podría ser de gran utilidad en aplicaciones relacionadas a IoT (Internet of Things).

Por lo tanto, se propone la implementación de un generador caótico de orden fraccionario con atractores ocultos mediante una tarjeta embebida del tipo ARM, debido a que en la actualidad, este tipo de tarjetas representan alrededor del 90 % de todos los procesadores RISC embebidos de 32-bit (electrónica de consumo, asistentes digitales personales - PDA, medios de comunicación digitales y reproductores de audio, consolas de juegos, calculadoras, HDD, routers, etc.), por lo que son ampliamente utilizados [15].

1.2. Objetivos

1.2.1. Objetivo general

- Diseñar e instrumentar sistemas caóticos de orden fraccionario con atractores ocultos para su aplicación en un generador de números aleatorios.

1.2.2. Objetivos específicos

- Implementar un algoritmo numérico en MATLAB, para la solución de ecuaciones diferenciales de orden fraccionario.
- Determinar el comportamiento de atractores caóticos ocultos mediante diagramas de fase y series de tiempo.
- Analizar la dinámica no lineal de atractores caóticos ocultos mediante diagramas de bifurcación, mapas de Poincaré y exponentes de Lyapunov.
- Instrumentar de forma electrónica por lo menos dos de los sistemas caóticos con atractores ocultos en un sistema embebido ARM.
- Diseñar la aplicación de un generador de números aleatorios (RNG).

1.3. Estado del arte

El estudio de sistemas caóticos con atractores ocultos es vital para las aplicaciones de ingeniería ya que permite analizar la aparición de comportamientos inesperados en algunos sistemas dinámicos no lineales reales, por ejemplo, el sistema de perforación accionado por un motor de inducción con un rotor en espiral, el sistema de control de vuelo de las aeronaves con entrada de saturación, etc. También, se ha observado que los atractores ocultos desempeñan un papel crucial en los campos de circuitos no lineales, el oscilados de Val der Pol-Duffing, convertidores DC/DC multinivel, sistemas de relé con histéresis y modelo matemático del sistema de perforación. Además, los atractores ocultos permiten comprender respuestas inesperadas y potencialmente desastrosas provocadas por perturbaciones en estructuras como puentes o alas de aviones [14].

La localización de este tipo de atractores caóticos es un desafío, ya que no hay posibilidad de utilizar la información acerca de los puntos de equilibrio para organizar una búsqueda numérica similar al procedimiento computacional estándar usando un método de integración; por lo tanto,es difícil predecir la existencia de ellos [16]. A pesar de la dificultad para encontrar atractores ocultos, existen atractores generados por diferentes familias de sistemas dinámicos que también son definidos como atractores ocultos: sistemas con un número infinito de puntos de equilibrio, con puntos de equilibrio estable y sin puntos de equilibrio. La principal característica es que estos atractores ocultos pueden ser encontrados utilizando métodos numéricos tradicionales de integración.

Los sistemas caóticos con atractores ocultos han sido estudiados considerando sus derivadas de orden entero [17–20]. Por otro lado, hay pocas referencias acerca de los sistemas caóticos de orden fraccionario con atractores ocultos [21]. Más específicamente, la investigación de sistemas dinámicos de orden fraccionario ha atraído mucha atención debido a que proporciona resultados más precisos, puede describir el efecto de memoria y las propiedades de herencia de varios procesos. Esta clase de sistemas caóticos tiene características especiales comparado con los sistemas caóticos de orden entero ya que tienen una interpretación geométrica compleja, su espectro de potencia fluctúa de manera compleja incrementando la caoticidad en el dominio de la frecuencia y el orden fraccionario es un parámetro extra que aumenta el grado de libertad del sistema dinámico.

La importancia del cálculo de orden fraccionario radica en el hecho de que los sistemas de orden fraccionario poseen características de memoria y cuando se compara con sus contra partes de orden entero, presenta una dinámica más sofisticada. Por lo tanto, el estudio de la dinámica fraccionario ayuda a revelar los muchos comportamientos del sistema.

Una parte significativa para el desarrollo de aplicaciones de ingeniería consiste en las realizaciones físicas de sistemas caóticos usando circuitos electrónicos. Por un lado, se ha reportado y validado la implementación de sistemas caóticos de orden entero en diversos sistemas de hardware digital incorporado (FPGA, DSP, microcontroladores, etc.) [22–26]. Por otro lado, para la implementación física de sistemas

de orden fraccionario, el problema principal esta relacionado con la complejidad del algoritmo numérico utilizado para discretizar el sistema dinámico. Para realizar la discretización existen algunos métodos como por ejemplo, la definición de Caputo basada en el algoritmo de Adams-Bashfotrh-Moulton, el algoritmo de memoria corta de Grünwald-Letnikov y el método de descomposición de Adomian [27].

La realización de circuitos digitales de sistemas caóticos de orden fraccionario puede ser más adecuada para futuras aplicaciones portátiles, por ejemplo, Internet de las cosas (IoT), tecnología de drones, redes de sensores, wereables, entre otros [28–30]

Capítulo 2

Marco teórico

En este capítulo se describirá la teoría acerca de los la derivada fraccionaria, método de Grünwald-Letnikov, estabilidad de sistemas en orden fraccionario, los sistemas no lineales, caos atractores ocultos y sobre generadores de números aleatorios.

2.1. Definición de la derivada fraccionaria

El concepto de integración y diferenciación de orden no entero, no es algo nuevo. En 1695 L'Hopital le envió una carta a Leibniz, en esta carta, una pregunta acerca del orden de las derivadas emergió: ¿que significado tendría una derivada de orden n, si n tuviera el valor de una fracción. En una respuesta pronosticada, Leibniz predijo el inicio de lo que hoy en dia es llamado cálculo fraccionario. Desde entonces, muchos matemáticos como lo son: Euler, Laplace, Fourier, Abel, Liouville, Riemann, entre otros, han realizado contribuciones a la teoría del cálculo fraccionario [31]. A pesar de tener una larga historia, no se utilizó en física o ingeniería por muchos años. En los últimos años ha surgido un gran interés no solo en matemáticos, sino también entre físicos e ingenieros dado que se ha encontrado que los sistemas de orden fraccionario tienen ventajas en comparación con los sistemas de orden entero.

La novedad más relevante es que los sistemas de orden fraccionario consideran la historia del sistema, por lo que se dice que tiene memoria [32] y por lo tanto pueden aproximar con una mejor precisión los fenómenos reales [27].

A continuación, se definirá una de las más importantes funciones usadas en el cálculo fraccionario, la función Gamma de Euler, la cual se define como:

$$\Gamma(n) = \int_0^\infty t^{(n-1)} e^{-t} dt \tag{2.1}$$

Esta función es la generalización de un factorial en la siguiente forma:

$$\Gamma(n) = (n-1)! \tag{2.2}$$

El cálculo fraccionario es una generalización de funciones de integración y diferenciación de orden no entero, el operador integro-diferencial continuo ${}_aD_t^\alpha$, donde

a y t son los limites de la operación y $\alpha \in \mathbb{R}$. El operador es definido como [31]:

$$ {}_{\alpha}D_t^{\alpha} = \begin{cases} \frac{d^{\alpha}}{dt^{\alpha}}, & \alpha > 0 \\ 1, & \alpha = 0 \\ \int_a^t (d\tau)^{\alpha}, & \alpha < 0 \end{cases} \tag{2.3} $$

Diferentes definiciones de la integración y diferenciación de orden fraccionario han emergido durante el desarrollo de la teoría del cálculo de orden fraccionario. Algunas de las definiciones son directamente extendida del calculo de orden convencional. Algunas de las definiciones comúnmente mas usadas son resumidas a continuación.

Tres de las definiciones con más frecuencia utilizadas para la integral-diferencial de orden fraccionario son: la definición de Riemann-Liouville (RL), de Grünwald-Letnikov (GL) y la de Caputo. La definición de orden fraccionario de RL está dada como [31]:

$$ {}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau \tag{2.4} $$

para $(n-1 < \alpha < n)$, donde a y t son los limites de la operación y $\Gamma(\cdot)$ es la función Gamma de Euler (2.1). La definición de RL es considerada la primera definición de la derivada de orden fraccionario. La definición de RL calcula la integral de orden no entero y posteriormente calcula la derivada de orden entero.

Caputo propuso una definición para discutir problemas que involucraran ecuaciones diferenciales fraccionarias con condiciones iniciales. La diferencia entre la definición de RL y Caputo, es que este segundo calcula primero la derivada de orden entero y posteriormente calcula la integral de orden no entero [10]:

$$ {}_aD_t^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha-n+1}} d\tau \tag{2.5} $$

para $(n-1 < \alpha < n)$. Las condiciones iniciales para las ecuaciones diferenciales de orden fraccionario con la derivada de Caputo están en la misma forma que para las ecuaciones diferenciales de orden entero.
La definición de GL esta definida de la siguiente manera:

$$ {}_aD_t^{\alpha} f(t) = \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{j=0}^{\frac{t-a}{h}} (-1)^j \binom{\alpha}{j} f(t-jh) \tag{2.6} $$

donde se considera que $n = \frac{t-\alpha}{h}$, donde a es una constante real la cual expresa el valor limite, a y t son los limites de la operación para ${}_aD_t^{\alpha} f(t)$ y $\binom{\alpha}{j}$ son los coeficientes binomiales. Los coeficientes binomiales se calculan utilizando la relación entre la función Gamma de Euler y el factorial, a continuación se definen:

$$ \binom{\alpha}{j} = \frac{\alpha!}{j!(\alpha-j)!} = \frac{\Gamma(\alpha+1)}{\Gamma(j+1)\Gamma(\alpha-j+1)} \tag{2.7} $$

donde $\binom{\alpha}{0} = 1$.

Algunas de las principales propiedades de las derivadas de orden fraccionario son [31]:

- Si $f(t)$ es una función analítica de t, entonces su derivada fraccionario ${}_0D_t^\alpha f(t)$ es una función analítica de t, α.
- Para $\alpha = n$, donde n es un entero, la operación ${}_0D_t^\alpha f(t)$ da el mismo resultado que el clásico diferenciador de orden entero n.
- Para $\alpha = 0$ la operación ${}_0D_t^\alpha f(t)$ es el operador identidad:

$$ {}_0D_t^0 f(t) = f(t). \tag{2.8}$$

- El integrador y diferenciador fraccionario son operaciones lineales:

$$ {}_aD_t^\alpha(\lambda f(t) + \mu g(t)) = \lambda {}_aD_t^\alpha f(t) + \mu {}_aD_t^\alpha g(t) \tag{2.9}$$

- La ley del índice aditivo (propiedad de semigrupos) se mantiene bajo ciertas restricciones razonables de la función $f(t)$
- La regla de Leibniz para diferenciación fraccionario esta dada por:

$$ {}_aD_t^r(\phi(t)f(t)) = \sum_{k=0}^{\infty} \binom{r}{k} \phi^k(t) {}_aD_t^{r-k} f(t) \tag{2.10}$$

si $\phi f(t)$, $f(t)$ y todas sus derivadas son continuas en el intervalo de $[\alpha, t]$.

2.2. Método numérico para el cálculo de derivadas de orden fraccionario (Grünwald-Letnikov)

Para el cálculo numérico de la derivada de orden fraccionario se puede utilizar la relación (2.11) derivada de la definición de GL (2.6). Esta aproximación esta basada en el he hecho de que para cierta clase de funciones, las definiciones de GL(2.6), RL(2.4) y Caputo(2.5) son equivalentes bajo ciertas condiciones. Para la discretización y solución de sistemas de orden fraccionario se ha optado por utilizar la relación de la definición de la derivada fraccionario de GL. La relación para la aproximación numérica explícita de la derivada q en los puntos kh, donde $(k = 1, 2, ..., n)$, tiene la siguiente forma [31]:

$$ {}_{(k-\frac{L_m}{h})}D_{t_k}^q f(t) \approx h^{-q} \sum_{j=0}^{k} (-1)^j \binom{q}{j} f(t_{k-j}), \tag{2.11}$$

donde L_m es la "longitud de memoria", $t_k = kh$, h es el paso de integración de cálculo y $(-1)^j \binom{q}{j}$ son los coeficientes binomiales $c_j^{(q)} (j = 0, 1, \dots)$. Para el cálculo de los coeficientes binomiales se puede utilizar la siguiente expresión:

$$ c_0^{(q)} = 1,$$

$$ c_j^{(q)} = \left(1 - \frac{1+q}{j}\right) c_{j-1}^{(q)} \tag{2.12}$$

Entonces, la solución general de la ecuación de la derivada fraccionario es:

$$ {}_aD_t^q y(t) = f(y(t), t). \tag{2.13} $$

la ecuación anterior puede ser expresada de la siguiente manera:

$$ y(t_k) = f(y(t_k), t_k) h^q - \sum_{j=v}^{k} c_j^{(q)} y(t_{k-j}) \tag{2.14} $$

Para el *término de memoria* expresado por la sumatoria, el principio de "memoria corta"puede ser usado. Entonces el índice inferior de la sumatoria en la relación(2.14) sería $v = 1$ para $k < (L_m/h)$ y $v = k - (L_m/h)$ para $k > (L_m)/h$, en caso de no ser usado el principio de "memoria corta"se utilizaría $v = 1$ para toda k.

2.3. Estabilidad de sistemas de orden fraccionario

La estabilidad como una propiedad importante de los sistemas dinámicos, generalmente se refiere al comportamiento cualitativo de los movimientos en relación con un conjunto invariante. Es bien sabido por la teoría de la estabilidad que un sistema lineal invariante en el tiempo (LTI) es estable si las raíces del polinomio característico son negativas o tienen parte real negativa si son conjugadas complejas. Esto significa que están localizadas en la mitad izquierda del eje imaginario del plano complejo s. El dominio puede ser visto como una superficie de Riemann con un número finito de hojas de Riemann v, donde el origen es un punto de rama y donde se asume que el punto de corte de rama pertenece a $\mathbb{R}^-$. El corte de rama es asumido que pertenece a $\mathbb{R}^-$ y la primera hoja de Riemann es denotada por Ω y definida como:

$$ \Omega = \{re^{j\phi} | r > 0, -\pi < \phi < \pi\}. \tag{2.15} $$

En el caos de LTI de orden fraccionario, la estabilidad es diferente del caso de orden entero [33].

$$ \begin{aligned} D_t^{\mathbf{q}} x(t) &= \mathbf{A}x(t) + \mathbf{B}u(t) \\ y(t) &= \mathbf{C}x(t) \end{aligned} \tag{2.16} $$

Un punto interesante es que un sistema fraccionario estable puede tener raíces en la mitad derecha del plano complejo w, como se puede ver en la figura 2.1. Dado que la hoja principal de la superficie de Riemann es definida $-\pi < arg(s) < \pi$, considerando el mapeo $w = s^\alpha$, el dominio de w es definido por $-\alpha\pi < arg(w) < \alpha\pi$ y la región del plano w corresponde a la mitad derecha del plano de esta hoja es descrito por $-\alpha\pi/2 < arg(w) < \alpha\pi/2$. Se ha demostrado que el sistema (2.16) es estable si las siguientes condiciones se satisfacen [34, 35].

Partiendo de las ecuaciones (2.3) y (2.5), es posible estudiar la estabilidad de sistemas de orden fraccionario. Una ecuación diferencial de orden fraccionario con $0 < q < 1$ (α es sustituido por q), típicamente presenta una región de estabilidad que es mayor que la de la misma ecuación con orden entero $q = 1$.

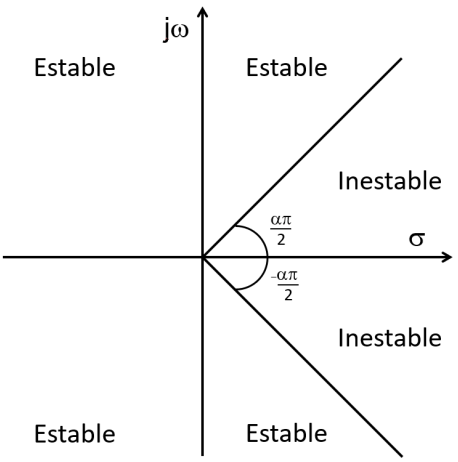


Figura 2.1: Región de estabilidad de un sistema LTI de orden fraccionario con orden $0 < \alpha < 1$.

Definición 2.3.1 *Las raíces de la ecuación* $\boldsymbol{f}(\boldsymbol{x}) = 0$ *son llamados equilibrios del sistema diferencial de orden fraccionario* $D^q\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})$*, donde* $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}$*,* $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}$ *y* $D^q x = (D^{q_1}x_1, D^{q_2}x_2, ..., D^{q_n}x_n)^T$*,* $q_i \in \mathbb{R}^+$*,* $i = 1, 2, ..., n$*.*

Teorema 2.3.1 *Considere un sistema de orden conmensurado descrito por:*

$$D^q\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}, \;\; \boldsymbol{x}(0) = x_0 \tag{2.17}$$

con $0 < q < 1$*,* $\boldsymbol{x} \in \mathbb{R}^n$ *y* $\boldsymbol{A} \in \mathbb{R}^{nxn}$*. Se ha demostrado en [36–41] que este sistema de orden fraccionario es asintóticamente estable si y solo si se satisface la siguiente condición:*

$$|arg(\lambda)| > q\pi/2, \tag{2.18}$$

donde $|arg(\lambda)|$ *representa todos los valores propios de* $\boldsymbol{A}$*. Además, los valores propios críticos de* $\boldsymbol{A}$ *satisfacen* $|arg(\lambda)| > q\pi/2$ *deben tener una multiplicidad geométrica de uno, que representa la dimensión del subespacio de* $\boldsymbol{v}$ *para* $\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$*.*

Teorema 2.3.2 *Considere un sistema de orden inconmensurado descrito por:*

$$D^q\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}, \;\; \boldsymbol{x}(0) = x_0 \tag{2.19}$$

donde $\boldsymbol{x} = (x_1, x_2, ..., x_n)^T \in \mathbb{R}$*,* $D^q\boldsymbol{x} = (D^{q_1}x_1, D^{q_2}x_2, ..., D^{q_n}x_n)^T$*,* $q_i \in \mathbb{R}^+$*,* $i = 1, 2, ..., n$*,* $0 < q_i < 1$ *y* $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{nxn}$*,* $i = 1, 2, ..., n$*,* $j = 1, 2, ..., n$*. Asumiendo* w *como el mínimo común múltiplo de los denominadores* u_i *de* q_i*, donde* $q_i = v_i/u_i$*,* $(u_i, v_i) = 1$*,* $u_i, v_i \in Z^+$ *para* $i = 1, 2, ..., n$*, la matriz característica de* (2.19) *es definida como:*

$$\Delta(\lambda) = \begin{bmatrix} \lambda^{wq_1} - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \lambda^{wq_2} - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \lambda^{wq_n} - a_{nn} \end{bmatrix} \tag{2.20}$$

Entonces, el sistema (2.19) *es globalmente asintóticamente estable en el sentido de Lyapunov si todas las raíces* λ *de su polinomio característico, dadas por la ecuación* $det(\Delta(\lambda)) = 0$*, satisface* $|arg(\lambda)| > \pi/2w$ *[36–41].*

Teorema 2.3.3 *El punto de equilibrio E_* es asintóticamente estable si y solo si la medida de inestabilidad:*

$$\rho = (\pi/2w) - min\{arg(\lambda_i)\} \tag{2.21}$$

es estrictamente negativo, donde los parámetros λ_i son raíces de las ecuaciones: $det(diag([\lambda^{wq_1}\ \lambda^{wq_2}...\lambda^{wq_n}]) - \partial f/\partial x|_{x=E_*}) = 0$, $\forall E_* \in \Omega$ [40, 41]. Si $\rho \geq 0$ y los valores propios críticos satisfacen $\rho = 0$ tiene una multiplicidad geométrica de uno, entonces E_* es estable.

Nota 2.3.1 *Si ρ es positivo, entonces E_* es inestable y el sistema puede exhibir comportamiento caótico [40, 41].*

2.4. Sistemas dinámicos no lineales

Considere un sistema general de dos variables de estado unidimensional:

$$\begin{aligned} D^\alpha x_1 &= f_1(x_1, x_2), \\ D^\alpha x_2 &= f_2(x_1, x_2), \end{aligned} \tag{2.22}$$

donde $0 < \alpha < 1$, con condiciones iniciales $(x_1(0), x_2(0))$ y dos funciones lineales suaves, $f_1(\cdot,\cdot)$ y $f_2(\cdot,\cdot)$. Aquí (f_1, f_2) describen el campo vectorial del sistema. La trayectoria recorrida por la solución del sistema (2.22), comenzando desde el estado inicial $(x_1(0), x_2(0))$, la cual es una trayectoria de solución o una órbita del sistema. Una solución de (2.22), con estado inicial $(x_1(0), x_2(0))$ es usualmente denotada como $\varphi(x_1(0), x_2(0))$. La familia de φ_t, $t \in [0, \infty)$, satisface $\varphi_{t_1+t_2} = \varphi_{t_1} \circ \varphi_{t_1}$, donde $\varphi_{t_0}(x_1(0), x_2(0)) = (x_1(0), x_2(0))$. Dado que, para sistemas autónomos, dos trayectorias de solución diferentes nunca se cruzan una entre ellas en el plano x_1, x_2, cualquier solución $\varphi_t(x_1(0), x_2(0))$ de un sistema autónomo, considerando como una familia de trayectorias con condiciones iniciales diferentes, es llamado flujo en el plano x_1, x_2. Todas las posibles trayectorias de solución de un sistema autónomo, graficadas en el plano x_1, x_2 correspondientes a diferentes condiciones iniciales, constituyen el retrato de fase de las soluciones del sistema.

Los equilibrios del sistema (2.22), si existen, son las soluciones que simultáneamente satisfacen la ecuación homogénea $f(x) = 0$, los equilibrios son usualmente denotados como (x_1, x_2). Un equilibrio es estable si todas las trayectorias cercanas del sistema, comenzando desde cualquier estado inicial, se acercan a él; se dice que es inestable, si las trayectorias cercanas se alejan de él. Los equilibrios pueden se clasificados, de acuerdo con sus estabilidades, como nodo, nodo estable o inestable, foco, punto de silla o centro.

Sean λ_1, λ_2 los valores propios de la matriz Jacobiana $J = \partial f/\partial x$, todos evaluados en el equilibrio (x_1, x_2). Como es bien sabido,

$$\lambda_{1,2} = \frac{1}{2}[trace(J) \pm \sqrt{D}], \tag{2.23}$$

donde $D = [trace(J^2) - 4det(J)]$. Los diferentes equilibrios y sus estabilidades, determinados por estos dos valores propios son resumidos en la figura 2.2 [42, 43]. Si

los dos valores propios de J satisfacen $\mathbb{R}\{\lambda_{1,2}\} \neq 0$, entonces el equilibrio x^*, sobre el cual se toma la linealización, se dice que es hiperbólico.

Considere el sistema lineal homogéneo,

$$D^{\alpha}x(t) = Ax(t), \quad x(t) \in \mathbb{R}^n, \tag{2.24}$$

donde A es una matriz constante de nxn, $x(0) = x_0$, y α esta restringido en $(0, 1)$.

Definición 2.4.1 *[44] Si todos los valores propios $\lambda(A)$ de A satisfacen: $|\lambda(A)| \neq 0$ y $|arg(\lambda(A))| \neq \alpha\pi/2$, entonces el origen O del sistema lineal es llamado punto de equilibrio hiperbólico.*

Definición 2.4.2 *[44] El sistema autónomo* (2.24) *se dice que es: (1) estable si $\forall x_0$, existe un $\epsilon > 0$ tal que $||x(t)|| < \epsilon$ para $t \geq 0$; (2) asintóticamente estable si $lim_{t\to\infty}||x(t)|| = 0$.*

El sistema diferencial autónomo no lineal en el sentido de Caputo es dado de la siguiente manera:

$$D^{\alpha}x(t) = f(x(t)) \tag{2.25}$$

donde $x(t) \in \mathbb{R}^n$, $x(0) = x_0$, f(x) es continuo.

Definición 2.4.3 *El punto e es un punto de equilibrio del sistema* (2.25)*, si y solo si $f(e) = 0$.*

Definición 2.4.4 *[44] Suponga que e es un punto de equilibrio del sistema* (2.25)*, y todos los valores propios $\lambda(Df(e))$ de la matriz linealizada $Df(e)$ en el punto de equilibrio e satisfacen: $|\lambda(Df(e))| \neq 0$ y $|arg(\lambda(Df(e)))| \neq \alpha\pi/2$, entonces llamamos e un punto de equilibrio hiperbólico.*

Definición 2.4.5 1) *El punto de equilibrio e de* (2.25) *se dice que es: (1) localmente estable si $\forall\epsilon > 0$, existe un $\delta > 0$ tal que $||x(t) - e|| < \epsilon$ valido para $\forall x_0 \in \{z : ||z - e|| < \delta\}$ y $\forall t > 0$, (2) localmente asintóticamente estable si el punto de equilibrio es localmente estable y $lim_{t\to+\infty}x(t) = e$.*

2) *Se denota $x(t)$, $\hat{x}(t)$ como la solución del sistema* (2.25) *con valores iniciales $x_0, \hat{x}_0$ respectivamente. La solución $x(t)$ se dice que es: (1) localmente estable si $\forall\epsilon > 0$, existe un $\delta > 0$ tal que $||x(t) - \hat{x}(t)|| < \epsilon$ valido para $||x_0 - \hat{x}_0|| < \delta$ y $\forall t \geq 0$; (2) localmente asintóticamente estable si la solución es localmente estable y $lim_{t\to+\infty}(x(t) - \hat{x}(t)) = 0$.*

2.5. Caos

No existe una definición universal aceptada de caos en la literatura. Entre algunas definiciones dadas en términos matemáticos hay dos ligeramente diferentes pero bien aceptadas ambas, una es dada por Li y Yorke donde formalmente empieza el uso del nombre caos en la literatura científica y de ingeniería moderna [42], mientras que la segunda es quizá mejor conocida. La segunda definición dada por Devaney establece que si un mapa, $M : S \to S$, donde S es generalmente un conjunto compacto e invariante bajo $M \in \mathbb{R}^n$, se dice que es caótico si [42, 45]:

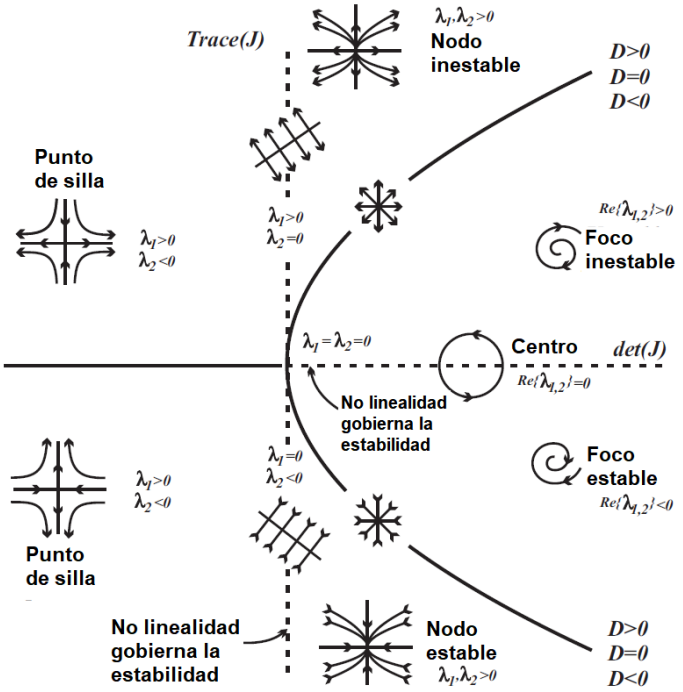


Figura 2.2: Clasificación de equilibrios bidimensionales: las estabilidades están determinadas por sus valores propios.

- M es transitivo en S, en el sentido de que para cualquier par de conjuntos abiertos no vacíos U y V en S, hay un número entero $k > 0$, tal que $M^k(U) \cap V$ es no vacío;
- Los puntos periódicos de M son densos en S;
- M tiene una dependencia sensible de las condiciones iniciales, es decir, hay un número real $\delta > 0$ que depende solo de M y S, de modo que en cada subconjunto abierto no vacío de S hay par de puntos que eventualmente itera bajo M están separados por una distancia de al menos δ.

Se han observado y analizado algunas características diferentes sobre caos, por ejemplo, la extrema sensibilidad de las condiciones iniciales; una característica distintiva del caos, en el sentido de que dos conjuntos de condiciones iniciales similares pueden dar lugar a dos estados asintóticos diferentes de la trayectoria del sistema. Exponentes positivos de Lyapunov que miden la tasa de divergencia de trayectorias cercanas; en concreto, el exponente positivo de Lyapunov es la tasa de crecimiento del logaritmo promedio en el tiempo de la máxima distancia entre las órbitas cercanas, y es quizá la más conveniente para verificar en aplicaciones en ingeniería. La entropía de Kolmogorov-Sinai, es otra característica distintiva, utilizó la idea relacionada con la entropía estática, la cual es una medida de la cantidad de información que es necesaria para determinar el estado de los sistemas para definir una medida de la intensidad de un conjunto de estados del sistema, la cual proporciona la información media pérdida en el estado del sistema cuando este evoluciona con el tiempo. Esta medida también puede ser usada para caracterizar atractores extraños y caos. El cero simple de la función de la teoría de Melnikov proporciona una medida

de la distancia entre una variedad estable e inestable. La teoría de Melnikov dice que el caos es posible si estas variedades se cruzan, lo cual corresponde a que la función de Melnikov tiene un cero simple, y puede ser usado para caracterizar los enfoques del caos en los puntos de silla del mapa de Poincaré de flujos continuos en el espacio de fase. Entre otros, que se consideran características distintivas del caos [10].

Los sistemas caóticos se conocen desde hace mucho tiempo, pero recientemente se demostró que el caos podía controlarse y, por lo tanto, sincronizarse [46, 47]. Por esta razón, esta clase de sistemas prometen tener un gran impacto en muchas aplicaciones novedosas, de tiempo crítico y de energía, como circuitos y dispositivos de alto rendimiento (por ejemplo, moduladores delta-sigma y convertidores de potencia), mezcla líquida, reacciones químicas, sistemas biológicos (por ejemplo, en el cerebro humano, el corazón, proceso perceptivo), la gestión de crisis (por ejemplo, en electrónica de potencia), el procesamiento seguro de la información y la toma de decisiones críticas en eventos políticos, económicos y militares [7, 8, 48].

El caos se encuentra intuitivamente relacionado con el desorden o la aleatoriedad y es un comportamiento impredecible a largo plazo de un sistema determinista [49]. El caos es un comportamiento determinista, es decir, que su evolución en el tiempo se encuentra gobernada por leyes precisas y, por tanto, su estado siguiente puede ser solo uno. La extrema sensibilidad a las condiciones iniciales puede entenderse mejor si tomamos como ejemplo dos trayectorias dentro del espacio de fase inicialmente vecinas y observamos como al pasar un cierto tiempo, estas tienden a separarse exponencialmente. Una manera de comprobar la existencia de caos en un sistema es mediante la presencia de exponentes positivos de Lyapunov. Cuando los exponentes de Lyapunov son negativos, significan una convergencia en las trayectorias, por el contrario, al ser positivos, estos muestran la velocidad de divergencia entre las trayectorias. El sistema debe contar con al menos un exponente positivo de Lyapunov para que sea *caótico* y si tiene más de dos se le llama al sistema *hipercaótico*.

Dentro de los sistemas caóticos se encuentran dos tipos de atractores, los atractores caóticos auto-excitados y los atractores ocultos.

2.6. Atractores ocultos

Los ejemplos más familiares de flujos caóticos de baja dimensión ocurre en sistemas que tienen uno o más puntos de silla. Sin embargo, estudios posteriores mostraron que los las oscilaciones autoexcitadas periódicas y caóticas no proporcionaban información exhaustiva sobre los posibles tipos de oscilaciones, es decir, "oscilasciones ocultasz "atractores ocultos". Entonces esta clase de atractores se introdujeron de acuerdo a la siguiente definición:

Definición 2.6.1 *Un atractor es llamado atractor auto-excitado si su cuenca de atracción cruza con alguna vecindad abierta de un equilibrio, de otra manera es llamado atractor oculto [16, 50].*

Con equilibrio, estamos indicando los puntos de equilibrio de las variables de estado. La definición anterior 2.6.1 también incluye sistemas dinámicos de orden

fraccionario sin puntos de equilibrio, en línea yb superficies de equilibrios, y equilibrios estables [9, 51–57].

El atractor oculto es de considerable importancia en aplicaciones de ingeniería debido a la ocurrencia de comportamientos inesperados en algunos sistemas dinámicos no lineales reales como lo son: el sistema de perforación accionado por motor de inducción con rotor enrollado, el sistema de control de vuelo de la aeronave con saturación de entrada, etc [14].

El problema de la localización numérica, el cálculo y la investigación analítica de los atractores ocultos es mucho más difícil. Esto sucede, ya que en este caso no hay posibilidad de utilizar información sobre equilibrios para la organización de procesos transitorios similares en el procedimiento computacional estándar. Por lo tanto, los atractores ocultos no se pueden calcular utilizando este procedimiento estándar [50]. Uno de los métodos efectivos para la localización numérica de atractores ocultos es basada en la homotopía y la continuación numérica [58]. Además, en este caso es poco probable que la integración de trayectorias en datos iniciales aleatorios proporcione una localización oculta del atractor, ya que una cuenca de atracción puede ser muy pequeña y la dimensión del atractor oculto en sí misma puede ser mucho menor que la dimensión del sistema considerado [14].

Un atractor oculto con una pequeña cuenca de atracción no está asociado con un punto de equilibrio inestable, debido a ello los atractores ocultos existentes se pueden encontrar en algunos sistemas dinámicos particulares relacionados con puntos de equilibrio infinitos, sin puntos de equilibrio y con puntos de equilibrio estable, es por ello que este tipo de sistemas han recibido mucha atención en los últimos años.

2.7. Generadores de números aleatorios (RNG)

Knuth en [59] define números aleatorios como una secuencia de números independientes con una distribución especifica y una probabilidad especifica de caer en cualquier rango de valores dado. Como resultado, el generador de números aleatorios (RNG) ideal proporcionará un flujo de bits distribuidos uniformemente, no deterministas e independientes sobre un conjunto de datos infinitos. Además, como se conoce hoy en día, debido a que la evaluación matemática de la aleatoriedad es difícil, solo es posible utilizar análisis estadísticos en conjuntos de datos de muestra para detectar características que apuntan a no aleatoriedad, una forma es mediante la realización de pruebas del conjunto NIST (National Institute of Standards and Technology) [60].

Hay muchas aplicaciones que necesitan RNG, por ejemplo [61]: entretenimiento, las loterías y máquinas de apuestas se basan en el uso de números aleatorios, los videojuegos usan números aleatorios para influir en la heurística de inteligencia o para variar el juego, la música y composición de gráficos entrelazando el contenido con bits aleatorios, modelos científicos y financieros complejos que utilizan números aleatorios para simulación; las aplicaciones de inteligencia artificial que requieren datos aleatorios para determinar la precisión de la clasificación o el comportamiento de la red neuronal, los fabricantes de software utilizan datos aleatorios para probar

programas y algoritmos para detectar errores, solución de ecuaciones, criptografía, firmas digitales, protocolos de comunicación protegidos, etc.

Un RNG es un dispositivo físico o software desde el cual se obtiene una secuencia de números binarios aleatorios. Hay dos métodos principales utilizados para generar números aleatorios. El primer método produce resultados que dependen de alguna fuente física impredecible que está fuera del control humano. Algunos ejemplos incluyen medir el ruido atmosférico, el ruido térmico y otros fenómenos electromagnéticos y cuánticos externos, si este generador es un sistema no determinista; entonces, se llama generador de números aleatorios verdadero (TRNG). El segundo método es un sistema determinista que utiliza algoritmos computacionales que pueden producir secuencias largas de resultados aparentemente aleatorios, a este sistema se le llama generador de números pseudoaleatorios (PRNG).

2.7.1. Generadores de números pseudoaleatorios (PRNG)

Una secuencia de números pseudoaleatorios es una secuencia de números generados de una manera sistemática tal que son independientes y estadísticamente indistinguibles de una secuencia verdaderamente aleatoria. Un generador de números pseudoaleatorios (PRNG) es un algoritmo matemático que, dado un estado inicial, produce una secuencia de números pseudoaleatorios. Los PRNG son algoritmos implementados en máquinas de estado finito y son capaces de generar secuencias de números que parecen aleatorios desde muchos aspectos. Aunque son necesariamente periódicos, sus períodos son muy largos, aprueba muchas pruebas estadísticas y se pueden implementarse fácilmente con rutinas de software simples y rápidas [62].

Un PRNG tiene varias ventajas sobre un verdadero número aleatorio generador en que la secuencia generada es repetible, tiene propiedades matemáticas conocidas y puede ser implementado sin necesidad de ningún hardware especializado.

Algunas características que presentan los PRNG son las siguientes:

- Eficiente: PRNG puede producir muchos números en poco tiempo y es ventajoso para aplicaciones que necesitan muchos números.
- Determinístico: una secuencia dada de números puede reproducirse en una fecha posterior si se conoce el punto de inicio de la secuencia. El determinismo es útil si necesita volver a reproducir la misma secuencia de números nuevamente en una etapa posterior.
- Periódico: los PRNG son periódicos, lo que significa que la secuencia eventualmente se repetirá. Si bien la periodicidad casi nunca es una característica deseable, los PRNG modernos tienen un período que es tan largo que puede ser ignorado para la mayoría de los propósitos prácticos.

Debido a que los PRNG emplean un algoritmo matemático para la generación de números, todos los PRNG poseen las siguientes propiedades: se requiere un valor semilla para inicializar la ecuación y la secuencia se realizara un ciclo después de un periodo en particular. Un requerimiento de estos generadores es que tengan buenas propiedades estadísticas, esto significa que las secuencias generadas deben

aproximarse a secuencias aleatorias, por lo tanto, los desarrolladores de aplicaciones deben proporcionar un valor inicial indiscutible y un algoritmo con un periodo suficientemente largo, también deben verificar que la salida de PRNG no contenga correlación ni sesgo. Como resultado, toda la secuencia aparentemente aleatoria se puede reproducir si se conoce el valor de la semilla [63].

El Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) de los Estados Unidos de América creo un software gratuito para probar generadores de números pseudo y aleatorios (PRNG). Específicamente, la NIST dijo que las pruebas son para PRNG para aplicaciones criptográficas, donde la aleatoriedad es una característica crucial. El paquete esta disponible en [1], y de acuerdo con [60] se aborda el problema de evaluar PRNG para determinar la aleatoriedad. Esto será útil para: identificar PRNG que producen secuencias binarias débiles (o con patrones), diseñar nuevos PRNG, verificar que las implementaciones de PRNG sean correctas, estudiar PRNG descritos en estándares, e investigando el grado de aleatoriedad de los PRNG utilizados actualmente [64].

Las pruebas que realiza el software de la NIST para evaluar los generadores de números aleatorios, son las siguientes [60]:

- Prueba de frecuencia (Frequency (Monobit) Test)
- Prueba de frecuencia dentro de un Bloque (Frequency Test within a Block)
- Prueba de recorrido (Runs Test)
- Prueba de recorrido más largo de "1.$^{\text{en}}$ un bloque (Tests for the Longest-Run-of-Ones in a Block)
- Prueba aleatoria de rango de matriz binaria (Binary Matrix Rank Test)
- Prueba discreta de transformación de Fourier (espectral) (Discrete Fourier Transform (Spectral) Test)
- Prueba de coincidencia de plantillas no superpuestas (aperiódicas) (Non-overlapping Template Matching Test)
- Prueba de coincidencia de plantillas superpuestas (periódicas) (Overlapping Template Matching Test)
- Prueba estadística universal de Maurer (Maurer's "Universal Statistical"Test)
- Prueba de complejidad Lineal (Linear Complexity Test)
- Prueba en serie (Serial Test)
- Prueba de entropía aproximada (Approximate Entropy Test)
- Prueba de suma acumulativa (Cusums) (Cumulative Sums (Cusums) Test)
- Prueba de excursiones aleatorias (Random Excursions Test)
- Prueba variante de excursiones aleatorias (Random Excursions Variant Test)

[1] https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software

Capítulo 3

Dinámica de los sistemas caóticos con atractores ocultos

En este capítulo se analizará la dinámica de diferentes sistemas caóticos con atractores ocultos los cuales están clasificados en diferentes familias que son: sistemas con puntos de equilibrio infinitos, sistemas con puntos de equilibrio estable y sistemas sin puntos de equilibrio. Además de estas familias se analizará un sistema que es llamado con mega estabilidad. Para realizar el análisis de los sistemas no lineales y así poder seleccionar los más adecuados, se utilizarán 5 métodos los cuales permitirán observar el comportamiento que presentan al cambiar sus parámetros, además, permitirán determinar en qué valores del orden fraccionario el sistema es caótico y en qué valores no. Los métodos que se utilizarán son: (i) series de tiempo, (ii) diagramas de fase, (iii) mapas de Poincaré, (iv) diagrama de bifurcación y (v) exponentes de Lyapunov. A continuación, se describirá un poco de cada uno de estos métodos.

3.1. Familias de sistemas con atractores ocultos

Dentro de los sistemas caóticos con atractores ocultos existen tres familias las cuales están definidas por sus puntos de equilibrio del sistema, a continuación se presentarán sistemas de cada una de esas familias.

3.1.1. Familia de sistemas con puntos de equilibrio estable

Como el nombre lo menciona, esta familia de sistemas esta definido debido a que los puntos de equilibrio que tiene son estables, a continuación se presentan las ecuaciones del sistema de Wang-Chen [14] un sistema perteneciente a dicha familia.

$$\begin{aligned} \dot{x} &= yz + a, \\ \dot{y} &= x^2 - y, \\ \dot{z} &= 1 - 4x. \end{aligned} \tag{3.1}$$

En este sistema el parámetro a es un parámetro de control. Partiendo de las ecuaciones (3.1) se procede a obtener sus puntos de equilibrio, esto se hace igualando el lado izquierdo de las ecuaciones con cero.

$$\begin{aligned} 0 &= yz + a, \\ 0 &= x^2 - y, \\ 0 &= 1 - 4x. \end{aligned} \tag{3.2}$$

Al resolver la ecuación anterior podemos observar que el punto de equilibrio de dicho sistema se encuentra en $E(0.25, 0.0625, -16a)$. Linealizando el sistema (3.1) en el punto de equilibrio E, la matriz Jacobiana esta dada por:

$$J_E = \begin{bmatrix} 0 & -16a & 0.0625 \\ 0.5 & -1 & 0 \\ -4 & 0 & 0 \end{bmatrix} \tag{3.3}$$

Por lo tanto, la ecuación característica del sistema (3.1) es:

$$\lambda^3 + \lambda^2 + (0.25 + 8a)\lambda + 0.25 = 0 \tag{3.4}$$

Acorde con el criterio de Routh-Hurwitz, el punto de equilibrio es estable para:

$$\begin{cases} 0.25 + 8a > 0 \\ 1(0.25 + 8a) > 0.25 \end{cases} \tag{3.5}$$

es decir $a > 0$. Un dato interesante del sistema (3.1) es que con un solo punto de equilibrio estable genera comportamiento caótico.

Las ecuaciones del sistema con puntos de equilibrio estable se pueden expresar en orden fraccionario con la derivada de orden fraccionario de Grünwald-Letnikov (2.14), se muestran de esta manera ya que posteriormente serán evaluadas con el algoritmo para la derivada de orden fraccionario que se presentará posteriormente:

$$\begin{aligned} x(t_k) &= (y(t_{k-1})z(t_{k-1}) + a)h^q - \sum_{j=1}^{k} c_j^{(q)} x(t_{k-j}), \\ y(t_k) &= (x(t_{k-1})^2 - y(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} y(t_{k-j}), \\ z(t_k) &= (1 - 4x(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} z(t_{k-j}). \end{aligned} \tag{3.6}$$

3.1.2. Familia de sistemas con puntos de equilibrio infinitos

Esta familia de sistemas cuenta con un gran número de puntos de equilibrio por eso es que se les conoce como infinitos, a continuación se presentan las ecuaciones de un sistema que es conocido como STR [14].

$$\begin{aligned} \dot{x} &= -yz, \\ \dot{y} &= (ax + y + z^2)z, \\ \dot{z} &= x - x^3. \end{aligned} \tag{3.7}$$

Para obtener los puntos de equilibrio de este sistema, el lado izquierdo de las ecuaciones son igualadas con cero.

$$
\begin{aligned}
0 &= -yz, \\
0 &= (ax + y + z^2)z, \\
0 &= x - x^3.
\end{aligned} \tag{3.8}
$$

Los equilibrios del sistema son: $Ex(\pm 1, y, 0), Ey(0, 0, z), Ez(-1, 0, \pm\sqrt{2})$. El sistema (3.7) presenta líneas de puntos de equilibrio cuando el parámetro $a = 2$, estas líneas se pueden observar en la variable y del equilibrio en Ex y en la variable z en el equilibrio Ey, además de presentar comportamiento caótico con estas líneas de equilibrios.

Las ecuaciones del sistema con puntos de equilibrio infinitos también se pueden expresar con la derivada de orden fraccionario de Grünwald-Letnikov (2.14), se muestran de esta manera ya que posteriormente serán evaluadas con el algoritmo para la derivada de orden fraccionario que será presentado más adelante:

$$
\begin{aligned}
x(t_k) &= (-y(t_{k-1})z(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} x(t_{k-j}), \\
y(t_k) &= ((ax(t_{k-1}) + y(t_{k-1}) + z(t_{k-1})^2)z(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} y(t_{k-j}), \\
z(t_k) &= (x(t_{k-1}) - x(t_{k-1})^3)h^q - \sum_{j=1}^{k} c_j^{(q)} z(t_{k-j}).
\end{aligned} \tag{3.9}
$$

3.1.3. Familia de sistemas sin puntos de equilibrio

El sistema que se presenta a continuación es un sistema sin puntos de equilibrio, es un sistema construido por Wei [14] con un parámetro de control a, a continuación se presentan las ecuaciones del sistema.

$$
\begin{aligned}
\dot{x} &= -y, \\
\dot{y} &= x + z, \\
\dot{z} &= 2y^2 + xz - a.
\end{aligned} \tag{3.10}
$$

Los puntos de equilibrio del sistema son obtenidos igualando el lado izquierdo de las ecuaciones con cero.

$$
\begin{aligned}
0 &= -y, \\
0 &= x + z, \\
0 &= 2y^2 + xz - a.
\end{aligned} \tag{3.11}
$$

Mediante esta igualdad se observa que no existen puntos de equilibrio cuando el parámetro $a > 0$, ya que los puntos de equilibrio son soluciones en valores reales y en este caso se obtiene que $x = z = \sqrt{-a}$ lo cual daría un valor complejo y es por ello que se dice que no cuenta con puntos de equilibrio, además se obtiene que este sistema (3.10) presenta un comportamiento caótico cuando $a = 0.35$.

Las ecuaciones del sistema sin puntos de equilibrio se pueden expresar con la derivada de orden fracciona de Grünwald-Letnikov (2.14), esta forma de expresarlas permite evaluarlas con el algoritmo para la derivada de orden fraccionario que será mostrado en la siguiente sección:

$$\begin{aligned} x(t_k) &= (-y(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} x(t_{k-j}), \\ y(t_k) &= (x(t_{k-1}) + z(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} y(t_{k-j}), \\ z(t_k) &= (2y(t_{k-1}) + x(t_{k-1})z(t_{k-1}) - a)h^q - \sum_{j=1}^{k} c_j^{(q)} z(t_{k-j}). \end{aligned} \tag{3.12}$$

3.1.4. Sistema con mega estabilidad

El sistema con mega estabilidad tiene un comportamiento interesante debido a que puede presentar múltiples atractores conectados. El sistema de ecuaciones es basado en el modelo de un fracmemristor [65], a continuación se muestran las ecuaciones del sistema:

$$\begin{aligned} \dot{x} &= z + 2.5x\sin(0.5\pi y), \\ \dot{y} &= 1 - |x|, \\ \dot{z} &= -x - 0.1z(-0.2 + 0.01w^2) \\ \dot{w} &= 1z. \end{aligned} \tag{3.13}$$

Al igualar el lado izquierdo de las ecuaciones con cero se obtienen los puntos de equilibrio del sistema.

$$\begin{aligned} 0 &= z + 2.5x\sin(0.5\pi y), \\ 0 &= 1 - |x|, \\ 0 &= -x - 0.1z(-0.2 + 0.01w^2) \\ 0 &= 1z. \end{aligned} \tag{3.14}$$

Resolviendo (3.14) se puede observar que el sistema no cuenta con puntos de equilibrio, esto se nota en el hecho de que se tiene una doble igualdad de la variable x al resolver la cuarta ecuación se obtiene que $z = 0$, sustituyendo este valor en la tercera ecuación se obtiene que $x = 0$ y en la segunda ecuación se obtiene que $|x| = 1$, aquí se encuentra la doble igualdad de x. Además se tiene que si se sustituyen en la primera ecuación estos valores $z = 0, x = 0$ daría como resultado $0 = 0$, entonces por este conjunto de resultados es que se dice que no tiene puntos de equilibrio, pero el sistema presenta comportamiento caótico con estas determinaciones.

Las ecuaciones del sistema con mega estabilidad se pueden expresar con la derivadas de orden fraccionario de Grünwald-Letnikov (2.14), se muestran de esta manera ya que posteriormente serán evaluadas con el algoritmo para la derivada de orden fraccionario, el algoritmo se muestra en la siguiente sección:

$$
\begin{aligned}
x(t_k) &= (z(t_{k-1}) + 2.5x(t_{k-1})\sin(0.5\pi y(t_{k-1})))h^q - \sum_{j=1}^{k} c_j^{(q)} x(t_{k-j}),\\
y(t_k) &= (1 - abs|x(t_{k-1})|)h^q - \sum_{j=1}^{k} c_j^{(q)} y(t_{k-j}),\\
z(t_k) &= (-x(t_{k-1}) - 0.1z(t_{k-1})(-0.2 + 0.01w(t_{k-1})^2))h^q - \sum_{j=1}^{k} c_j^{(q)} z(t_{k-j}),\\
w(t_k) &= (z(t_{k-1}))h^q - \sum_{j=1}^{k} c_j^{(q)} w(t_{k-j}),
\end{aligned}
\tag{3.15}
$$

3.2. Algoritmo para método de Grünwald-Letnikov

A continuación, se describirá un pseudocódigo que fue utilizado para llevar a cabo la implementación del método de Grünwald-Letnikov, el cual fue descrito en la sección 2.2 y parte de la ecuación (2.14), que se muestra a continuación:

$$y(t_k) = f(y(t_k), t_k)h^q - \sum_{j=v}^{k} c_j^{(q)} y(t_{k-j})$$

Este algoritmo servirá de base para poder llevar a cabo el análisis de los sistemas de las diferentes familias que fueron mencionadas en la sección anterior con ello se podrán mostrar sus series de tiempo y el atractor caótico que se muestra, además de que es base para llevar a cabo los métodos de análisis de la dinámica no lineal inicialmente propuestos. El código fue realizado en el program Matlab.

```
1  //Se asignan los valores de los parametros del sistema
2  a = 1
3  b = 0.1
4  c = 1
5
6  //Se asignan las condiciones iniciales
7  x(1) = 1
8  y(1) = 2
9  z(1) = -1
10
11 //Se asigna el orden de las ecuaciones del sistema
12 q1 = 1
13 q2 = 1
14 q3 = 1
15
16 //Se asignan los valores de la longitud de memoria (Lm), el paso de
     integracion (h) y las iteraciones (k)
17 Lm = 300
18 h = 0.015
19 k = 20000
20
21 //Se asigna el valor de uno al primer coeficiente, despues se
     calculan los coeficientes binomiales restantes
```

```
C1p = 1
C2p = 1
C3p = 1

for j = 1 hasta k
C1(j)=(1-(1+q1)/j)*C1p
C2(j)=(1-(1+q2)/j)*C2p
C3(j)=(1-(1+q3)/j)*C3p
C1p = C1(j)
C2p = C2(j)
C3p = C3(j)
end for
//Se calculara el sistema de acuerdo a la memoria utilizada para las ecuaciones del sistema. Tambien se determina la longitud de memoria que sera utilizada
for n = 1 hasta k
if k es menor o igual que (Lm/h) entonces
v = 1
j = n
else, si k es mayor que (Lm/h) entonces
v = n-(Lm/h)+1
j = Lm/h
end if
x(n+1)=f(x(n))(h^{q1}) - sum(C1(1:j)*fliplr(x(v:n)))
y(n+1)=f(y(1))(h^{q2}) - sum(C2(1:j)*fliplr(y(v:n)))
z(n+1)=f(z(n))(h^{q3}) - sum(C3(1:j)*fliplr(z(v:n)))
end for

//Finalmente, se grafican los valores almacenados en los vectores
plot(x, y)
```

3.3. Series de tiempo

Una serie temporal (o simplemente una serie) es una secuencia de N observaciones (datos) ordenadas y equidistantes cronológicamente sobre una característica (serie univariante o escalar) o sobre varias características (serie multivariante o vectorial) de una unidad observable en diferentes momentos. El primer objetivo del análisis de una serie temporal consiste en la elaboración de un modelo estadístico que describa la procedencia de dicha serie. Después, el modelo elaborado a partir de la serie temporal considerada puede utilizarse para [66]:

- Describir la evolución observada de dicha serie, así como las relaciones contemporáneas y dinámicas entre sus componentes (en el caso de series multivariantes).
- Prever la evolución futura de dicha serie.
- Contrastar alguna teoría sobre las características o variables a las que se refieren los componentes de dicha serie.

Una serie temporal es el resultado de observar una variable a lo largo del tiempo, generalmente en intervalos regulares (cada día, cada mes, cada año, etc). Cuando los valores de la serie oscilan alrededor de un nivel fijo con variabilidad constante a lo

largo del tiempo, la dependencia entre las observaciones sólo depende de la distancia que las separa y permanece constante en el tiempo decimos que la serie es estable o estacionaria. En otro caso, decimos que la serie es no estacionaria [67]. Algunas características que se pueden apreciar en las series de tiempo son la tendencia, variación estacional, variación cíclica y variación irregular.

3.3.1. Serie de tiempo del sistema con punto de equilibrio estable

Partiendo del algoritmo de Grünwald-Letnikov descrito en la sección 3.2 se calculan los valores para el sistema con puntos de equilibrio estable (3.1) para poder graficar las series de tiempo correspondientes a cada una de las ecuaciones. La figura 3.1 muestra las series de tiempo obtenidas.

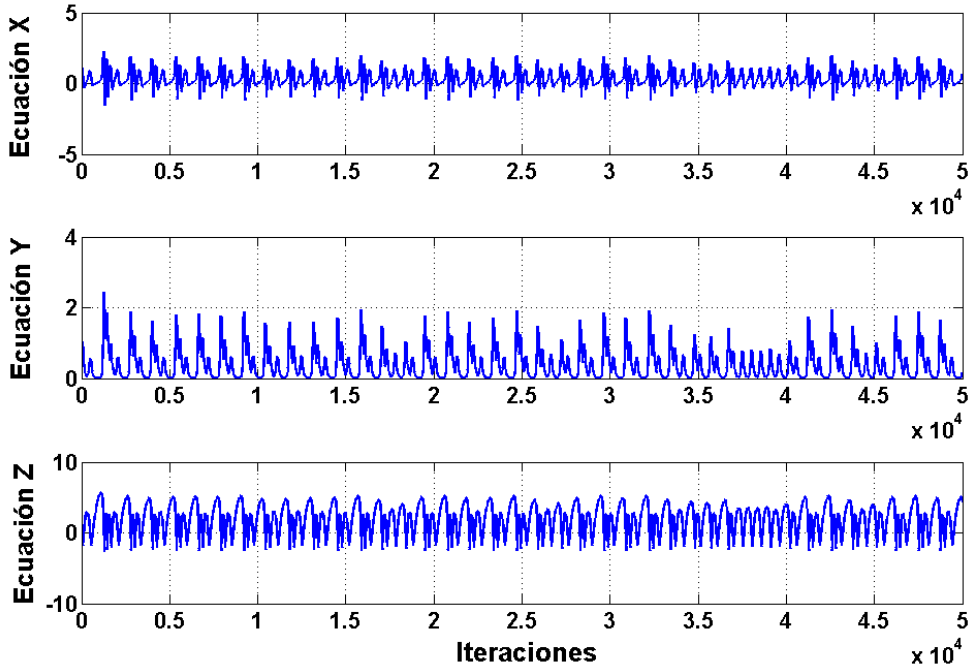


Figura 3.1: Serie de tiempo del sistema con puntos de equilibrio estable (3.1).

3.3.2. Serie de tiempo del sistema con puntos de equilibrio infinitos

De igual manera para el sistema con puntos de equilibrio infinito (3.7) se obtienen sus series de tiempo. La figura 3.2 muestra las series de tiempo de las ecuaciones del sistema con puntos de equilibrio infinitos.

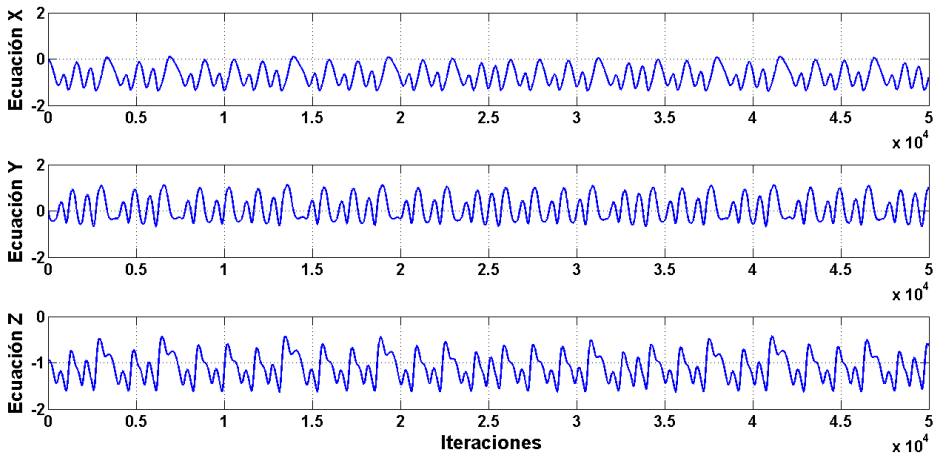


Figura 3.2: Serie de tiempo del sistema con puntos de equilibrio infinitos (3.7).

3.3.3. Serie de tiempo del sistema sin puntos de equilibrio

Se utilizo de base el mismo código 2.2 cambiando los parámetros, las condiciones iniciales y el sistema. Haciendo esto se procede a calcular las series de tiempo del sistema sin puntos de equilibrio (3.10), la figura 3.3 muestra los resultados obtenidos.

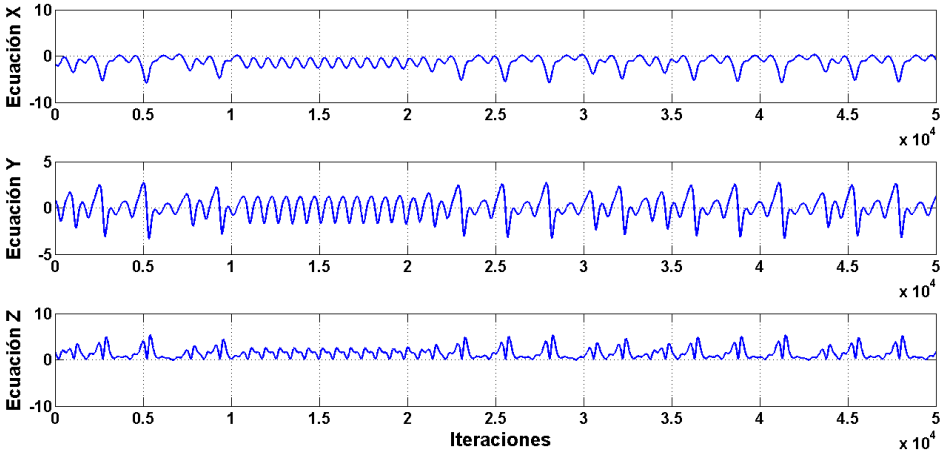


Figura 3.3: Serie de tiempo del sistema sin puntos de equilibrio (3.10).

3.3.4. Serie de tiempo del sistema con mega estabilidad

Se calcula también los valores necesarios para poder graficar las series de tiempo del sistema con mega estabilidad (3.13). La figura 3.4 muestra los resultados obtenidos para las series de tiempo.

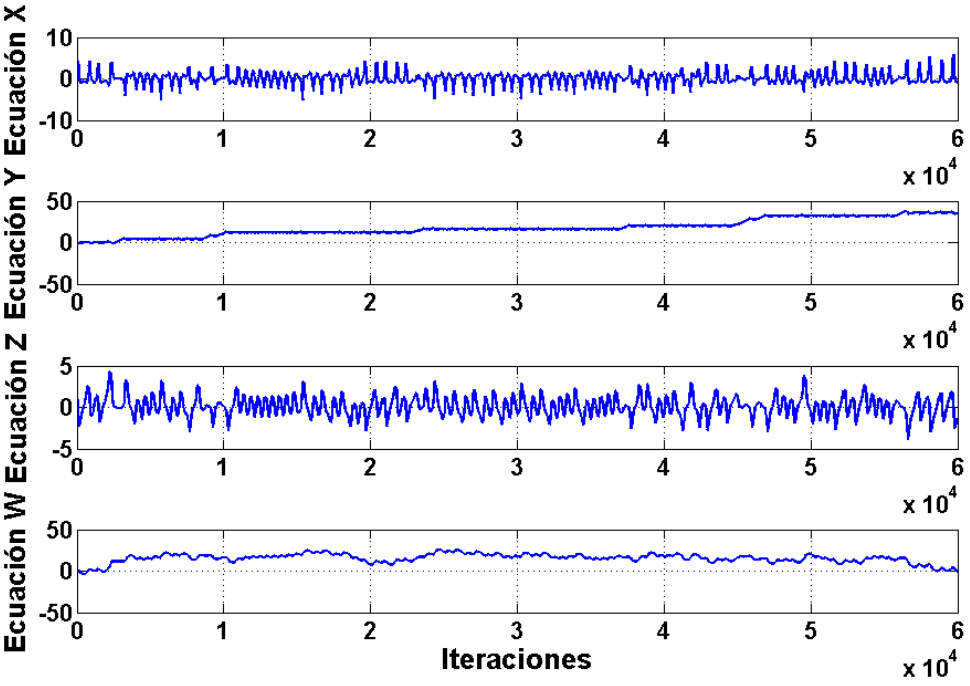


Figura 3.4: Serie de tiempo del sistema con mega estabilidad (3.13).

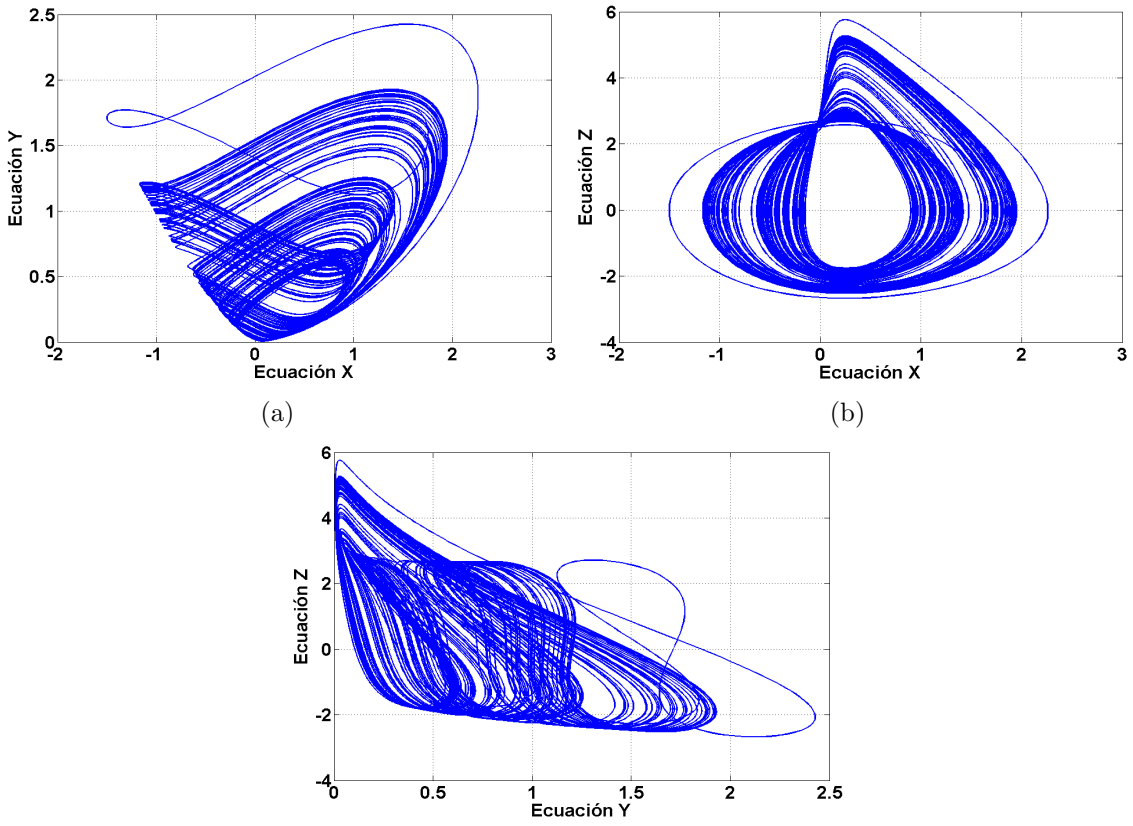
3.4. Diagramas de fase

Los diagramas de fase son un conjunto de curvas que describe el comportamiento de un sistema, es una construcción matemática que permite representar el conjunto de posiciones y momentos conjugados de un sistema. Los sistemas de ecuaciones representables con este tipo de diagramas son los autónomos, es decir los que no dependen directamente de la variable independiente; en estos sistemas no aparece la variable independiente explícitamente en las ecuaciones que forman el sistema. Los diagramas de fases no muestran una trayectoria bien definida, sino que ésta es errante alrededor de algún movimiento bien definido. Cuando esto sucede se dice que el sistema es atraído hacia un tipo de movimiento, es decir, que hay un atractor.

La representación de diagramas de fase, es una manera de poder observar los atractores creados por los diferentes sistemas que se han presentado.

3.4.1. Diagramas de fase del sistema con punto de equilibrio estable

Para obtener los diagramas de fase también se utiliza el algoritmo de la sección 3.2, a partir de dicho algoritmo se calculan los valores de cada una de las ecuaciones para el sistema con punto de equilibrio estable (3.1). Habiendo obtenidos todos los valores se graficaron las fases correspondientes, la figura 3.5 muestra los resultados. En la figura 3.6 se muestra el diagrama de fase en 3 dimensiones.



(a) (b) (c)

Figura 3.5: Sistema con puntos de equilibrio estable (3.1) con diagrama de fase (a) $x - y$, (b) $x - z$, (c) $y - z$.

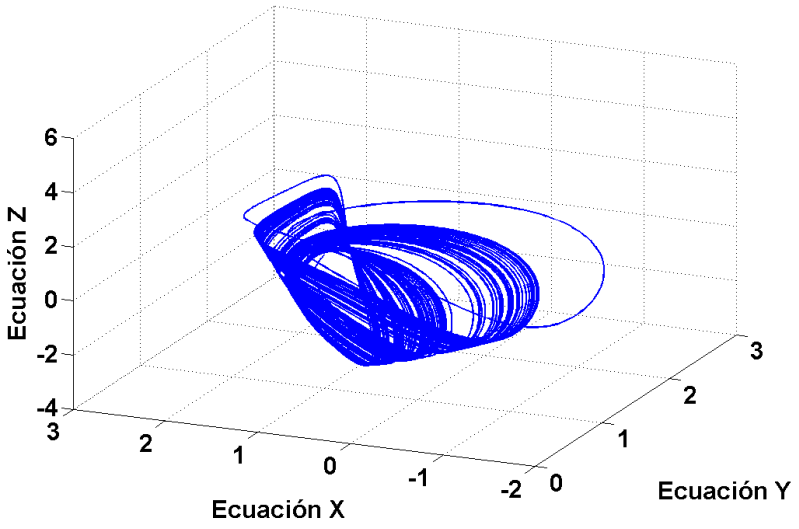
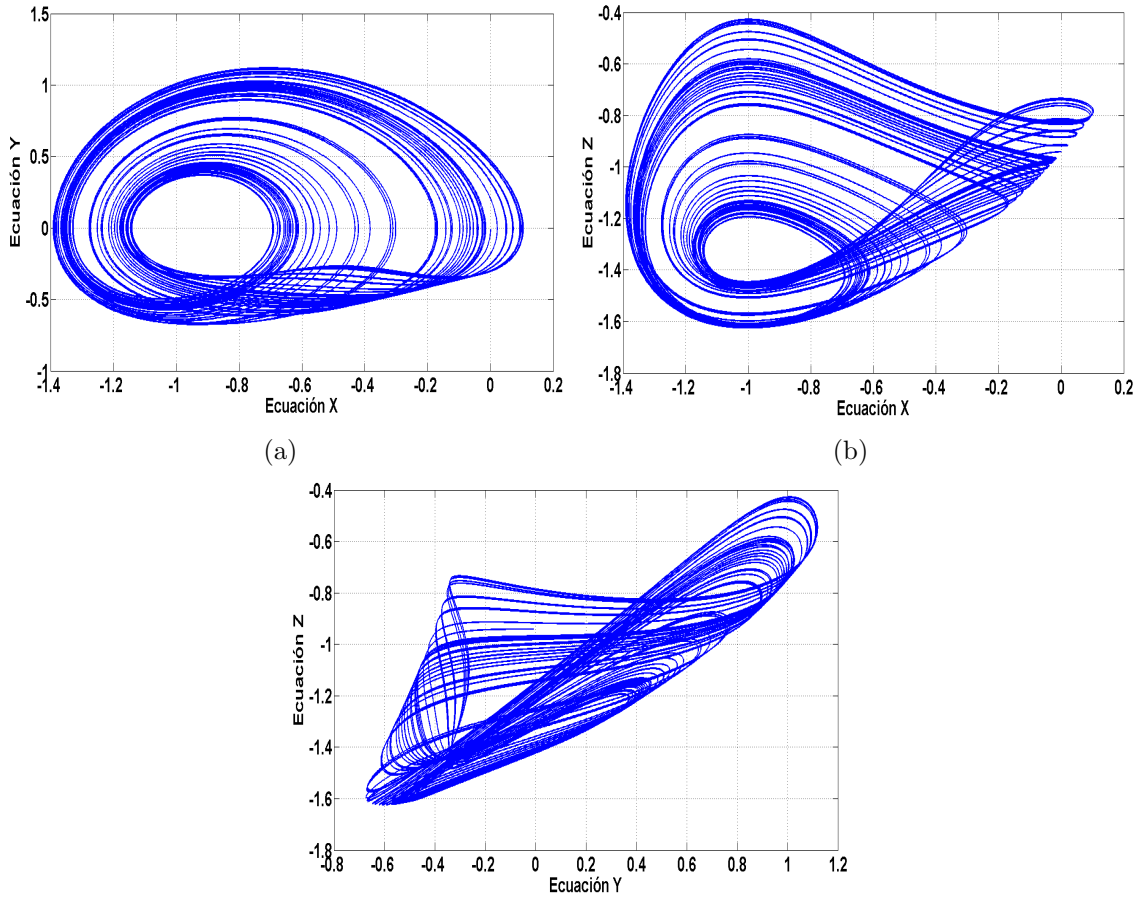


Figura 3.6: Sistema con puntos de equilibrio estable (3.1) con diagrama de fase en 3 dimensiones.

3.4.2. Diagramas de fase del sistema con puntos de equilibrio infinitos

De igual manera se procedió a calcular los valores para poder graficar los diagramas de fase del sistema con puntos de equilibrio infinitos (3.7), la figura 3.7 muestra los diagramas obtenidos. En la figura 3.8 se muestra el diagrama en 3 dimensiones.



(a) (b)

(c)

Figura 3.7: Sistema con puntos de equilibrio infinitos (3.7) con diagrama de fase (a) $x - y$, (b) $x - z$, (c) $y - z$.

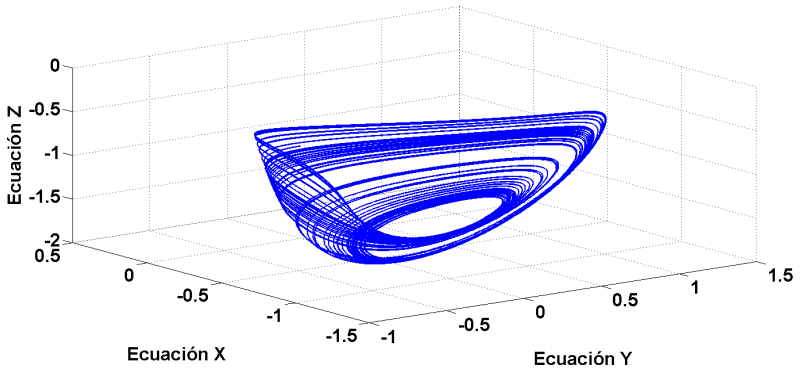


Figura 3.8: Sistema con puntos de equilibrio infinitos (3.7) con diagrama de fase en 3 dimensiones.

3.4.3. Diagramas de fase del sistema sin puntos de equilibrio

Para el sistema sin puntos de equilibrio (3.10), también se utilizó el algoritmo de la sección 3.2 para calcular sus valores correspondientes y así poder graficar sus diagramas de fase, la figura 3.9 muestra los resultados obtenidos. En la figura 3.10 se puede observar el diagrama de fase obtenido en 3 dimensiones.

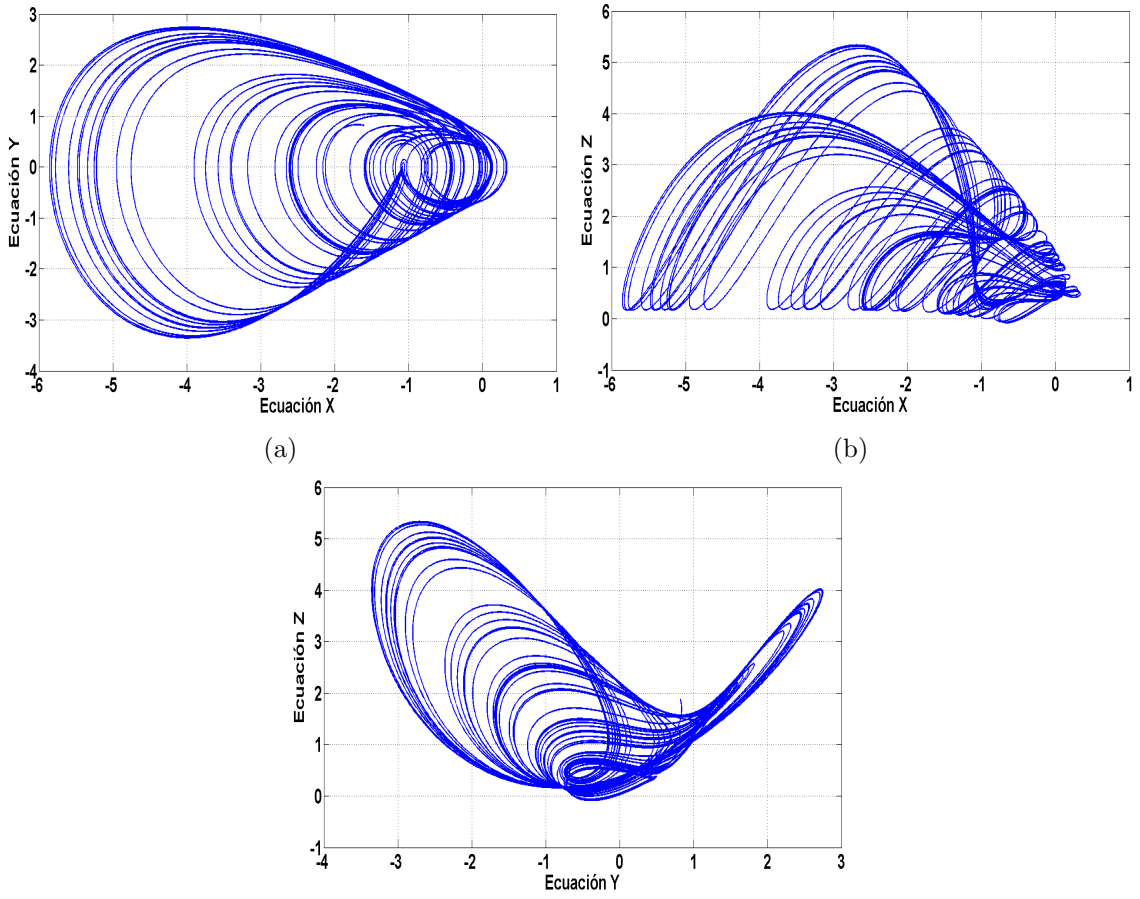


Figura 3.9: Sistema sin puntos de equilibrio (3.10) con diagrama de fase (a) $x - y$, (b) $x - z$, (c) $y - z$.

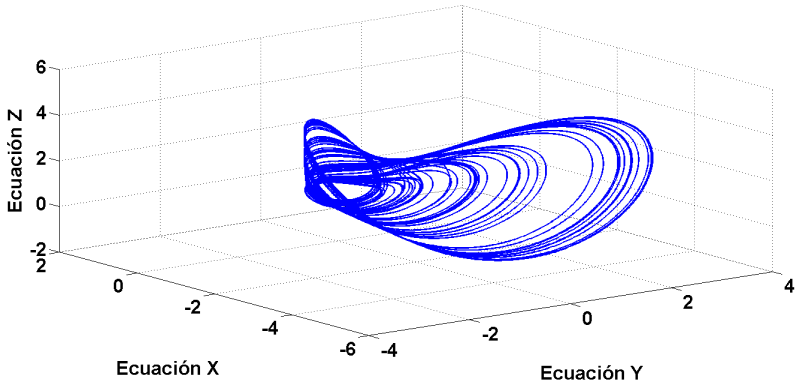


Figura 3.10: Sistema sin puntos de equilibrio (3.10) con diagrama de fase en 3 dimensiones.

3.4.4. Diagramas de fase del sistema con mega estabilidad

EL algoritmo de la sección 3.2 se modificó un poco, para las cuatro ecuaciones, y fue utilizado para calcular los valores del sistema con mega estabilidad (3.13), obteniendo los valores se graficaron sus fases, la figura 3.12 muestra dichas fases, en la fase $x-y$ se puede observar que este sistema genera múltiples atractores conjuntos. En la figura 3.11 se observa el diagrama de fase en 3 dimensiones.

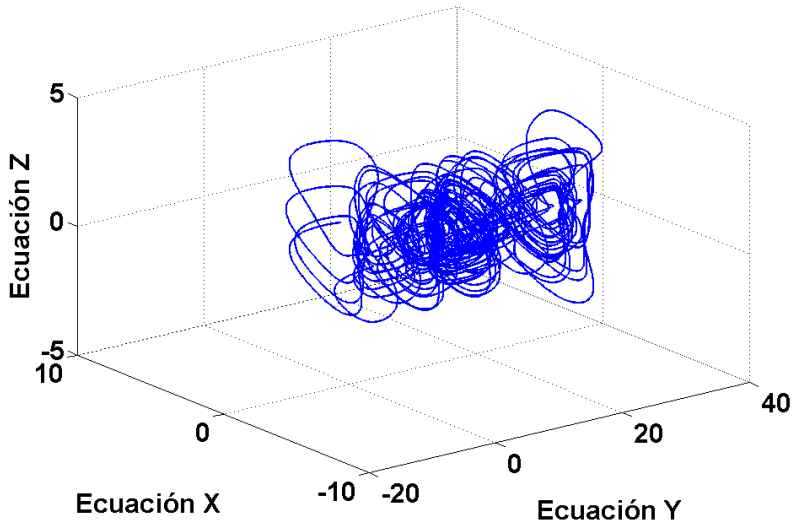
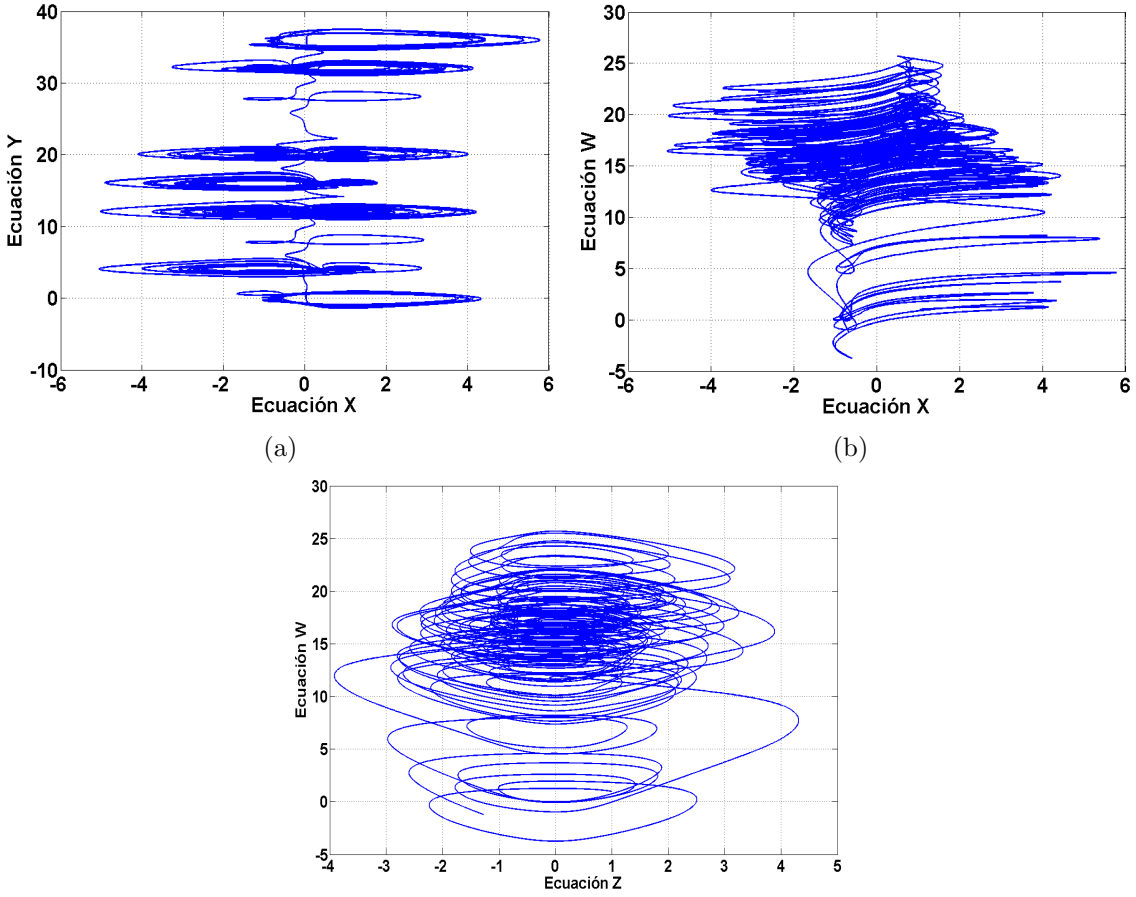


Figura 3.11: Sistema con mega estabilidad (3.13) con diagrama de fase en 3 dimensiones.

3.5. Mapas de Poincaré



(a) (b)

(c)

Figura 3.12: Sistema con mega estabilidad (3.13) con diagrama de fase (a) $x - y$, (b) $y - z$, (c) $z - w$.

Una técnica clásica para la discretización de un sistema dinámico es el mapa de Poincaré. Este reemplaza el flujo de un sistema continuo de orden n con un mapa discreto de orden $(n-1)$ el cual es llamado mapa de Poincaré. El mapa de Poincaré es útil para reducir el sistema, ordena y cierra la brecha entre sistemas continuos y discretos [68].

Esta técnica ofrece varias ventajas en el estudio de ecuaciones diferenciales ordinarias, incluidas las siguientes [69]:

- Reducción dimensional. La construcción del mapa de Poincaré involucra la eliminación de al menos una de las variables del problema resultante, permitiendo así, el estudio de un problema dimensional inferior.
- Dinámica global. En problemas dimensionales inferiores (por ejemplo, dimensión ≤ 4) los mapas de Poincaré numéricamente calculados proporcionan una visión profunda de la dinámica global del sistema.

- Claridad conceptual. Muchos conceptos que son algo complejos para declarar en ecuaciones diferenciales ordinarias a menudo pueden ser indicado para el mapa de Poincaré asociado. Un ejemplo sería la noción de estabilidad orbital de una órbita periódica de una ecuación diferencial ordinaria. En términos del mapa de Poincaré, este problema se reduciría al problema de la estabilidad de un punto fijo del mapa, que simplemente se caracteriza en términos de los valores propios del mapa linealizado sobre el punto fijo.

El mapa de Poincaré juega un rol importante en la clasificación de atractores de sistemas autónomos. Por ejemplo, atractores periódicos tienen un número finito de puntos en una sección del mapa de Poincaré, los atractores cuasi periódicos tienen puntos en forma cerrada en la superficie, y los atractores caóticos tienen un número infinito de puntos en el mapa de Poincaré [70].

Hénon presentó un método en el cual una sección del plano (mapa de Poincaré) es definido y se recolectan los puntos distribuidos en la trayectoria [71]. La variable dependiente la cual define el mapa de Poincaré es tomada como variable independiente alterando las ecuaciones dinámicas apropiadamente, inmediatamente después de que la trayectoria cruza el mapa de Poincaré. Entonces, las ecuaciones resultantes son integradas de manera *backward* (hacia atrás), para encontrar la posición exacta del punto en el que la trayectoria del sistema original cruzo el mapa de Poincaré. Por último, la integración de las ecuaciones dinámicas originales se mantiene del punto en el cual se detectó el cruce por el mapa.

Sin perder esta idea Tucker ha presentado un nuevo método [72], en este método las ecuaciones de evolución se transforman de tal manera que una de las variables dependientes (es decir, la variable cuya evolución se describe por el componente dominante del campo vectorial) se convierte en la variable independiente y t (la variable independiente inicial) se convierte en una variable dependiente.

La definición más frecuente de la forma para la intersección del mapa de Poincaré es:

$$x_N - a = 0, \tag{3.16}$$

donde a es constante. Para aproximar x_N al valor de a, Hénon introdujo una idea simple en la cual el sistema con el que se trabaja es transformado en una manera que x_N es una variable independiente en lugar de t. Esto es realizado dividiendo las primeras $(N-1)$ ecuaciones del sistema por la última e invirtiendo la última ecuación. El resultado es descrito por:

$$\begin{aligned} \frac{\mathrm{dx_1}}{\mathrm{d}x_N} &= \frac{f_1}{f_N}, \\ &\vdots \\ \frac{\mathrm{dx_{N-1}}}{\mathrm{d}x_N} &= \frac{f_{N-1}}{f_N}, \\ \frac{\mathrm{dt}}{\mathrm{d}x_N} &= \frac{1}{f_N}. \end{aligned} \tag{3.17}$$

A continuación se presentan los pasos de este nuevo método que es conocido como Algoritmo de Hénon modificado [70].

1. Se integra el sistema con un paso de integración definido h.

2. Se detiene la integración inmediatamente después de que se detecta un cruce por el mapa de Poincaré ($\sum$). El componente x_N de la distancia entre el mapa de Poincaré y el primer punto de integración (después del cruce) se calcula como Δx_N.

3. Se guardan los valores actuales que tienen todas las variables (i.e., $x_i, i = 1, 2, ..., N$).

4. Se calcula el siguiente punto integrando la ecuación (2.19) con un paso de integración $-\Delta x_N$.

5. Se resetean los valores de todas las variables guardadas en el paso 3.

6. Se continúa integrando el sistema con el paso de integración h.

3.5.1. Algoritmo para calculo de mapa de Poincaré

Habiendo estudiado el método para la obtención del mapa de Poincaré se procede a calcularlo, para ello se hizo un código en el programa Matlab, a continuación se mostrará un pseudocódigo describiendo los puntos más relevantes para el calculo del mapa de Poincaré. El código es complementario al código utilizado en la sección 3.2.

```
1  //Se inicia por seleccionar un punto/valor en alguna de las
     ecuaciones que sera la referencia de cruce, por ejemplo se
     selecciona el valor de 44.92 en la ecuacion x. Se procede por
     integrar el sistema con un paso de integracion h con el metodo de
     integracion seleccionado, en este caso es con el metodo de Grunwald
     -Letnikov.
2
3  //Se utiliza un condicional para detectar cada vez que la ecuacion
     haga un cruce por el valor seleccionado
4  if x(n) > 44.92 y x(n-1) < 44.92 entonces
5
6  //Cuando detecte un cruce se detendra la integracion y se realizara
     una diferencia entre el valor actual de x(n) y el valor en que se
     puso el plano 44.92, este valor sera el nuevo paso de integracion
     Delta_{xN} (se sustituira por h)
7  Deltax = x(n) - 44.92
8
9  //Se guardan los ultimos valores de cada ecuacion x(n), y(n), z(n) y
     se calcula el valor de ten el cual es una multiplicacion entre el
     paso de integracion y la iteracion en la que paso por el punto del
     mapa.
10 ec1 = x(n)
11 ec2 = y(n)
12 ec3 = z(n)
13 ten = h*n
14
15 //Se hace el calculo con el nuevo paso de integracion -Delta_{xN} y
     se dividen las ecuaciones sobre la ecuacion donde se coloco el
     plano de Poincare
16 t(tk)/x(tk) = 1/f(x(tk))*((-Deltax)^{q1}) - sum(C1(1)*ten)
17 y(tk)/x(tk) = f(y(tk))/f(x(tk))*((-Deltax)^{q2}) - sum(C2(1:j)*y(v:n)
     )
18 z(tk)/x(tk) = f(z(tk))/f(x(tk))*((-Deltax)^{q3}) - sum(C3(1:j)*z(v:n)
     )
```

```
19
20 //Se crea una variable inicializada en cero que se incrementara con
     cada integracion, ademas se utilizaran vectores para guardar los
     valores obtenidos con el nuevo paso de integracion
21 w = 0
22 w = w + 1
23 F(1,w) = t(tk)/x(tk)
24 G(1,w) = y(tk)/x(tk)
25 H(1,w) = z(tk)/x(tk)
26
27 //Se resetean los valores de las variables almacenadas anteriormente
     x(n), y(n), z(n)
28 x(n) = ec1
29 y(n) = ec2
30 z(n) = ec3
31 end if
32
33 //Se continua integrando con el paso de integracion h
34 x(tk) = f(x(tk-1))(h^{q1}) - sum(C1(1:j)*x(v:n))
35 y(tk) = f(y(tk-1))(h^{q2}) - sum(C2(1:j)*y(v:n))
36 z(tk) = f(z(tk-1))(h^{q3}) - sum(C3(1:j)*z(v:n))
37
38 //Se procede a graficar los puntos obtenidos en los vectores donde no
     se establecio el mapa de Poincare, para este ejemplo se usaran los
     vectores G y H
39 plot(G, H)
```

3.5.2. Mapa de Poincaré del sistema con punto de equilibrio estable

Con el algoritmo anteriormente mencionado 3.5.1, se procede a calcular el mapa de Poincaré para el sistema con equilibrios estables (3.1), recordar que un sistema considerado caótico tiene múltiples puntos que pasan por el mapa de Poincaré, la figura 3.13 muestra el mapa obtenido. El punto a cruzar fue puesto en el valor 0.5 de la ecuación x.

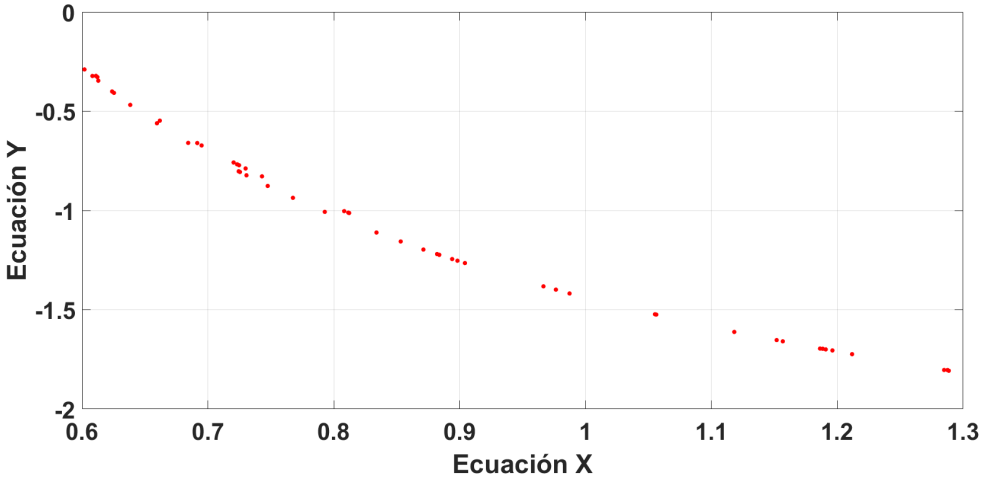


Figura 3.13: Mapa de Poincaré para el sistema con punto de equilibrio estable (3.1), al obtener múltiples puntos el sistema es considerado caótico.

3.5.3. Mapa de Poincaré del sistema con punto de equilibrio infinitos

De igual manera se auxilia del algoritmo de la sección 3.5.1 para obtener el mapa de Poincaré del sistema con puntos de equilibrio infinitos (3.7), para determinar si presenta un comportamiento caótico, la figura 3.14 muestra los resultados obtenidos. El valor del punto a cruzar fue puesto en -1.1 de la ecuación z.

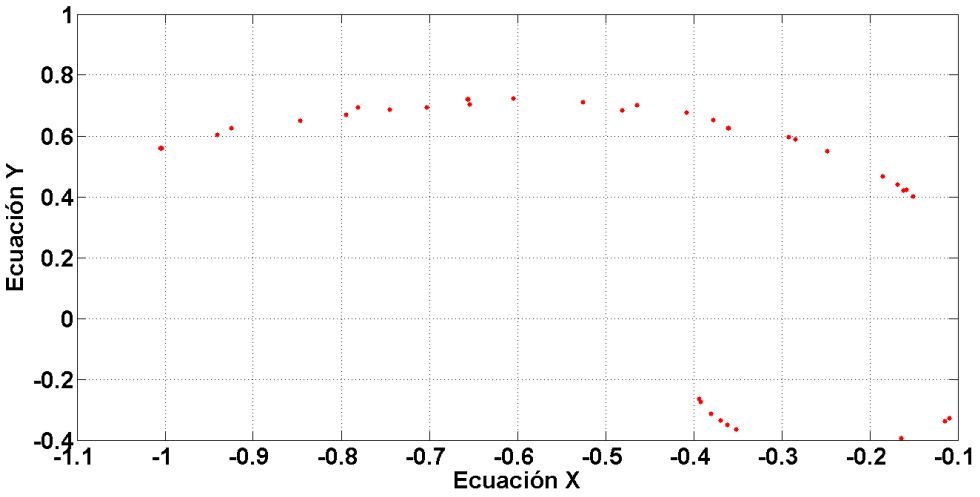


Figura 3.14: Mapa de Poincaré para el sistema con puntos de equilibrio infinitos (3.7), al obtener múltiples puntos el sistema es considerado caótico.

3.5.4. Mapa de Poincaré del sistema sin puntos de equilibrio

Para si sistema sin puntos de equilibrio (3.10) también se calculo su mapa de Poincaré el cual es mostrado en la figura 3.15. El punto de cruce fue considerado en el valor 1 de la ecuación z.

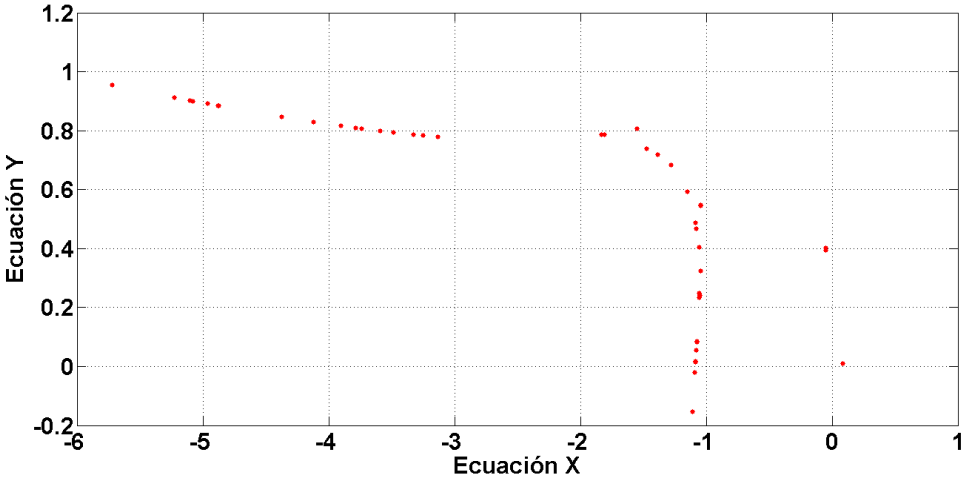


Figura 3.15: Mapa de Poincaré para el sistema sin puntos de equilibrio (3.10), al obtener múltiples puntos el sistema es considerado caótico.

3.5.5. Mapa de Poincaré del sistema con mega estabilidad

Para el sistema con mega estabilidad (3.13) también fue obtenido el mapa de Poincaré, para este caso que se tiene cuatro ecuaciones de igual manera se eligieron dos para graficar el mapa, en la figura 3.16 se muestra el mapa obtenido. El punto del mapa fue puesto en el valor 1 de la ecuación z.

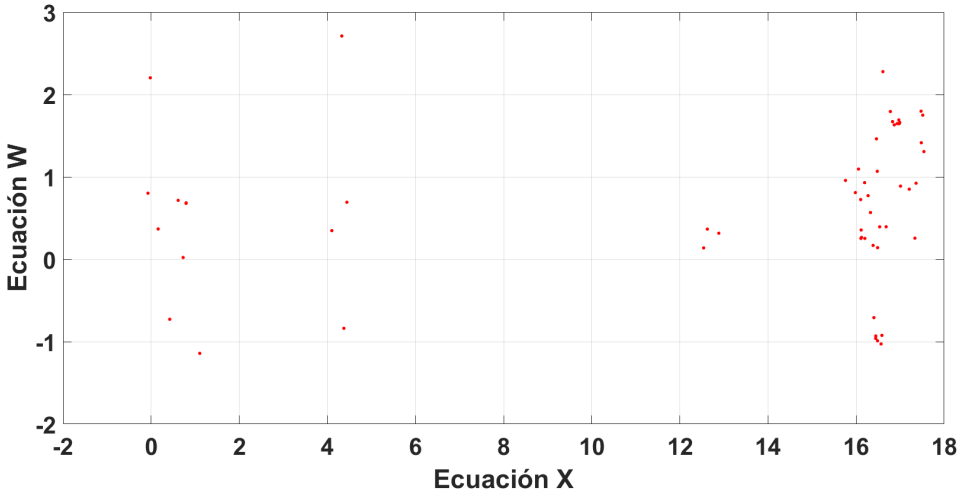


Figura 3.16: Mapa de Poincaré para el sistema con mega estabilidad (3.13), al obtener múltiples puntos el sistema es considerado caótico.

3.6. Diagramas de bifurcación

La estructura cualitativa del flujo de un sistema dinámico no lineal puede cambiar si se modifican los parámetros del mismo. En particular, pueden aparecer nuevos puntos de equilibrio u órbitas cerradas, o bien, desaparecer los que ya existen, o ver alteradas sus propiedades de estabilidad. Un sistema donde esto ocurre recibe el nombre de sistema estructuralmente inestable, o se dice equivalentemente, que una bifurcación ha ocurrido para ciertos valores de los parámetros [73].

Los diagramas de bifurcación tienen como función el ilustrar los comportamientos de bifurcación en el sistema cuando cambian sus parámetros. Cuando se cambia un parámetro en el sistema para un diagrama de bifurcación, otros parámetros deberían permanecer constantes. Basado en el mapa de Poincaré, los estados del sistema son obtenidos aplicando la sección de Poincaré para sistemas autónomos, para sistemas no autónomos se aplican pruebas estroboscópicas. El parámetro varía paso a paso, y los estados del sistema son observados en cada valor de este parámetro. En sistemas dinámicos donde múltiples atractores existen, diferentes condiciones iniciales deben ser probadas para cada valor del parámetro variable con esto, diferentes atractores pueden ser observados [74]. Los diagramas de bifurcaciones clasifican de manera condensada todos los comportamientos posibles de un sistema y las transiciones entre ellos (bifurcaciones) cuando cambian los parámetros del sistema. Los diagramas de bifurcaciones están compuestos por un número finito de regiones reparadas por los límites de bifurcación.

3.6.1. Algoritmo para la obtención del diagrama de bifurcación

Previamente se ha explicado en que consiste el diagrama de bifurcación, para calcularlo se realizó un código que lleva a cabo los pasos necesario. Para complementar el algoritmo y poder obtener el diagrama de bifurcación se utiliza la ecuación de la distancia euclidiana (3.18), la cual es tomada como la distancia del punto que se esta calculando al origen, dicha ecuación es mostrada a continuación:

$$V_q = \sqrt{(yt - 0) + (zt - 0)^2} \tag{3.18}$$

donde yt y zt son los valores calculados por las ecuaciones del sistema cuando cruza por el mapa de Poincaré sobre la ecuación x (cambia dependiendo de la ecuación que se elija). A continuación se muestra un pseudocódigo el cual es complementario a los algoritmos mostrados en las secciones 3.2 y 3.5.1, el diagrama de bifurcación tiene una cierta dependencia de este último algoritmo.

```
1  //Se realizara para un parametro elegido y en variacion de ciertos
     intervalos
2  for c = 1 con aumentos de 0.0001 hasta 2
3
4  //Se quita el transiente del sistema (se utilizan solo la mitad de
     las iteraciones)
5  if n > k/2 //Donde n es la iteracion actual y k el numero de
     iteraciones totales
6
7  //Se evalua el mapa de Poincare, como se hizo anteriormente
8  t(tk)/x(tk)=1/f(x(tk))*((-Deltax)^{q1}) - sum(C1(1:j)*x(v:n))
9  y(tk)/x(tk)=f(y(tk))/f(x(tk))*((-Deltax)^{q2}) - sum(C2(1:j)*y(v:n))
10 z(tk)/x(tk)=f(z(tk))/f(x(tk))*((-Deltax)^{q3}) - sum(C3(1:j)*z(v:n))
11
12 //Se crea una variable inicializada en 1 la cual servira para ir
     guardando valores, se calcula la distancia euclidiana con el punto
     (0,0) de las ecuaciones donde no esta el plano de Poincare
13 num = 1
14 Vq = sqrt((y(tk)/x(tk) - 0)^2 + (z(tk)/x(tk) - 0)^2)
15 Vr(num, w) = Vq
16
17 //Para las condiciones iniciales de las siguientes iteraciones del
     parametro se toman los ultimos valores calculados porlas ecuaciones
      o en caso de que esten muy alejados de las condiciones inciales,
     se toman las condiciones inciales originales del sistema
18 x(0) = x(k)  o  x(0) = 1  //Donde k es la ultima iteracion
19 y(0) = y(k)  o  y(0) = 2  //Donde k es la ultima iteracion
20 z(0) = z(k)  o  z(0) = -1 //Donde k es la ultima iteracion
21 end if
22
23 num = num + 1
24 end for
25
26 //Se graficaran los incrementos en el parametro contra los valores
     obtenidos de la distancia euclidiana, para este caso serian el
     parametro c y el vector Vr
27 plot(c, Vr)
```

3.6.2. Diagrama de bifurcación del sistema con punto de equilibrio estable

Basado en el algoritmo anterior 3.6.1 se procede a calcular el diagrama de bifurcación del sistema (3.1), este diagrama muestra el comportamiento del sistema conforma cambia su parámetro, que en este caso es el orden fraccionario. Si se observa una o dos lineas de puntos se considera que el sistema es periódico, si se ven múltiples lineas/puntos se considera que el sistema presenta comportamiento caótico. Los resultados obtenidos en para el sistema con puntos de equilibrio estable se muestran en la figura 3.17.

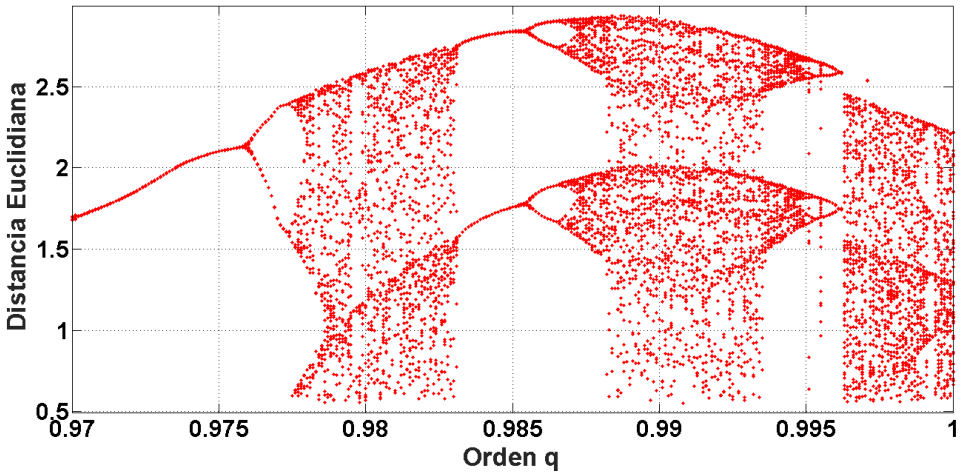


Figura 3.17: Diagrama de bifurcación para el sistema con puntos de equilibrio estable (3.1).

.

De acuerdo a los resultados obtenidos, el sistema presenta comportamiento caótico en los intervalos de $0.977 < q < 0.983$, $0.987 < q < 0.995$ y $0.996 < q < 1$.

3.6.3. Diagrama de bifurcación del sistema con puntos de equilibrio infinitos

Se procedió a calcular el diagrama de bifurcación del sistema con puntos de equilibrio infinitos (3.7), el cual es mostrado en la figura 3.18.

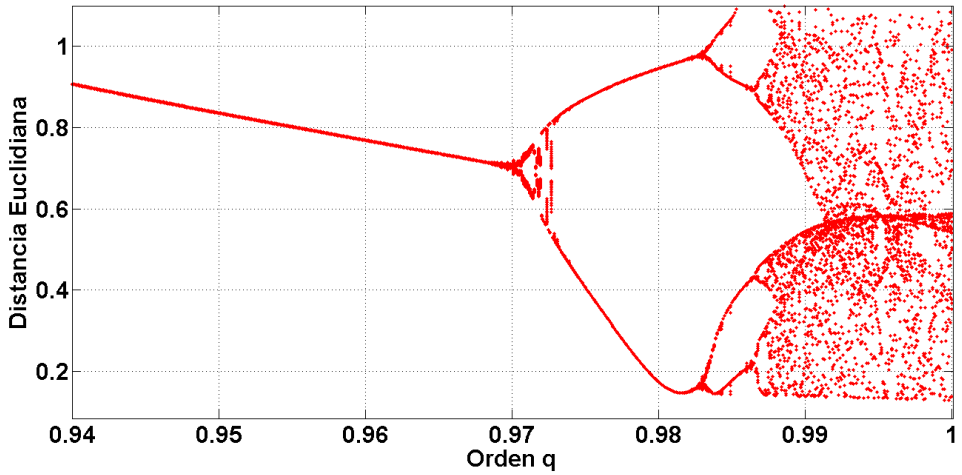


Figura 3.18: Diagrama de bifurcación para el sistema con puntos de equilibrio infinitos (3.7).

Como se puede apreciar en la figura anterior, el sistema presenta un comportamiento caótico en el intervalo de $0.986 < q < 1$.

3.6.4. Diagrama de bifurcación del sistema sin puntos de equilibrio

De igual manera se calculo el diagrama de bifurcación del sistema sin puntos de equilibrio (3.10), utilizando el algoritmo descrito en la sección 3.6.1. La figura 3.19 muestra el diagrama obtenido para dicho sistema.

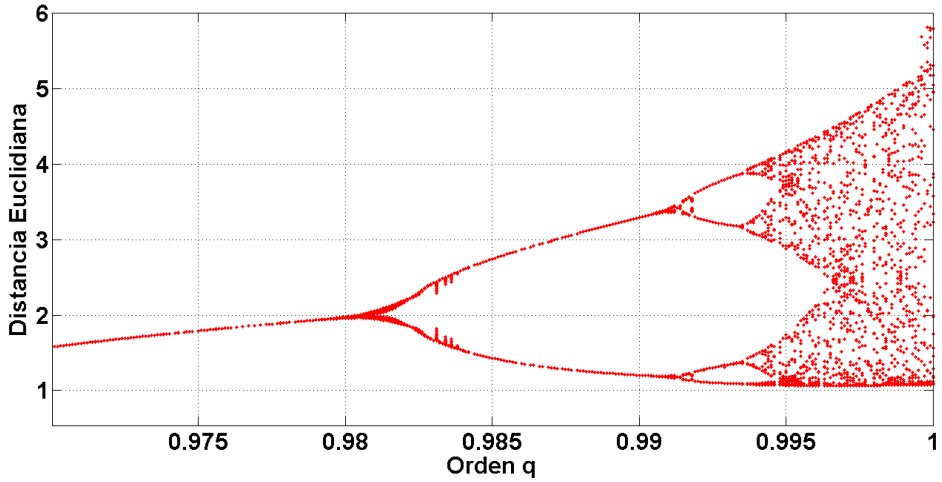


Figura 3.19: Diagrama de bifurcación para el sistema sin puntos de equilibrio (3.10).

Como se puede observar el la figura 3.19 este sistema cuenta con un intervalo pequeño en el cual presenta un comportamiento caótico, este intervalo esta definido entre $0.994 < q < 1$

3.6.5. Diagrama de bifurcación del sistema con mega estabilidad

Para el sistema con mega estabilidad (3.13) se calculo su diagrama de bifurcación con el algoritmo 3.6.1. En la figura 3.20 se puede observar el diagrama obtenido.

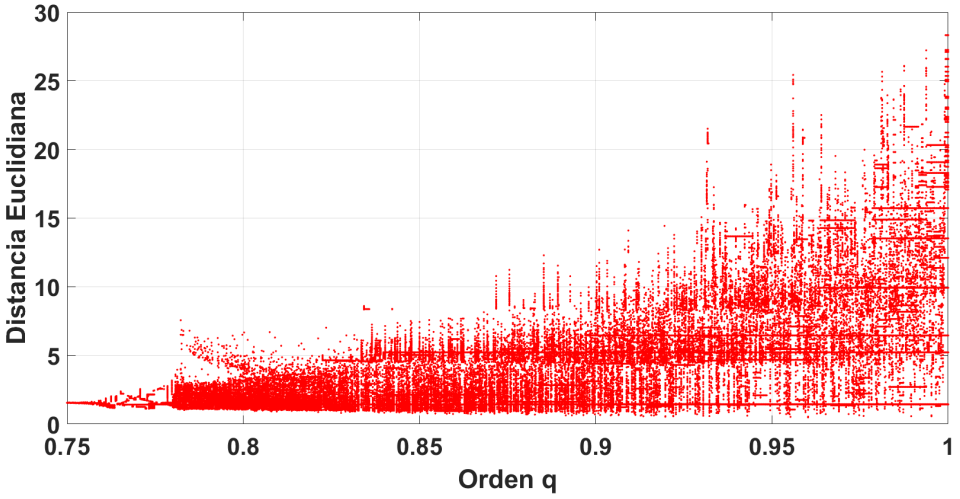


Figura 3.20: Diagrama de bifurcación para el sistema con mega estabilidad (3.13).

Para el caso de este sistema, el comportamiento caótico es presentado en un mayor intervalo, este intervalo esta definido por $0.775 < q < 1$, brindando una mejor posibilidad para trabajar con el orden fraccionario.

3.7. Exponente de Lyapunov

La principal caracterización de los sistemas caóticos son la dimensión fractal, la entropía de Kolmogorov-Sinai y el exponente de Lyapunov. Entre ellos, el exponente de Lyapunov mide la tasa promedio de divergencia o convergencia de órbitas que inician de puntos cercanos, además, proporciona una forma de determinar si el comportamiento de un sistema es caótico. Los exponentes de Lyapunov nos dan la descripción más característica de la presencia de un flujo determinista no periódico. Por lo tanto, los exponentes de Lyapunov 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 [68, 75]. El número de exponentes de Lyapunov es igual al número de variables de estado, y si al menos uno es positivo, esto es un indicador de caos.

Los exponentes de Lyapunov pueden ser utilizados para determinar la estabilidad de comportamientos cuasi-periódicos y caóticos, a continuación se presentan unas definiciones del exponente de Lyapunov.

Definición 3.7.1 *Sistema en tiempo continuo: Selecciona cualquier condición inicial* $x_0 \in \mathbb{R}^n$. *Los valores propios de* $\Phi_t(x_0)$ *son* $m_1(t), ..., m_n(t)$. *Los números de*

Lyapunov de x_0 son:

$$\lambda_i := \lim_{t\to\infty} \frac{1}{t} \ln |m_i(t)|, \quad i = 1, ..., n \tag{3.19}$$

siempre que exista el límite.

Definición 3.7.2 *Sistemas en tiempo discreto: Seleccione cualquier condición inicial x_0, y permita que $\{x_k\}_{k=0}^{\infty}$ sea la órbita correspondiente de un sistema en tiempo discreto de dimensión P. Los valores propios de $DP^k(x_0)$ son $m_1(k), ..., m_p(k)$. Los números de Lyapunov de x_0 son:*

$$m_i := \lim_{t\to\infty} |m_i(k)|^{1/k}, \quad i = 1, ..., p \tag{3.20}$$

siempre que exista el límite.

Definición 3.7.3 *Exponentes de Lyapunov de un punto de equilibrio: Siendo $\lambda_1, ..., \lambda_n$, los valores propios de $Df(x_{eq})$, y tomando la equivalencia de que $\Phi_t(x_{eq}) = e^{Df(x_{eq})t}$, se tiene que $m_i(t) = e^{\lambda_i t}$, por lo tanto:*

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \ln |e^{\lambda_i t}| = \lim_{t\to\infty} \frac{1}{t} Re[\lambda_i]t = Re[\lambda_i] \tag{3.21}$$

Por lo tanto, los exponentes de Lyapunov son iguales a la parte real de los valores propios en el punto de equilibrio. Ellos indican la tasa de contracción (si $\lambda_i < 0$) o expansión (si $\lambda_i > 0$) cerca del punto de equilibrio

El exponente de Lyapunov se denota como "λ", es útil para distinguir entre los distintos tipos de órbitas. Si el exponente de Lyapunov es negativo, la órbita atrae a un punto fijo estable u órbita periódica estable. Si el exponente de Lyapunov es igual a cero, la órbita es un punto fijo neutral (o neutralmente estable). Si el exponente de Lyapunov es positivo, la órbita es inestable y se define como caótica.

La tabla 3.1, muestra la dinámica de un atractor dependiendo de sus exponentes de Lyapunov, también se puede observar que esta dinámica dependerá del orden del sistema y del número de exponentes (el número de exponentes es igual al orden del sistema), para este caso de trabajo con un sistema de tercer orden por lo que se obtienen tres exponentes de Lyapunov y la dinámica de sus atractores podía ser Caos o Torus [76].

3.7.1. Algoritmo para calcular el exponente de Lyapunov

Para la obtención del exponente de Lyapunov, se realizaron algunas modificaciones a un programa en Matlab obtenido en [77], el cual está basado en el código de Wolf [75], este programa estaba en otro lenguaje pero se realizó para trabajar en Matlab. A continuación, se muestra un pseudocódigo de dicho programa.

```
1 //Se utiliza una funcion
2 function [Texp,Lexp] = Lyap_base(n,rhs_ext_fcn,fcn_integrator,tstart,
    stept,tend,ystart,ioutp);
```

Tabla 3.1: Tabla de atractores para exponentes de Lyapunov.

Dimensión del sistema	Dinámica del atractor	Exponente de Lyapunov
1	Punto fijo	-
2	Periódico	0 -
3	Caos	0 0 -
	Torus	+ 0 -
4	Hipertorus	+ 0 0 -
	Caos	+ 0 - -
	Hipercaos	+ + 0 -

```
3
4//Se especifican el numero de ecuaciones (n), el sistema que se
    utilizara (rhs_ext_fcn), el metodo de integracion a utilizar (
    fcn_integrator), el tiempo de inicio (tstart), el paso de
    integracion (stept), el tiempo final (tend), las condiciones
    iniciales (ystart) y el valor del parametro que se quiere modificar
     (ioutp).
5 n = 3
6 fcn_integrator = @FDE_PI12_PC //Metodo de integracion para sistemas
    de orden fraccionario (Predictor-Corrector)
7 tstart = 0
8 stept = 0.5
9 tend = 300
10 ystart = [0.1 0 0]
11 ioutp = 10
12
13 //Se obtienen el numero de pasos
14 nin = round((tend - tstart)/stept)
15
16 //Se introducen los valores que se utilizaran para el metodo de
    integracion, se obtendran dos salidas
17 [T,Y] = feval(fcn_integrator,q,rhs_ext_fcn,tstart,1,y,a);
18 q = 1 //Orden del sistema
19 a = 0.005 //Valor a modificar
20
21 //Se hace una reortogonalizacion del sistema
22
23 //Se grafican los valores obtenidos para cada una de las ecuaciones
24 str1=num2str(Lexp(nit,1));
25 str2=num2str(Lexp(nit,2));
26 str3=num2str(Lexp(nit,3));
27
28 //Se crea una funcion del sistema que se va a utilizar
29 function f=lorenz_ext(t,X)
30
31 //En esta funcion se introducel los parametros, asi como, las
    ecuaciones del sistema
32 sigma = 16
33 R = 45.92
34 beta =  4
35
36 x = X(1)
37 y = X(2)
```

```
38 z = X(3)
39
40 Y = [X(4), X(7), X(10)
41       X(5), X(8), X(11)
42       X(6), X(9), X(12)]
43
44 f = zeros(9,1)
45
46 f(1) = SIGMA*(y - x)
47 f(2) = -x*z + R*x - y
48 f(3) = x*y - BETA*z
49
50 Se linealiza el sistema, se obtiene la matriz jacobiana del sistema
51 Jac=[matriz jacobiana]
52
53 //Se obtiene una ecuacion variacional
54 f(4:12) = Jac*Y
```

3.7.2. Exponente de Lyapunov del sistema con punto de equilibrio estable

Para calcular el exponente de Lyapunov del sistema (3.1) se utiliza el código explicado en la sección 3.7.1, los resultados obtenidos son mostrados en la figura 3.21

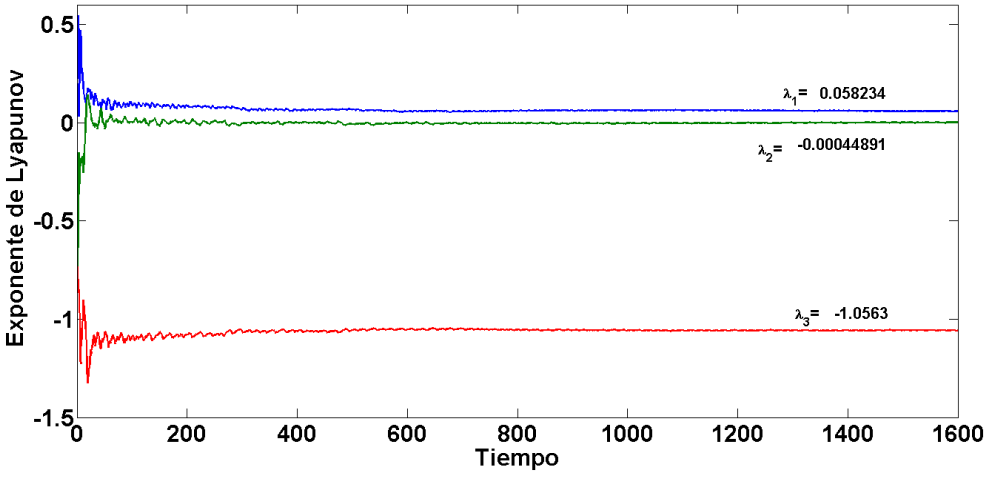


Figura 3.21: Exponente de Lyapunov para el sistema con puntos de equilibrio estable (3.1)

Como se puede apreciar en la figura 3.21 los exponentes son uno positivo, uno negativo y uno muy cercano a cero, con ello se puede concluir que el sistema tiene un comportamiento caótico.

3.7.3. Exponente de Lyapunov del sistema con puntos de equilibrio infinitos

Se calcularon los exponentes de Lyapunov con el código de la sección 3.7.1, para comprobar si el sistema con puntos de equilibrio infinitos (3.1) presenta un comportamiento caótico. La figura 3.22

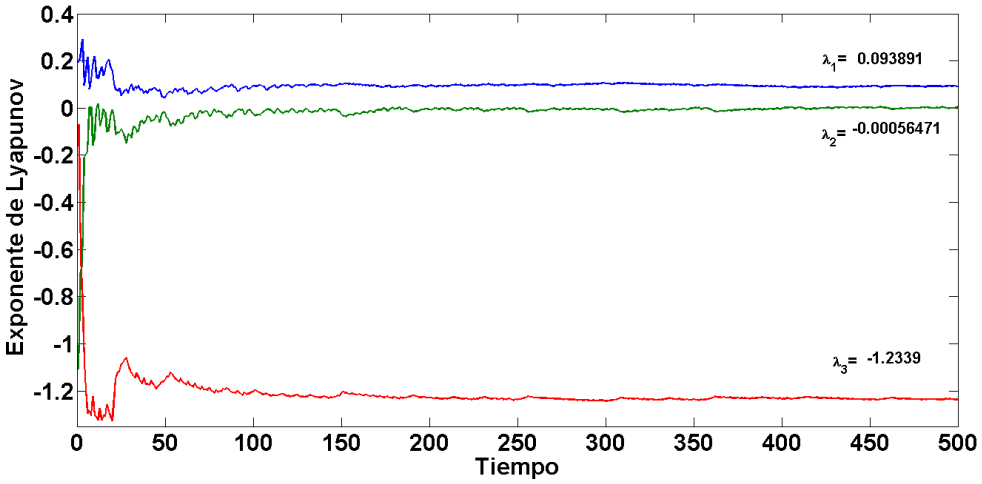


Figura 3.22: Exponente de Lyapunov para el sistema con puntos de equilibrio infinitos (3.7)

Como se puede observar en la figura 3.22 los exponentes de Lyapunov son uno positivo, uno negativo y uno muy cercano a cero, con ello se puede concluir que el sistema presenta un comportamiento caótico.

3.7.4. Exponente de Lyapunov del sistema sin punto de equilibrio

Se calcularon los exponentes de Lyapunov para el sistema sin puntos de equilibrio (3.10), los resultados son mostrados en la figura 3.23.

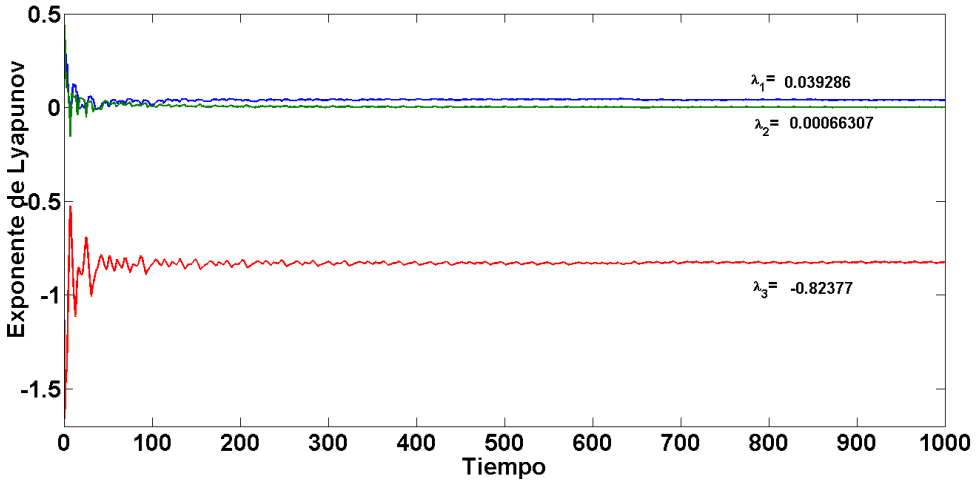


Figura 3.23: Exponente de Lyapunov para el sistema sin puntos de equilibrio (3.10)

La figura 3.23 muestra que los exponentes son uno positivo, uno negativo y uno muy cercano a cero, con ello se puede concluir que el sistema presenta un comportamiento caótico.

Capítulo 4

Instrumentación electrónica en sistema embebido ARM

El Internet de las cosas (IoT) se ha concebido como la próxima ola en la era de la tecnología cibernética, en la cual millones de dispositivos inteligentes (incluyendo varios sensores y actuadores) son conectados e integrados de manera inalámbrica a través de Internet. Este paradigma emergente creará e incrementará en gran número nuevas aplicaciones en muchos campos, incluidos el monitoreo ambiental, agricultura de precisión, redes inteligentes, ciudades inteligentes y sistemas de salud electrónica. Con ello se ha renovada la demanda de los sistemas embebidos [78].

Un llamado sistema embebido generalmente se compone de un grupo de dispositivos programables con algunos dispositivos de memoria y un conjunto de puertos de Entrada/Salida (E/S) de dispositivos periféricos. De hecho, un sistema embebido puede considerarse como un sistema integrado mediante la incorporación de algunas Unidades de Procesamiento central (CPU) con algunos subsistemas de memoria, puertos de E/S y tal vez varios dispositivos periféricos juntas en un solo chip semiconductor para obtener una unidad de control inteligente, llamada Unidad de microcontroladores (MCU), por lo tanto, un sistema embebido típico puede considerarse como un MCU. Los componentes más importantes involucrados en este MCU incluyen:

- CPU o procesador $ARM^{®}$
- Memoria (SRAM y Memoria Flash)
- Puertos de entrada y salida en paralelo (PIO)
- Algunos dispositivos internos (Temporizador/Contadores) o dispositivos periféricos (ADC y USB)

Diferentes sistemas embebidos o MCUs han sido desarrollados y construidos por diferentes vendedores en los últimos años. Una de las MCU más populares es la familia MCU $ARM^{®}$ $Cortex^{®}$-M. Este tipo de MCU proporciona multifunciones y capacidades de control, baja potencia de consumo, funcionalidad de procesamiento de señal de alta eficiencia, bajo costo y fácil de usar, estas son algunas de sus ventajas [79]. En particular, se utiliza una arquitectura ARM Soc debido a que son

utilizados ampliamente en dispositivos electrónicos como lo son: teléfonos inteligentes, tablets y otros dispositivos móviles. También permite un tamaño más pequeño, menor complejidad y menor consumo de energía. El aumento en la popularidad de los Sistemas Cibernéticos (CPS) y el Internet de las cosas (IoT), en gran escala, ha beneficiado enormemente a la Raspberry Pi, esta tarjeta es utilizada en múltiples tareas debido a las características que presenta. La Raspberry Pi es una tarjeta de bajo costo que soporta una variedad de sistemas operativos como lo son: BSD, Debian, Risc OS, Windows y algunas variaciones de Linux [78]. La Raspberry Pi 3 modelo B+ (figura 4.1), es el último de los productos lanzados de la gama Raspberry Pi 3, esta tarjeta incluye un microcontolador Broadcom BCM2837B0, Cortex-A53 (ARMv8) SoC, la cual tiene una mejora en cuanto rendimiento. Esta tarjeta también cuenta con algunos entornos de programación como lo son IDLE para Python y Scratch.

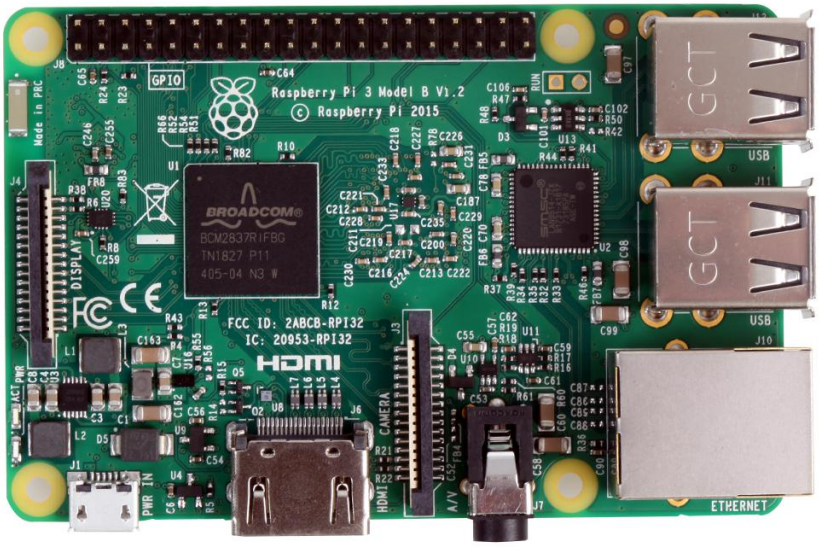


Figura 4.1: Imagen de la tarjeta Raspberry Pi 3 modelo B+

4.1. Implementación del método de Grünwald-Letnikov

El primer paso para obtener una implementación física en hardware de un sistema caóticos de orden fraccionario y por consecuencia su aplicación de ingeniería consiste en solucionar el sistema de ecuaciones mediante un método de discretización. Sin embargo, esta no es una tarea sencilla debido a que este tipo de sistemas requieren de algoritmos especializados, además del correcto manejo de la longitud de memoria del operador fraccionario. Por lo tanto, como se ha mencionado anteriormente, se ha propuesto utilizar el algoritmo numérico de Grünwald-Letnikov para solucionar los sistemas de orden fraccionario debido a que este algoritmo incorpora el concepto de memoria corta, la cual es adecuada para la implementación en hardware donde los recursos son limitados. Además, se propone utilizar el lenguaje de programación de Pyhton para obtener un algoritmo que pueda ser transferido a diversas plataformas digitales. Python es un lenguaje de programación de alto nivel de paradigmas múltiples enfocado en la generación de algoritmos de alto rendimiento.

Se ha decido utilizar Python como lenguaje de programación para llevar a cabo la síntesis de alto nivel para obtener una implementación física de sistemas caóticos en una placa comercial con plataforma SoC (System on a Chip) ARM (Advanced RISC Machine) ya que puede permitir aplicaciones de ingeniería basadas en caos fraccionario de bajo costo [27]. Se ha optado por la tarjeta Raspberry pi 3 modelo B+ para dicha implementación, la cual incluye un Soc ARM. A continuación, se presenta un pseudocódigo realizado para la implementación del algoritmo de Grünwald-Letnikov.

```
1  //Se mandan a llamar las librerias que se van a utilizar
2  import numpy as nm
3  import math
4  from decimal import getcontext
5  import RPi.GPIO as GPIO
6
7  //Se declaran las salidas de la Raspberry
8  GPIO.setmode(GPIO.BCM)
9  listax = [4,17,27,22,5,6,13,19]
10 listaz = [23,24,25,12,16,20,21,26]
11 GPIO.setup(listax,GPIO.OUT)
12 GPIO.setup(listaz,GPIO.OUT)
13
14 //Se crea una funcion para hacer la conversion de numero decimal a
    numero binario que sera utilizado para el invio de bits de la
    Raspberry, su parametro a utilizar es el numero a convertir
15 def dec_bin_out(numero)
16       num = abs(int((numero+4)*10))           //Offset para evitar
    numero nenegativo
17       bina = [0,0,0,0,0,0,0,0]
18       for i in range(0,8):
19               div = nim//2
20               res = num - div*2
21               bina[i] = res
22               num = div
23       sal_bina = bina[::-1]
24       return sal_bina
25
26 //Se crea una funcion para calcular los coeficientes binomiales, se
    introduce como parametro el orden fraccionario y la longitud de
    memoria
27 def coeficientes(q,lon):
28       Cp = 1
29       C = nm.zeros(lon)
30       for i in range(1,lon):
31               C[i-1] = (1 - (1 + q)/(i))*Cp
32               Cp = C[i-1]
33       return C
34
35 //Se crea una funcion para evaluar el atractor que se va utilizar
    donde solo se necesitan introducir como parametros los valor de las
     ecuaciones a evaluar
36 def atractor(xe,ye,ze,we):
37       xa = (ze + ze*2.5*math.sin(0.5*math.pi*ye))*(h**q)
38       ya = (1 - abs(xe))*(h**q)
39       za = (-xe - 0.1*ze*(-0.2 + (0.01*we**2)))*(h**q)
40       wa = (1*ze)*(h**q)
41       return (xa,ya,za,wa)
```

```
42
43 //Se crea una funcion para calcular la sumatoria del metodo de
    Grunwald-Letnikov utilizando como parametros los coeficientes
    binomiales, la ecuacion que se esta utilizando, el parametro j
    utilizado como limite de los coeficientes, el parametro v utilizado
     como inicio de la memoria e i que es el numero de iteracion
44 def suma(C,x,j,v,i):
45       rev = C[0:j]
46       rev1 = rev[::-1]
47       m = x[v:i]*rev1
48       s = m.sum(axis=0)
49       return s
50
51 //Se declaran los parametros del sistema que se van a utilizar
52 tk = 500                          //Tiempo de simulacion
53 h = 0.01                          //Paso de integracion
54 Lm = 50                           //Longitud de memoria
55 k = int(tk/h))                    //Numero de iteraciones
56 q = 0.95                          //Orden fraccionario
57 lon = int(Lm/h)                   //Valor en iteraciones de la longitud
    de memoria
58
59 //Se declaran los vectores a utilizar para almacenar los valores que
    se van obteniendo
60 x = nm.zeros(k+2)
61 y = nm.zeros(k+2)
62 z = nm.zeros(k+2)
63
64 //Se asignan las condiciones iniciales al primer valor del vector
65 x[0] = 1
66 y[0] = 1
67 z[0] = 1
68 w[0] = 1
69
70 //Se manda a llamar la funcion de los coeficientes binomiales
71 C1 = coeficientes(q,lon)
72
73 //Se crea un ciclo while para hacer el envio constante de los valores
     y un ciclo for para el calculo del sistema mediante el metodo de
    Grunwald-Letnikov completo, la seleccion de la memoria se introduce
     en una condicional if, los variables valx y valy son utilizadas
    para guardar los valores en binarios que son enviados a traves de
    la intruccion GPIO.output
74 while True:
75       for i in range(0,k):
76             if i>lon:
77                   v = i - lon
78                   j = lon
79             else:
80                   v = 0
81                   j = i
82             (x1p,y1p,z1p,w1p) = atractor(x[i],y[i],z[i],w[i])
83             x[i+1] = x1p - suma(C1,x,j,v,i)
84             y[i+1] = y1p - suma(C1,y,j,v,i)
85             z[i+1] = z1p - suma(C1,z,j,v,i)
86
87             valx = dec_bin_out(x[i+1])
88             valz = dec_bin_out(y[i+1])
```

```
89
90                  GPIO.output(listax,valx)
91                  GPIO.output(listaz,valz)
```

Una vez realizado el código que será utilizado para la programación, se procedió a realizar el armado del circuito ya que las salidas de la Raspberry son digitales y no podrían ser vistas en un osciloscopio, para ello se utilizó un convertidor digital-analógico (DAC0800) y un amplificador operacional (LM741), además de algunas resistencias de diferentes valores, este circuito es basado en la configuración que se muestra en la figura 4.2.

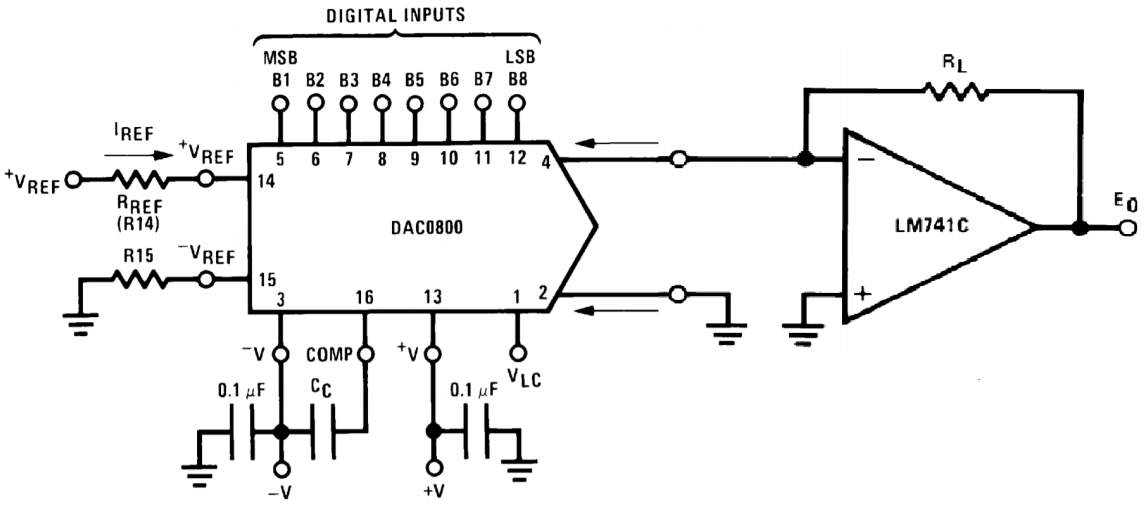


Figura 4.2: Diagrama para realizar la conversión digital-analógica de los valores proporcionados por la Raspberry.

Las conexiones de salida de los puertos GPIO de la Raspberry son conectadas a los pines del DAC0800 como se muestra en la figura 4.3, se ocupan 16 pines de la Raspberry ya que 8 pines serán utilizados para los valores de x y otros 8 para los valores de y, de esta manera se podrá graficar el atractor caótico calculado.

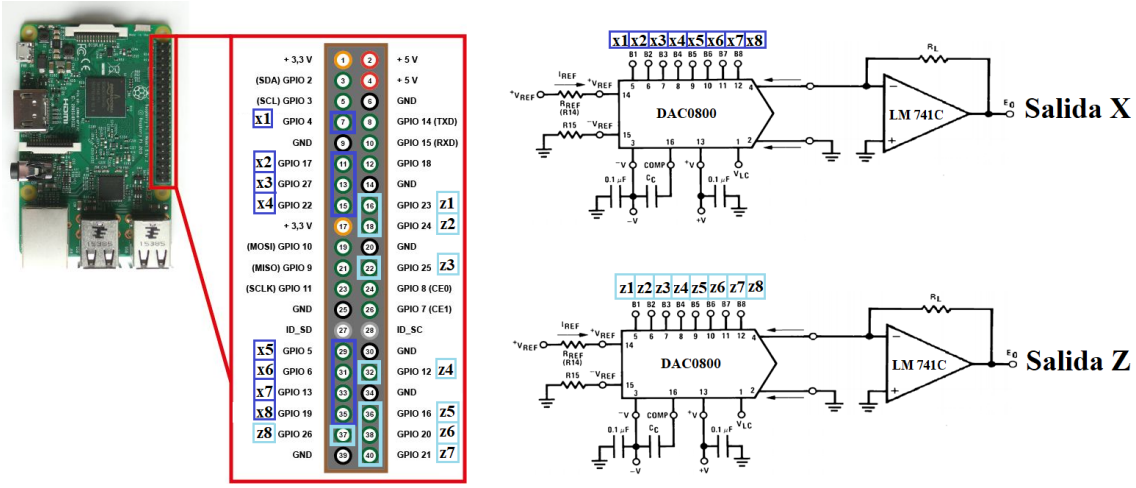


Figura 4.3: Conexiones de los puertos GPIO de la Raspberry, el convertidor DAC0800 y el amplificador LM741.

Teniendo los componentes necesarios, con ayuda de una fuente de voltaje, un monitor y un osciloscopio, se procedió a realizar todas las conexiones necesarias para llevar a cabo la implementación, en la figura 4.4.

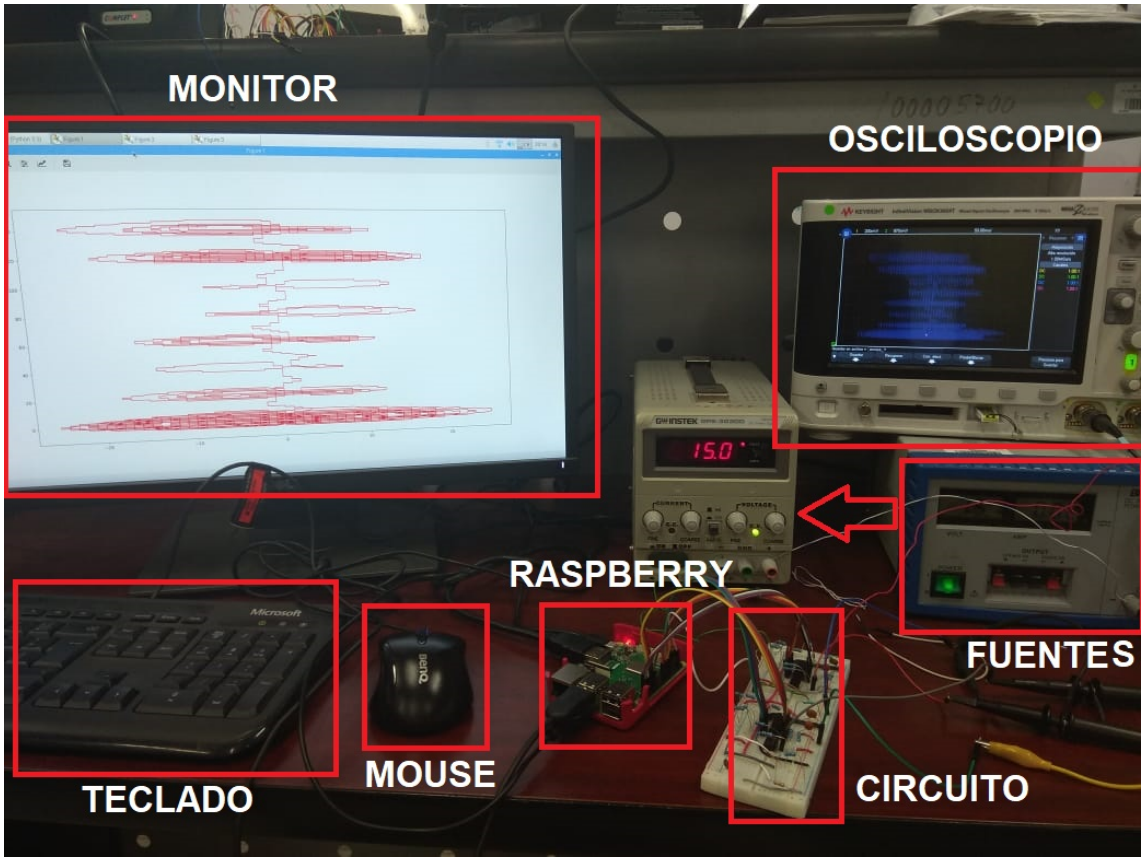
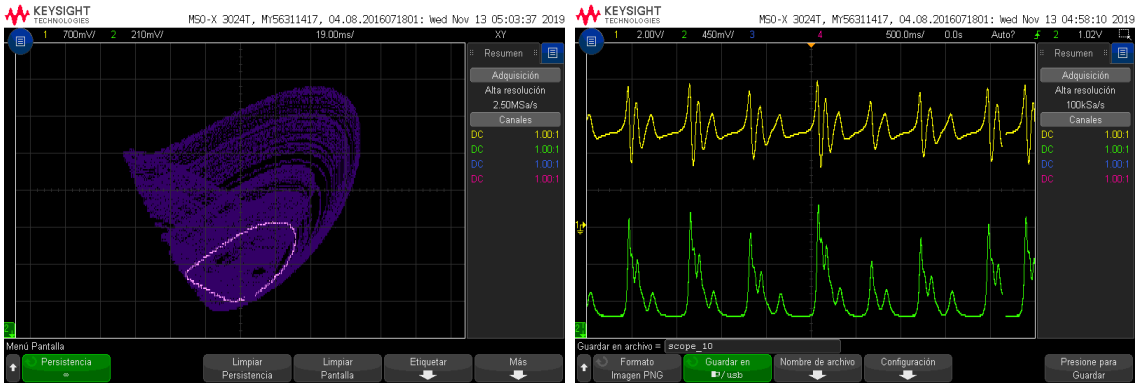


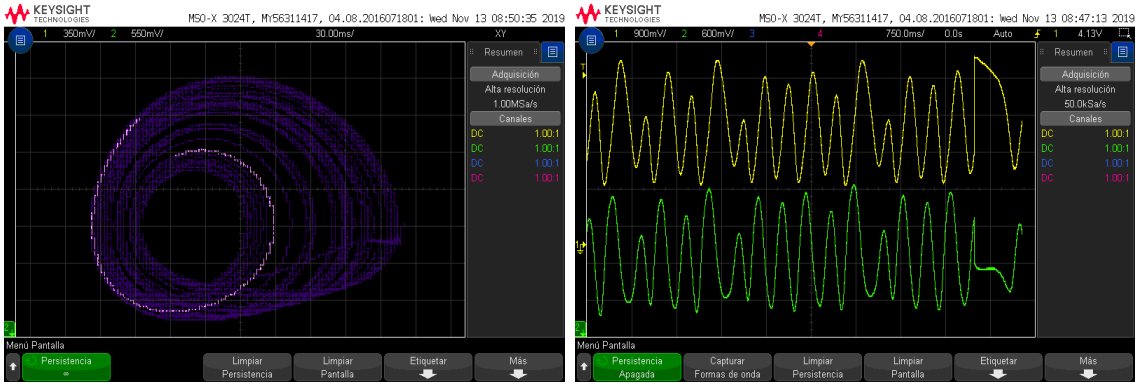
Figura 4.4: Conexiones realizadas para la implementación de sistemas caóticos en una Raspberry.

Habiendo comprobado el correcto funcionamiento de las conexiones realizadas se llevo a cabo la implementación de cada uno de los sistemas de las diferentes familias, en la figura 4.5 se muestra la implementación del sistema con punto de equilibrio estable, en la figura 4.6 se encuentra la implementación del sistema con puntos de equilibrio infinitos, en la figura 4.7 se observa la implementación del sistema sin puntos de equilibrio y por último en la figura 4.8 se puede apreciar la implementación del sistema con mega estabilidad.



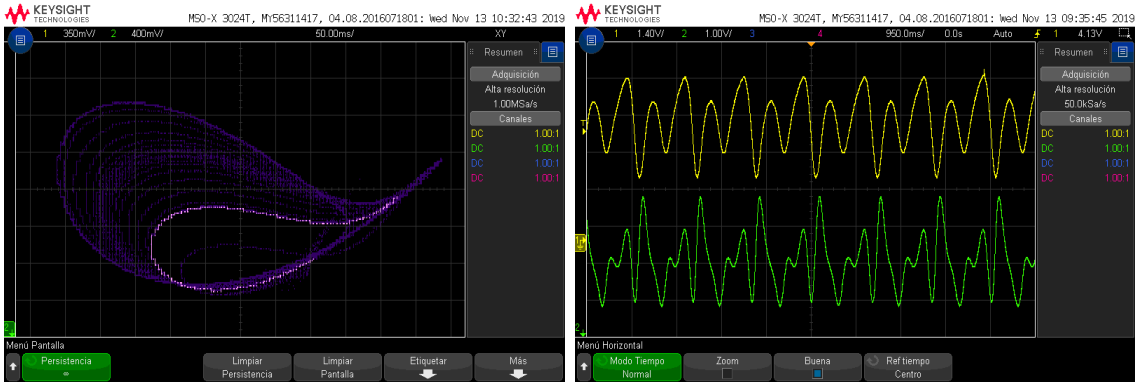
(a) (b)

Figura 4.5: Implementación del sistema con puntos de equilibrio estable (3.1) en Raspberry, (a) atractor caótico fase $x - y$, (b) series de tiempo (ecuación x en amarillo, ecuación y en verde).



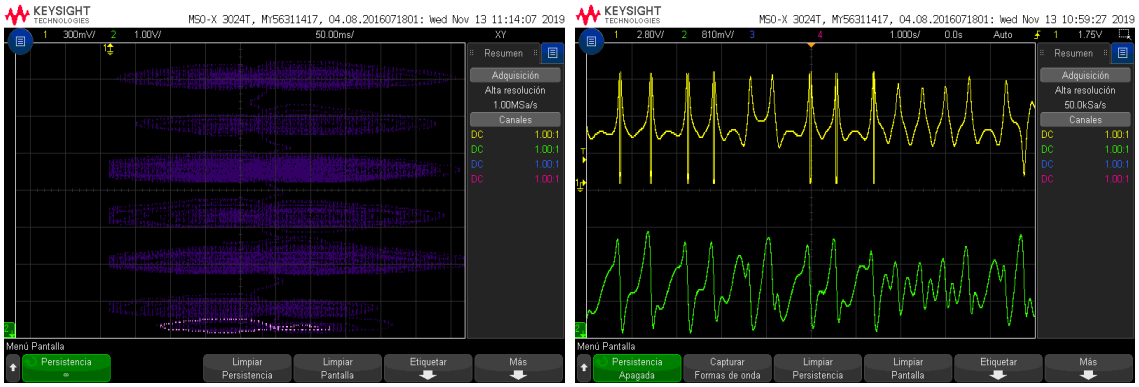
(a) (b)

Figura 4.6: Implementación del sistema con puntos de equilibrio infinitos (3.7) en Raspberry, (a) atractor caótico fase $x - y$, (b) series de tiempo (ecuación x en amarillo, ecuación y en verde).



(a) (b)

Figura 4.7: Implementación del sistema sin puntos de equilibrio (3.10) en Raspberry, (a) atractor caótico fase $y - z$, (b) series de tiempo (ecuación y en amarillo, ecuación z en verde).



(a) (b)

Figura 4.8: Implementación del sistema con mega estabilidad (3.13) en Raspberry, (a) atractor caótico fase $x-y$, (b) series de tiempo (ecuación x en amarillo, ecuación y en verde).

4.2. Implementación de generador de números aleatorios

A partir de haber estudiado la teoría de los generadores de números aleatorios se realizó un código el cual tomaría las series de tiempo de los sistemas caóticos con atractores ocultos y los utilizaría para realizar el generador de números aleatorios, para ello se realizaron varias pruebas de como obtener los bits necesarios para lograr una mayor aleatoriedad. En base a una configuración realizada para que se convirtieran un cierto número de valores después del punto decimal a binario, se obtenía que se generaban 44 valores binarios de los cuales se fueron realizando pruebas para observar cual generaba mayor aleatoriedad, entre las pruebas que se estuvieron realizando para obtener el generador de números aleatorios, se muestran las siguientes:

- Tomar los primeros diez valores generados.
- Tomar los diez valores generados del medio.
- Tomar los últimos diez valores generados.
- Tomar los primeros veinte valores generados.
- Tomar los últimos veinte valores generados.
- Tomar los veinte valores generados del medio.
- Tomar aleatoriamente diez, veinte o treinta valores.
- Tomar los diez primeros y los diez últimos valores.
- Tomar todos los bits generados.

De acuerdo a algunas pruebas hechas, se llegó a la conclusión de seleccionar los veinte bits generados del la parte media, ya que mostraron mayor aleatoriedad, para ello se dividía entre dos el número de bits generados, a partir de ahí se tomaban los diez valores anteriores y los diez posteriores. A continuación se describe un pequeño código el cual es complementario al visto en la sección 4.1.

```
1 //Se crea una funcion para hacer la conversion a decimal, esta
    funcion es diferente a la de salida bits de la Raspberry debido a
    que la salida de la raspberry ocupa un vector o dupla y para el
    generador se usan datos tipo string. Se utilizadn como parametros
    el numero a convertir y los datos de precision de decimales
2 def dec_bin(numero,dn,nb)
3       num1 = abs(int(numero*dn))
4       bina1 = ''
5       for i in range(0,nb):
6               div1 = num1//2
7               res1 = num1 - div1*2
8               bina1 = str(res1) + bina1
9               num1 = div1
10      return (bina1)
11
12 // Se hace la conversion a binario de los valores calculados de la
    ecuacion Z, se toma la longitud del vector, se obtiene la mitad del
     valor y se guardan los diez valores  anteriores y los diez
    posteriores al valor medio calculado, estos valores formaran el
    generador de numeros aleatorios.
13 zp1 = dec_bin(z[i+1])
14 p2z = len(zp1)//2
15 zp2 = zp1[p2z-10:p2z+10]
16 convz = zp2 + convz
17
18 // Se trunca el vector de valores binarios a un millon ya que son los
     valores que se utilizan para las pruebas NIST y se guardan en un
    archivo de texto
19 convz02 = convz[0:1000000]
20 f4 = open ('archivo711z.txt','w')
21 f4.write(convz02)
22 f4.close()
```

4.2.1. Aplicación de pruebas NIST a generador de números aleatorios

Habiendo realizado el código del generador de números aleatorios, se procedió a realizar las pruebas de la NIST (National Institute of Standards and Technology), para ello se utilizo el archivo archivo711z.txt creado en el código anteriormente explicado. El archivo creado se modifico para que almacenar un millón de bits eso se hizo con el fin de realizar mil pruebas en la NIST, cada millón de bits tiene unas condiciones iniciales diferentes del sistema para probar la aleatoriedad de los bits obtenidos. El procedimiento para realizar las pruebas de la NIST se describirá a continuación.

Primeramente se descarga el software de la NIST en el cual están contenidas

las pruebas que se van a evaluar [1]. El archivo que se descarga lleva el nombre de sts-2.1.2, dentro de esa carpeta hay otra con el mismo nombre, dentro de esta segunda carpeta hay otras 6 carpetas (data, experiments, include, obj, src, templates) y un archivo (makefile). Para instalar las pruebas se abre una terminal o consola de MAC, se selecciona la ruta donde se guardo la carpeta descargada y posteriormente se manda a llamar el comando "make" para comenzar la instalación y al terminar se creará un ejecutable llamado "assess.c" en la carpeta sts-2.1.2.

Para evaluar las pruebas se abre una terminal, se ingresa la ruta de la carpeta sts-2.1.2 y se manda a llamar el archivo creado con la instalación (./ assess) y se agrega el número de bits que contiene el vector (un millón para evaluar todas las pruebas). Esto desplegará diez opciones, nueve de ellas son generadores que trae por default el software, para ingresar un archivo nuevo se selecciona la opción del valor cero la cual es para introducir una ruta donde se encuentra el archivo que se va a evaluar, en la figura 4.9 se puede observar lo antes descrito.

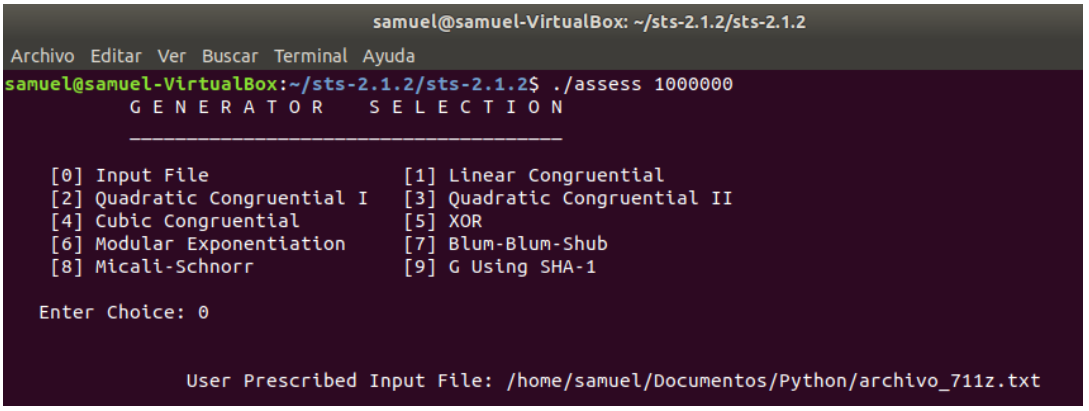


Figura 4.9: Introducción del archivo a evaluar con las pruebas NIST.

Posteriormente se elegirá entre las pruebas que se desean aplicar, hay dos opciones para elegir, una de las opciones es ingresar el número cero y con ello se precedería a elegir las pruebas que se desean aplicar. La segunda opción es ingresar el número uno, esta opción da pauta a aplicar las quince pruebas. En la figura 4.10 se puede apreciar la explicación y los nombres de las pruebas.

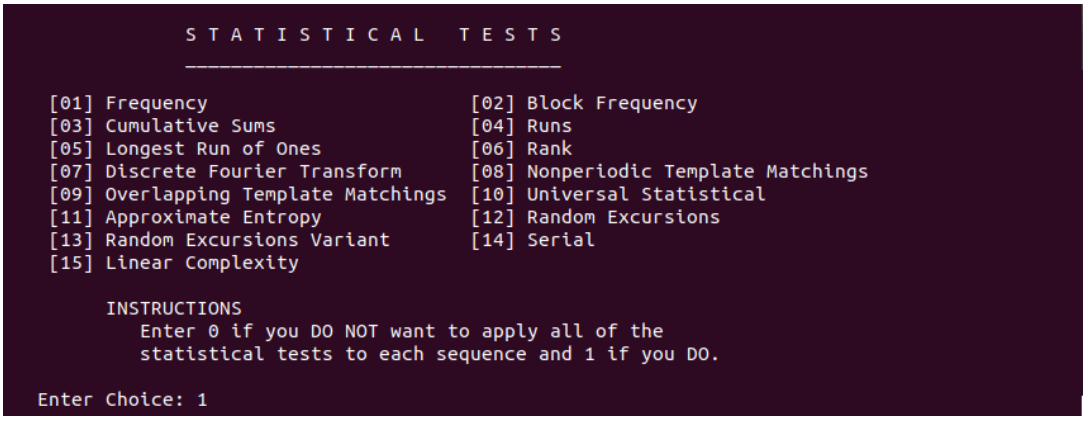


Figura 4.10: Selección de las pruebas que se desean evaluar.

[1] https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software

El siguiente paso es la configuración de algunos parámetros relacionados con las pruebas que se evaluarán, en caso de que no se desee realizar ninguna modificación se presiona el número cero para continuar con el siguiente paso. El siguiente paso es ingresar cuantos vectores del valor inicial (un millón) contiene el archivo seleccionado. Posteriormente se elige el formato del archivo que puede ser en formato ASCII o en formato Binario, para estas pruebas se selecciona el primer formato. En la figura 4.11 se pueden observar los pasos realizados.

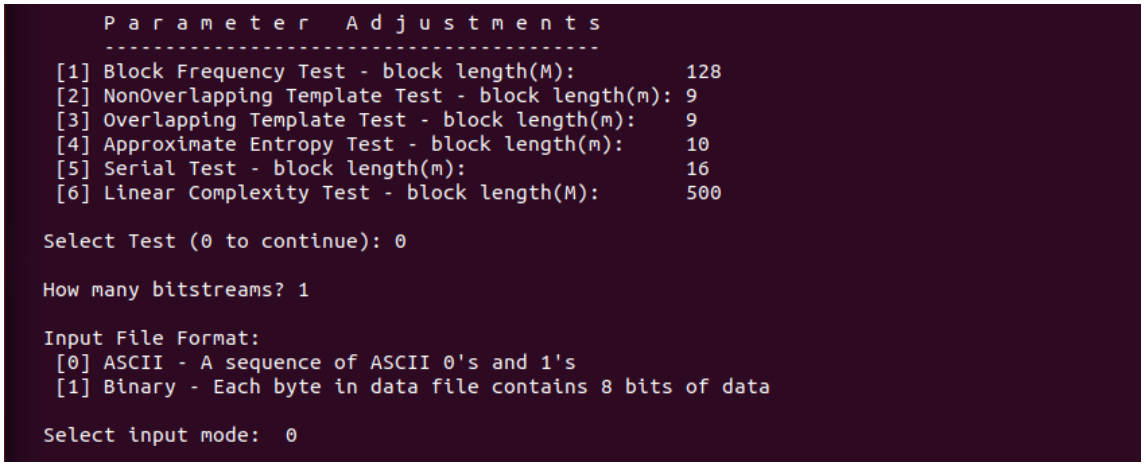


Figura 4.11: Selección de parámetros, número de vectores y formato del archivo.

En el momento de realizar las pruebas se pudieron observar dos resultados, uno de ellos en el cual aparece la frase igamc: UNDERFLOW, que pareciera ser debido a que el vector de bits introducido no cumple con las especificaciones necesarias para evaluar adecuadamente las pruebas en la figura 4.12 se puede apreciar el resultado comentado. El otro caso obtenido fue cuando simplemente se obtuvo la frase de que las pruebas habían terminado de ser evaluadas sin mostrar la frase anteriormente mencionada, esto se puede observar en la figura 4.13.

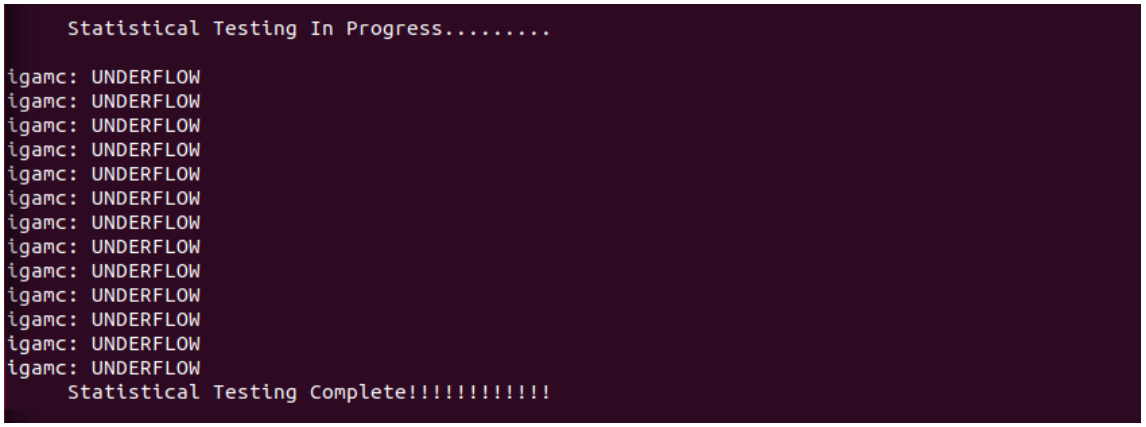


Figura 4.12: Salida mostrada al no cumplir con las especificaciones para evaluar el vector de bits.

Para probar la fiabilidad del generador de números aleatorios se procedió a evaluar los mil millones de bits para cada uno de los sistemas pertenecientes a las diferentes familias. Para las pruebas de la NIST existe un mínimo y un máximo

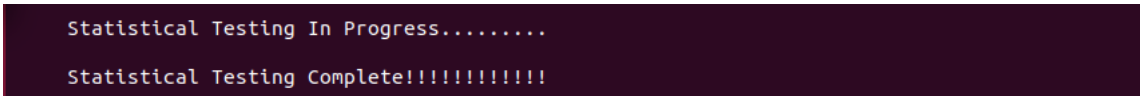


Figura 4.13: Salida mostrada cuando se cumple con las especificaciones para evaluar el vector de bits.

passing ratio (PR, relación de paso), estos valores son calculados con la ecuación siguiente [80]:

$$PR = 0.99 \pm 3\sqrt{0.99\frac{0.01}{N}} \tag{4.1}$$

donde N es el número de muestras/entradas. La ecuación (4.1) proporciona el valor mínimo y el valor máximo del rango en el que deben están los valores de proporción obtenidos de las pruebas NIST.

Como se mencionó, este es un estándar industrial el cual ya esta establecido, por lo que en su manual menciona que para considerar adecuado un generador de números aleatorios pensado para aplicaciones de criptografía deben ser evaluadas mil pruebas de un millón de bits, lo cual da un total de un gygabyte de bits (1GB).

Sustituyendo el valor de N en la ecuación (4.1) se obtiene el passing ratio mínimo $0.98056 \approx 0.981$ y el passing ratio máximo $0.99944 \approx 0.999$, por ello, cada una de las quince pruebas de la NIST deberá estar entre este rango de valores para poder ser considerado un buen generador para aplicaciones de cifrado de datos.

A continuación, se muestran tablas evaluando los 4 sistemas con que se han estado trabajando ((3.1), (3.7), (3.10), (3.13)), en cada tabla se muestran los resultados proporcionado para cada una de las pruebas de la NIST, así como su passing ratio promedio obtenido entre los mil millones de bits calculados con el generador de números aleatorios.

Tabla 4.1: Tabla de resultados para evaluación de pruebas NIST, sistema con puntos de equilibrio estable (3.1).

Nombre de la prueba	Valor P	Proporción	Passing ratio	Resultado
Frecuency	0.000000	1/1000	0.001	No aprobado
Block frecuency	0.000000	1/1000	0.001	No aprobado
Cumulative sums	0.000000	1/1000	0.001	No aprobado
Runs	0.000000	0/1000	0.000	No aprobado
Longest run	0.000000	1/1000	0.001	No aprobado
Rank	0.000000	36/1000	0.036	No aprobado
FFT	0.000000	0/1000	0.000	No aprobado
Nonoverlapping template	0.000000	47/1000	0.047	No aprobado
Overlapping template	0.000000	1/1000	0.001	No aprobado
Universal	0.000000	0/1000	0.000	No aprobado
Approximate entropyy	0.000000	0/1000	0.000	No aprobado
Random excursions	0.178234	410/420	0.976	Aprobado
Random excursions variant	0.394552	419/420	0.997	Aprobado
Serial	0.000000	1/1000	0.001	No aprobado
Linear complexity	0.125456	999/1000	0.999	Aprobado

Tabla 4.2: Tabla de resultados para evaluación de pruebas NIST, sistema con puntos de equilibrio infinitos (3.7).

Nombre de la prueba	Valor P	Proporción	Passing ratio	Resultado
Frecuency	0.296834	988/1000	0.988	Aprobado
Block frecuency	0.154398	992/1000	0.992	Aprobado
Cumulative sums	0.831671	990/1000	0.990	Aprobado
Runs	0.394195	990/1000	0.990	Aprobado
Longest run	0.200115	989/1000	0.989	Aprobado
Rank	0.935716	989/1000	0.989	Aprobado
FFT	0.070212	986/1000	0.986	Aprobado
Nonoverlapping template	0.765632	988/1000	0.988	Aprobado
Overlapping template	0.763677	985/1000	0.985	Aprobado
Universal	0.647530	982/1000	0.982	Aprobado
Approximate entropyy	0.245490	988/1000	0.988	Aprobado
Random excursions	0.578556	614/623	0.985	Aprobado
Random excursions variant	0.139964	617/623	0.990	Aprobado
Serial	0.564639	986/1000	0.986	Aprobado
Linear complexity	0.664168	986/1000	0.986	Aprobado

Tabla 4.3: Tabla de resultados para evaluación de pruebas NIST, sistema sin puntos de equilibrio (3.10).

Nombre de la prueba	Valor P	Proporción	Passing ratio	Resultado
Frecuency	0.000000	1/1000	0.001	No aprobado
Block frecuency	0.000000	0/1000	0.000	No aprobado
Cumulative sums	0.000000	1/1000	0.001	No aprobado
Runs	0.000000	0/1000	0.000	No aprobado
Longest run	0.000000	1/1000	0.001	No aprobado
Rank	0.329850	992/1000	0.992	Aprobado
FFT	0.000000	0/1000	0.000	No aprobado
Nonoverlapping template	0.000000	96/1000	0.096	No aprobado
Overlapping template	0.000000	0/1000	0.000	No aprobado
Universal	0.000000	1/1000	0.001	No aprobado
Approximate entropyy	0.000000	0/1000	0.000	No aprobado
Random excursions	0.000341	484/504	0.960	No aprobado
Random excursions variant	0.000196	480/420	0.952	No aprobado
Serial	0.000000	1/1000	0.001	No aprobado
Linear complexity	0.000000	1/1000	0.001	No aprobado

Tabla 4.4: Tabla de resultados para evaluación de pruebas NIST, sistema con mega estabilidad (3.13).

Nombre de la prueba	Valor P	Proporción	Passing ratio	Resultado
Frecuency	0.366918	992/1000	0.992	Aprobado
Block frecuency	0.014550	983/1000	0.983	Aprobado
Cumulative sums	0.888356	992/1000	0.992	Aprobado
Runs	0.486588	990/1000	0.990	Aprobado
Longest run	0.205531	987/1000	0.987	Aprobado
Rank	0.901959	995/1000	0.995	Aprobado
FFT	0.228367	991/1000	0.991	Aprobado
Nonoverlapping template	0.583145	991/1000	0.991	Aprobado
Overlapping template	0.672470	987/1000	0.987	Aprobado
Universal	0.711601	987/1000	0.987	Aprobado
Approximate entropyy	0.739918	993/1000	0.993	Aprobado
Random excursions	0.853234	625/639	0.978	Aprobado
Random excursions variant	0.417019	633/639	0.990	Aprobado
Serial	0.445836	990/1000	0.990	Aprobado
Linear complexity	0.387264	992/1000	0.992	Aprobado

Para poder determinar que una prueba ha sido satisfactoria se observa el valor P (P-value), si este valor es mayor a 0.01 se dice que esa prueba es aprobada, de lo contrario no aprueba. La proporción nos permite ver el número de pruebas que han sido aprobadas, de la proporción se obtiene el passing ratio y este es el que nos indica el porcentaje de pruebas que pasan, este porcentaje debe estar dentro del calculado con la ecuación (4.1) de lo contrario se considera que esa prueba tampoco es aprobatoria y no cuenta con la suficiente aleatoriedad. Para el caso de las pruebas de random excursions y random excursions variant la el passing ratio cambia un

poco debido a que esta prueba excluye algunos valores, ya que así esta diseñada. Cuando se evalúan los vectores de bits en el archivo de resultados aparece cuantas pruebas tienes que pasar para considerarse aprobatoria.

Como se puede observar en las tablas 4.1 y 4.3, el passing ratio de prácticamente todas las pruebas esta en un rango inferior al que anteriormente fue calculado con la ecuación (4.1), lo cual indica que esos números generados no tiene aleatoriedad suficiente para poder ser utilizados en aplicaciones de cifrado de datos. De las tablas 4.2 y 4.4 podemos observar que tanto el passing ratio como el valor P están dentro de los valores establecidos para considerarse aprobatorios y con buena aleatoriedad. Estos resultados de los sistemas (3.7) y (3.13) son satisfactorios y aptos para ser utilizados en aplicaciones de cifrado de datos.

4.3. Aplicación para cifrado de datos

El Estándar de Cifrado Avanzado (AES) el cual es un algoritmo criptográfico utilizado para proteger datos electrónicos. Es un cifrado de bloque simétrico que puede cifrar y descifrar información. El cifrado convierte los datos a una forma ininteligible llamada texto cifrado. El descifrado convierte los datos nuevamente en su forma original llamada texto sin formato. AES tiene una longitud de clave de cifrado de 128, 192 y 256 bits, que puede cifrar y descifrar datos en bloques de 128 bits.

El AES es ampliamente utilizado en protocolos seguros de transferencia de archivos como FTPS, HTTPS (que tenemos en este sitio web), en la protección WPA2 de las redes WiFi como lo son el estándar SSH o IPsec. Frecuentemente se usan los métodos AES también en la telefonía a través del internet (Voice over IP, VoIP) con el fin de asegurar los datos de señalización o también los datos útiles. AES también se han incluido en el desarrollo de aplicaciones de comunicaciones como: Servicio de Seguridad: Secure Socked Layer (SSL), para el intercambio de registros, Servicio de Seguridad (Secure Electronic Transaction por sus siglas en inglés SET) para servicios de pagos electrónicos creado por VISA y Master Card y Servicio de Seguridad (Private Enhanced Mail por siglas en inglés PEM), entre otros.

Algunos dispositivos conectados a través del Internet de las cosas (IoT) como lo son: PC, celulares, tablets, electrodomésticos de casa inteligentes (refrigerador, foco, apagador, etc.), entre otros, ocupan el cifrado AES para hacer envió de información, esto debido a que en ocasiones se envia información sensible como lo es: números de tarjetas de crédito/débito, operaciones bancarias, datos personales, contraseñas, entre otras.

El AES cuenta con dos métodos para llevar a cabo el cifrado de información, el método de Substitución-Permutación y el método Feistel Network, en la figura 4.14 se pueden observar estos métodos.

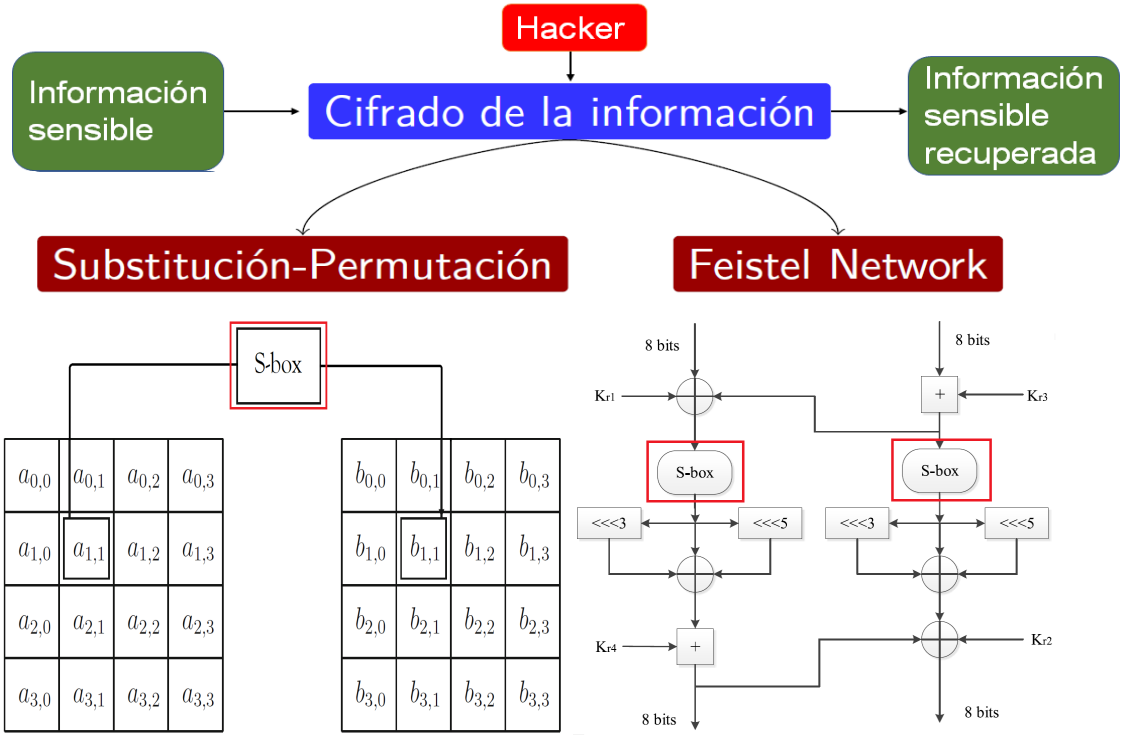


Figura 4.14: Metodologías para cifrado con AES.

Como se observa en la figura 4.14 ambas metodologías cuentan con una S-box, es por ello que tiene gran relevancia el enfocar la aplicación de este trabajo en estas S-box.

Las cajas de sustitución (S-box) es el único componente no lineal para proporcionar la confusión necesaria para garantizar la seguridad del cifrado de bloque. La fuerza criptográfica de una S-box tiene un impacto directo en la seguridad de los criptosistemas. La S-box también es conocida como la unidad más básica con función de codificación en algoritmos de cifrado de bloque. Una buena S-box puede hacer que el algoritmo de cifrado tenga una mayor seguridad y una mejor capacidad para resistir los ataques, las cajas de sustitución son utilizadas en la mayoría de los métodos criptográficos.

Para la creación de S-box se encontraron dos métodos diferentes los cuales serán implementados para los dos sistemas que cumplieron con las pruebas de la NIST, ya que con ello aseguramos que las S-box proporcionen una buena aleatoriedad cada vez que es creada. La primera metodología mencionada en [1] por los autores Liu-Yang-Liu, será descrita a continuación, los pasos son los siguientes:

1. Seleccionar los parámetros del sistema caótico.
2. Ingresar los parámetros de iteración y el paso de integración.
3. Se itera el sistema caótico con las condiciones iniciales.
4. Se tomará el número de valores D (donde $D = SnxSm$, $Sn = filas$, $Sm = columnasdelaS-box$), se seleccionará un valor de D por cada longitud $L(io+DxL < n$, n es el número de iteraciones, $io \epsilon N+)$ los valores D serán almacenados en un vector $w(i)$.

5. Se ordenan los valores del vector $w(i)$ de menor a mayor y se asignan en un nuevo vector llamado $w(i)$. Se crea un vector llamado o(i) el cual denota el índice de $w'(i)$ en la secuencia original $w(i)$.

6. Como resultado, la S-box es obtenida poniendo los elementos de la secuencia $o(i)$ en una matriz de tamaño $SnxSm$.

7. En caso de ser necesarias más S-box, se vuelve a realizar el proceso desde el paso 2 al paso 6 ajustando los parámetros,

Partiendo del códogo de la sección 4.1se realiza un comlemento para poder llevar el adecuado funcionamiento de las S-box. A continuacion, se presenta el pseudocódigo realizado.

```
1  //Se agrega la libreria para funciones aleatorias
2  from random import randint
3
4  //Se agregan los parametros necesarios para la Sbox, ademas de crear
     unos vectores para almacenar los valores
5  D = 256
6  L = 190
7  io = 1
8  Sm = 16
9  Sn = 16
10 wi = []
11 wip = []
12 wio = []
13
14 //Se selecciona un valor aleatorio entre 0 y L, posteriormente se
     calculan los valores del atractor
15 wx = randint(0,L)
16 for i in range(0,k+1)
17                 (x1p,y1p,z1p,w1p) = atractor(x[i],y[i],z[i],w[i])
18                 x[i+1] = x1p - suma(C1,x,j,v,i)
19                 y[i+1] = y1p - suma(C1,y,j,v,i)
20                 z[i+1] = z1p - suma(C1,z,j,v,i)
21
22 //Una vez calculados los valores del sistema se precede a tomar el
     valor de la ecuacion con el indice seleccionado (wx) y se agraga la
      distancia L para tomar el siguiente, estos valores se almancenan
     en el vector wi.
23 while io != D+1:
24       wi.append(y[wx])
25       wx = L + wx
26       io = io + 1
27
28 //Se ordenan los valores del vector wi de menor a mayor y se
     almacenan en el vector wip
29 wip = sorted(wi)
30
31 //Se asignan los indices del vector wi a los valores del vector wip y
      se almacenan en el vector wio, para al final hacer un arreglo en
     una matriz de SnxSm
32 for i in range(0,D):
33       wio.append(wi.inden(wip[i]))
34 sbox = nm.array(wio).reshape(Sm,Sn)
```

La segunda metodología descrita en [2] de los autores Lai-Akgul-Li-Xu-Cavusoglu, es descrita con los siguientes pasos:

1. Se introducen los parámetros y condiciones iniciales del sistema.
2. Se define un intervalo de integración para iterar el sistema y se crea un vector para almacenar los valores llamado Sbox.
3. Se resuelve el sistema con un método de integración y se seleccionan dos ecuaciones de las cuales serán tomados valores.
4. Se convierten a binario los valores de las ecuaciones seleccionadas.
5. Se toman los 8-bits menos significativos y se aplica una XOR.
6. Se convierte de binario a decimal el valor obtenido.
7. Si el valor es igual a uno anteriormente calculado se deshecha y se sigue calculando, si el valor es diferente se agrega al vector de la Sbox.
8. Se hace el arreglo del vector en forma de matriz.

Para la segunda forma de realizar una Sbox descrita en los pasos anteriores también se creo un complemento al código de la sección 4.1, a continuación se muestra el pseudocódigo de esta segunda manera de crear Sbox.

```
// Se agrega el paso de integracion con el que se iterara, el numero de columnas y el numero de filas de la matriz de la S-box
h1 = 000001
Sm = 16
Sn = 16

// Se crea una funcion que realizara el comportamiento de una compuerta XOR
def XOR_func(a,b):
        dx = ''
        for j in range(0, 8):
                if a[j] == b[j]:
                        dl =  str(0)
                else:
                        dl =  str(1)
                dx = dx + dl
                return dx

// Se inicial unas variables y se crea un vector para almacenar los valores de la S-box, de igual manera se crea una funcion la cual se encargara de realizar la creacion de la S-box
i = 0
j = 0
sbox = []
while j < 256):
        (xp,yp,zp,wp) = atractor(x[i],y[i],z[i],w[i])
        x[i+1] = xp - suma(C1,x,i+1)
        y[i+1] = yp - suma(C2,y,i+1)
        z[i+1] = zp - suma(C3,z,i+1)
        w[i+1] = wp - suma(C4,w,i+1)
```

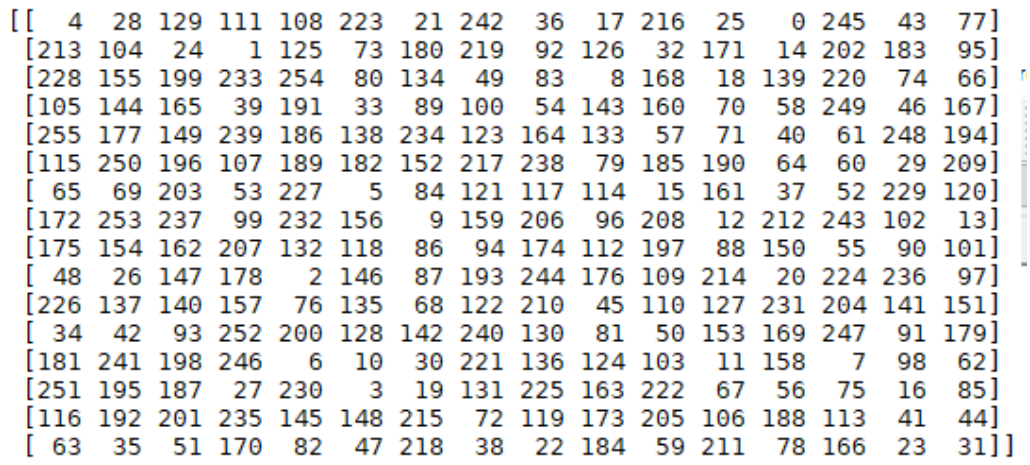
```
27        zps1 = dec_bin(y[i+1])
28        zs1 = len(zps1)
29        zps2 = dec_bin(z[i+1])
30        zs2 = len(zps2)
31        zps3 = XOR_func(zps1[zs1 - 8], zps2[zs2 - 8])
32        sale = int(zps3,2)
33        if sale in sbox:
34                i = i + 1
35        else:
36                sbox.append(sale)
37                j = j + 1
38        return sbox
39
40 // Se manda a llamar la funcion para la S-box y los resultados se
    ordenan en forma de matriz
41 xs = sbox()
42 xl = nm.array(xs).reshape(Sn,Sm)
```

Habiendo obtenido ambas metodologías se procedió a implementarlas con los sistemas que si aprobaron las pruebas de la NIST, en la figura 4.16 se aprecia la metodología de [1] con el atractor de mega estabilidad y en la figura 4.15 la metodología de [2] con el sistema de mega estabilidad.

```
[[ 23 212 201  34  83 241  68 255 155  31 160  87   6 125 200  37]
 [  0 156 246 173  73  26 165  20   8 139  92 188  43  79  40 140]
 [ 10 229  77 107 250 159  67 220 164   7  84 176  16  76  95 185]
 [ 74  21 214 122   3 181 211 209 147 213 218 207  70  12 128  17]
 [178 180 101 210 106 192 148  24 136 105 190 205 195  13  81  27]
 [ 99 239   2  22  35 186 227 162 138  29  32  33 110 129 121 150]
 [131 102   5 130 112  28  51  75 235 222  78 189 120 230   1 215]
 [221 171 158 247 113   9  47  71  54 141 145  49  53  72 134 182]
 [238 161 225 224 219 223  66 248 206 202 118  98 167 123  94 127]
 [193   4  42 116  65 208 117 191  36 146  90 249  60  44 137 203]
 [197  38  55 126  59 108 149  86 179 199 194  63 103 242 236 100]
 [226 109 240 233  14 135 166  96  80  46 187 119  88 175 243 204]
 [196 144 244 231 245  19  61 142 153  45  48  15  82 253 124 111]
 [152 251  18 237  41 216 252 217 114  97  25 163  52 228 169 184]
 [ 50  62 183 168 115 172 174  57  58  56  69 151 133  91 234 104]
 [ 11  64 170 254 177 232  30 143 154  89 157 198  39  85  93 132]]
```

Figura 4.15: S-box creada con la metodología [1] y el sistema con mega estabilidad (3.13).

De igual manera se evaluaron dichas metodologías para la obtención de Sbox en el caso de sistemas con puntos de equilibrio infinitos, los resultados de la metodología de [1] se muestran en la figura 4.17 y los resultados de la metodología [2] se muestran en la figura 4.18.



```
[[  4  28 129 111 108 223  21 242  36  17 216  25   0 245  43  77]
 [213 104  24   1 125  73 180 219  92 126  32 171  14 202 183  95]
 [228 155 199 233 254  80 134  49  83   8 168  18 139 220  74  66]
 [105 144 165  39 191  33  89 100  54 143 160  70  58 249  46 167]
 [255 177 149 239 186 138 234 123 164 133  57  71  40  61 248 194]
 [115 250 196 107 189 182 152 217 238  79 185 190  64  60  29 209]
 [ 65  69 203  53 227   5  84 121 117 114  15 161  37  52 229 120]
 [172 253 237  99 232 156   9 159 206  96 208  12 212 243 102  13]
 [175 154 162 207 132 118  86  94 174 112 197  88 150  55  90 101]
 [ 48  26 147 178   2 146  87 193 244 176 109 214  20 224 236  97]
 [226 137 140 157  76 135  68 122 210  45 110 127 231 204 141 151]
 [ 34  42  93 252 200 128 142 240 130  81  50 153 169 247  91 179]
 [181 241 198 246   6  10  30 221 136 124 103  11 158   7  98  62]
 [251 195 187  27 230   3  19 131 225 163 222  67  56  75  16  85]
 [116 192 201 235 145 148 215  72 119 173 205 106 188 113  41  44]
 [ 63  35  51 170  82  47 218  38  22 184  59 211  78 166  23  31]]
```

Figura 4.16: S-box creada con la metodología [2] y el sistema con mega estabilidad (3.13).

```
[[ 75 168  29 205 242   6  57  47 177 251 103 214  66 140  19  38]
 [186  85 223 145  10 233 112  34 173 158  94 210 195 127  99 192]
 [117 130 229  84 149 220 131 183 232  62  43 137 247 239 202 109]
 [ 71 121   3  26 196 113 165 102  56  89 122  81  16 224  93 154]
 [155  54  53   5  90  74  15  61 167  65  28 150 187 159 204  82]
 [141   2 164 176  33 241  46  25 213 250 126 238 201  37 110  72]
 [139 248 172  44 191 144  18   9 118 138 185 209 182  20 111   4]
 [219 157 116  98 222 228 148 136  11  83 221  42 246 230  27 194]
 [120  88  55 203 100 184  70 108 166 240 129 153 231  60 243  92]
 [ 14 211  80 206 193  32 101 125 146 252   7 256  52 190 171  30]
 [ 64 174  39  73 215  97 227 128 163 208 181 237 218 200 143  24]
 [ 69  48 169  41   1  17 245 175 135  45 107  79  36 255 249  51]
 [162 115 212 236 199  23 156 178  13 134  35 152   8 180 106  87]
 [ 63 217  58   0 147  78  91 119 189  76 124  59  67  50 226 254]
 [ 86  31  96 114 142  22 170 161  95 207  68 104 198  40 244 151]
 [123 235  12 234 133 225 188 132 105 179 197 216 160  77 253  21]]
```

Figura 4.17: S-box creada con la metodología [1] y el sistema con puntos de equilibrio infinitos (3.7).

```
[[217  71 119 172 113  38 237 144 222  78  27 153 213   4 106 170]
 [ 21  63 181 241 141  18  64 238  15   9 239 174 152 212 249 250]
 [ 94 219 164   5 195   0 133  33 134   1  99 107 245 194  34 166]
 [116  88  98  54 139  10  48 254   2  51  89 118 182  41  75 148]
 [ 36 129  49   7  28  68 207 228 198  43 128  13 193  17 145  12]
 [125  26  83  65 165 150 196 248 183 100  58  42 184 130 208 163]
 [189 123 180 246 191  93 215 209 232  82  35 175 154 235 114  55]
 [229 138 214  16  45  84  86 127 140  44  80  74  46 109 159 149]
 [ 92 234 161  66 227  24  96   3 251 151 157 200 132 167 112  79]
 [216 226  85 143 244  91 178  53 240 190 205 211 203  56 255 122]
 [ 76  52  70 220 253  72  81 247 210 104 147 126 231  77  61 243]
 [142 224 188  39   8 252 168 137 117 121   6  59  69 120 221 105]
 [131 124 192  11  22  62 173  32  40  97 110 223 103 160 187 225]
 [179 177 146  50 135  60 202  25  23  90 230 186 185 218 111 197]
 [ 73 115  87  30 102  14 136 206  19 155 162  29  20  67 236 176]
 [158 199 169 233 204  31  37 108 242  95  57 171 201  47 156 101]]
```

Figura 4.18: S-box creada con la metodología [2] y el sistema con puntos de equilibrio infinitos (3.7).

Capítulo 5

Conclusiones

Este trabajo fue realizado con la finalidad de comprobar si los sistemas caóticos con atractores ocultos en orden fraccionario podían ser utilizados para aplicaciones de cifrado de datos por lo que fueron evaluados cuatro diferentes sistemas pertenecientes a las diferentes familias de los sistemas con atractores ocultos, se realizó un análisis de su dinámica para conocer el comportamiento de estos sistemas y saber que parámetros eran los más óptimos. Tras conocer la dinámica de estos sistemas se crearon generadores de números aleatorios y se evaluaron en las pruebas de la NIST, esto dio como resultado que solo dos de los cuatro sistemas evaluados (sistema (3.7), sistema (3.13)) pasaran las pruebas de la NIST y con ello demostrar que eran buenos generadores, que contaban con buena aleatoriedad y que podrían ser utilizados para aplicaciones criptográficas, a partir de ello se obtuvieron las cajas de sustitución. Esto demuestra que a pesar de que existen sistemas que no aprobaron las pruebas de generadores de números aleatorios, existen algunos otros sistemas con atractores ocultos que si tienen buenas características para ser utilizados en aplicaciones para cifrado de datos.

Apéndice

Artículo Internacional

2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC 2018). Ixtapa, Mexico

On different families of hidden chaotic attractors in fractional order dynamical systems

Samuel Giovanni Hernández-Orbe, Jesús Manuel Muñoz-Pacheco, Germán Ardúl Muñoz-Hernández, Ernesto Zambrano-Serrano
Facultad de Ciencias de la Electrónica
Benemérita Universidad Autónoma de Puebla
Puebla, México
samuel.hernandezo@alumno.buap.mx

***Abstract*—Chaotic systems with hidden attractors and their families (infinite number of equilibria, stable equilibria and without equilibria) are important in applications of engineering. However, studies about hidden attractors in fractional order dynamical systems are limited. In this paper, we perform a numerical analysis of fractional order chaotic systems with hidden attractors by using the Gründwald-Letnikov integration method. In particular, we determine that the fractional order is a key parameter to observe hidden chaotic attractors belonging to the aforementioned families.**

Keywords – Chaos, hidden attractors, fractional derivative, Gründwald-Letnikov.

I. Introduction.

The chaotic behavior refers to a complex dynamic phenomenon with an extreme sensitivity to small variations of initial conditions, confined trajectories in phase space, a positive Lyapunov exponent, a finite Kolmogorov-Sinai entropy, a continuous power spectrum, and so on [1]. Chaos have been observed in several scientific areas such as turbulence in fluids, retro-fed laser devices, economic models, robotics, and biological processes [2]. Chaos can be defined as an unpredictable behavior of a deterministic system (its current condition is a consequence of previous states) in long-term. Additionally, it exhibits behavior which is "difficult to distinguish from random behavior" [3] since is very sensitive to initial conditions.

A common technique to visualize the chaos is by plotting the strange attractor of the system. In simple words, a chaotic attractor displays the trajectory of a state variable around one or multiple equilibrium points with respect to another state variable in the phase space. Also, the trajectories around a certain equilibrium point form a scroll, a chaotic attractor can be composed by an arbitrary number of scrolls. A recently classification defines two kinds of attractors: self-excited and hidden [4]. An attractor is called self-excited if its basin of attraction intersects with an unstable equilibrium point. These chaotic attractors are easy to locate because they are very close from the equilibrium points of the system [5]. This is the most overwhelming type of chaotic attractors. From that point of view, the well-known nonlinear systems, such as Lorenz system, Rössler system, Chen system, or Sprott system, belong to chaotic systems with self-excited attractors [6].

On the other hand, an attractor is called hidden if its basin of attraction does not intersect with small neighborhoods of the equilibrium points. The localization of this kind of chaotic attractors is a challenge since there is no possibility to use information about equilibria for organizing a numerical search similar to the standard computational procedure using a integration method [6]. Therefore, specific approaches based on homotopy and numerical continuation have been proposed [7].

Additionally, the attractors generated by the different families of dynamical systems with an infinite number of equilibrium points, with stable equilibrium and without equilibrium are also defined as hidden attractors. The main characteristic is that those hidden attractors can be found by using traditional integration numerical methods. Some works related to those families are presented below. The works of Nóse [8] and Hoover [9] in 1984-1985 have led the study of the dynamical system without equilibria ant its various modifications, where hidden chaotic oscillations can be found [10]. Other works such as Jafari et al. to determine simple quadratic flows with no equilibria [11]. Wang and Chen found a new system without equilibrium while studying a chaotic system with any number of equilibria [12]. Akgul et al. designed a random number generator with a 3D chaotic system without equilibrium point [13]. Wang and Chen designed a chaotic flow with only one stable equilibrium [14]. Molaie et al. discovered a quadratic chaotic flow with one stable equilibrium [15]. Regarding to the chaotic systems with an infinite number of equilibrium points, there are two main research directions. One of them is focused on lines of equilibria, e.g., simple chaotic systems with a line of equilibrium were introduced by Wang and Chen in [14], and Jafari and Sprott in [16]. Other five chaotic systems were discovered by Li and Sprott [17], where one system has two infinite parallel lines of equilibrium [6]. The other direction is concentrated on chaotic systems with circular equilibrium [18]. Gotthans et al. reported another chaotic system with circular equilibrium [19]. From the last one, it was proposed a 3D system with square equilibriums points.

The study of hidden attractors is vital for engineering applications because it permits to analyze the occurrence of unexpected behaviors in some real nonlinear dynamic systems.

978-1-5386-5935-9/18/$31.00 ©2018 IEEE

IEEE

Advancing Technology for Humanity

CENTRO OCCIDENTE SECTION

THE ORGANIZING COMMITTEE OF THE
2018 IEEE AUTUMN MEETING ON POWER, ELECTRONICS AND COMPUTING

ROPEC 2018

GRANTS THIS

CERTIFICATE

TO

Samuel Giovanni Hernandez-Orbe, Jesús Manuel Muñoz-Pacheco, Germán Ardúl Muñoz-Hernández and Ernesto Zambrano-Serrano

for the presentation of the paper

On different families of hidden chaotic attractors in fractional order dynamical systems

M.C. Juan Carlos Olivares Rojas
GENERAL CHAIR
ROPEC 2018

IEEE
SECCION
CENTRO
OCCIDENTE

M.C. Jose Alberto Avalos González
CHAIR
IEEE CENTRO OCCIDENTE SECTION

Ixtapa, Zihuatanejo, México; November 14 - 16, 2018

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Artículo Nacional

Python-based Implementation of the Grünwald-Letnikov algorithm for fractional-order systems

Samuel Giovanni Hernández-Orbe, Jesús Manuel Muñoz-Pacheco, Germán Ardúl Muñoz-Hernández, Ernesto Zambrano-Serrano
Facultad de Ciencias de la Electrónica
Benemérita Universidad Autónoma de Puebla
Puebla, México
samohdez@gmail.com

***Abstract*—The fractional order calculus became in an important research subject during the last years because it presents relevant properties such as more accurate results, memory properties, and an extra freedom degree. These characteristics are crucial to obtain new engineering applications. In this paper, we develop a computer program by using multi-parading Python language in order to implement the Grünwald-Letnikov algorithm. This algorithm is necessary to study fractional-order dynamical systems. Additionally, we use a short memory principle to reduce the computational effort. To validate our proposal, we also present experimental results obtained in an ARM processor for a fractional-order chaotic system.**

Keywords* — *Cálculo fractional, Grünwald-Letnikov, Python, derivada fraccional.

I. Introducción.

El concepto de la derivada de orden fraccional nace en 1695 cuando L'Hopital and Leibniz se preguntaron qué significado tendría $D^n f$, si n tuviera el valor de una fracción. Desde entonces, muchos matemáticos como lo son Euler, Laplace, Fourier, Abel, Liouville, Riemann, entre otros, han realizado contribuciones a la teoría del cálculo fraccional [1]. A pesar de tener una larga historia, no se utilizó en física o ingeniería por muchos años. En los últimos años ha surgido un gran interés no solo en matemáticos, sino también entre físicos e ingenieros dado que se ha encontrado que los sistemas de orden fraccionario tienen ventajas en comparación con los sistemas de orden entero. La novedad más relevante es que los sistemas de orden fraccionario consideran la historia del sistema, por lo que se dice que tiene memoria [2] y por lo tanto pueden aproximar con una mejor precisión los fenómenos reales [3]. Actualmente, en múltiples áreas del conocimiento se han utilizado sistemas de orden fraccionario [4], por ejemplo: en física se ha determinado la impedancia espectroscópica eléctrica usando un modelo de orden fraccionario, en sistemas de control se han obtenido sistemas de posicionamiento de precisión basados en controladores PID de orden fraccionario, en procesamiento de imágenes y señales, se ha empleado el orden fraccionario como un grado extra de libertad para mejorar la resolución y la calidad de nanoimágenes, también en sistemas mecánicos como en el análisis de vibración de una viga/placa sobre una base viscoelástica, en biología para estudiar la interacción del VIH en la presencia de múltiples fármacos, en economía para obtener modelos más cercanos a los mercados macroeconómicos con memoria dinámica, entre otros.

Una parte significativa en el desarrollo de aplicaciones de ingeniería consiste en realización física de los sistemas de orden fraccionario, para la cual se pueden utilizar circuitos electrónicos [3]. En la literatura, se reportan comúnmente circuitos analógicos para la implementación electrónica [5, 6]. Sin embargo, en los últimos años, se ha también reportado y validado la utilización de diversos sistemas embebidos tales como FPGA, DPS, microcontroladores, etc. para implementar sistemas dinámicos de orden fraccionario [7-11].

El primer paso para obtener una implementación física en hardware de un sistema de orden fraccionario y por consecuencia su aplicación de ingeniería consiste en solucionar el sistema de ecuaciones mediante un método de discretización. Sin embargo, esta no es una tarea trivial desde que este tipo de sistemas requieren de algoritmos especializados, además de el correcto manejo de la longitud de memoria del operador fraccionario.

Por lo tanto, en este trabajo se propone la utilización del algoritmo numérico de Grünwald-Letnikov para solucionar los sistemas de orden fraccionario debido a que este algoritmo incorpora el concepto de memoria corta, la cual es adecuada para la implementación en hardware donde los recursos son limitados. Además, se propone utilizar el lenguaje de programación de Pyhton para obtener un algoritmo que pueda ser transferido a diversas plataformas digitales. Finalmente, para validar nuestra propuesta, se ha implementado un sistema de orden fraccionario con comportamiento caótico en una plataforma SoC (System on a Chip) ARM (Advanced RISC Machine).

El artículo se encuentra organizado de la siguiente manera: en la sección II se describen las definiciones matemáticas de la derivada en orden fraccional; en la sección III se muestra el algoritmo numérico de Grünwald-Letnikov. En la sección IV se describen detalladamente las sub-rutinas de programación para implementar sistemas de orden fraccionario usando Python.

IEEE
Advancing Technology for Humanity

SECCION MEXICO

OTORGA EL PRESENTE:

RECONOCIMIENTO

A:

Samuel Giovanni Hernández Orbe

POR SU AMABLE PARTICIPACION EN LA:

R C&C XXVIII REUNION INTERNACIONAL DE OTOÑO DE COMUNICACIONES, COMPUTACION, ELECTRONICA, AUTOMATIZACION, ROBOTICA Y EXPOSICION INDUSTRIAL '2018 / 2019

LAS COMUNICACIONES, LA ELECTRONICA, LA ROBOTICA Y LAS NUEVAS TECNOLOGIAS EN EL CONTEXTO ACTUAL

CON LA PONENCIA:

PYTHON-BASED IMPLEMENTATION OF THE GRÜNWALD-LETNIKOV ALGORITHM FOR FRACTIONAL-ORDER SYSTEMS

M.C. CESAR FUENTES ESTRADA
PRESIDENTE
IEEE SECCION MÉXICO

M.C. GILBERTO ENRIQUEZ HARPER
COORDINADOR GENERAL
ROC&C'2018/2019

6, 7 y 8 de Marzo de 2019, Acapulco, Gro.

Bibliografía

[1] G. Liu, W. Yang, and W. Liu, "Designing s-boxes based on 3-d four-wing autonomous chaotic system," *Nonlinear Dynamics*, vol. 82, pp. 1867–1877, 2015.

[2] Q. Lai, A. Akgul, C. Liu, G. Xu, and U. Cavusoglu, "A new chaotic system with multiple attractors: Dynamic analysis, circuit realization and s-box design," *Entropy*, vol. 20, pp. 1–15, 2017.

[3] S. H. Strogatz, *Nonlinear Dynamics and Chaos: with Applications to Physics, Biology, Chemistry and Engineering*. Boulder, CO: Westview Press, 2001.

[4] V. S. Anishchenko, V. Astakhov, A. Neiman, T. Vadivasova, and L. Schimansky-Geier, *Nonlinear Dynamics of Chaotic and Stochastic Systems: Tutorial and Modern Developments*. Heidelberg, DE: Springer-Verlag GmbH, 2007.

[5] J. A. Rodríguez and J. Morales, "Sincronización de caos mediante observadores para cifrado en comunicaciones," *Revista Ingenierías*, vol. 10, pp. 44–50, 2007.

[6] P. A. Cook, *Nonlinear Dynamical Systems*. Hertfordshire, UK: Prentice Hall, 1994.

[7] J. Lu and G. Chen, "Generating multiscroll chaotic attractors: theories, methods and applications," *Int. J. Bifurcation and Chaos*, vol. 16, pp. 775–858, 2006.

[8] J. Cordova-Zecena, *Chaotic Dynamical Systems and Their Applications*, pp. 167–180. Singapore: Proc. XXv CURCCAF, 2001.

[9] S. T. Kingni, V. T. Pham, S. Jafari, and P. Woafo, "A chaotic system with an infinite number of equilibrium points located on a line and on a hyperbola and its fractional-order form," *Chaos, Solitons and Fractals*, vol. 99, pp. 209–218, 2017.

[10] E. Zambrano-Serrano, *Fractional Order Chaotic Systems and Their Electronic Design*. PhD thesis, 2017.

[11] Y. Feng and W. Pan, "Hidden attractors without equilibrium and adaptive reduced-order function projective synchronization from hyperchaotic," *Prama Journal of Physiscs*, vol. 88, p. 62, 2017.

[12] J. Moreno, F. Parra, R. Húerfano, C. Suárez, and I. Amaya, "Modelo de encriptación simétrica basada en atractores caóticos," *Revista Ingenierías*, vol. 21, pp. 378–390, 2016.

[13] E. Zambrano-Serrano, E. Campos-Cantón, and J. M. Muñoz-Pacheco, "Strange attractor generated by a fractional order switching system and its topological horseshoe," *Nonlinear Dynamics*, vol. 82, pp. 1623–1641, 2015.

[14] V. Pham, C. Volos, and T. Kapitaniak, *Systems with Hidden Attractors From Theory to Realization in Circuits*. Switzerland: Springer, 2017.

[15] F. Pereira, *Tecnología ARM - Microcontroladores de 32 bits*. Brasil: ERICA, 2007.

[16] G. A. Leonov and N. V. Kuznetsov, "Hidden attractors in dynamical systems. from hidden oscillations in hilbert–kolmogorov, aizerman and kalman problems to hidden chaotic attractor in chua circuits," *Int. J. Bifurcat. Chaos*, vol. 23, p. 1330002, 2013.

[17] S. Nose, "A molecular dynamics method for simulations in the canonical ensamble," *Molecular Phys.*, vol. 52, pp. 255–268, 1984.

[18] W. G. Hoover, "Canonical dynamics: equilibrium phase-space distributions," *Phys. Rev. A.*, vol. 31, pp. 1695–97, 1985.

[19] H. A. Posch, W. G. Hoover, and F. J. Vesely, "Canonical dynamics of the nose oscillator: stability, order, and chaos," *Phys. Rev. A.*, vol. 33, pp. 4253–65, 1986.

[20] S. Jafari, J. Sprott, and S. M. R. H. Golpayegani, "Elementary quadratic chaotic flows with no equilibria," *Phys. Leters. A*, vol. 377, pp. 699–702, 2013.

[21] J. M. Muñoz-Pacheco, E. Zambrano-Serrano, C. Volos, S. Jafari, J. Kengne, and K. Rajagopal, "A new fractional-order chaotic system with different families of hidden and self-excited attractors," *Entropy*, vol. 20, pp. 1–18, 2018.

[22] E. Dong, Z. Liang, S. Du, and Z. Chen, "Topological horseshoe analysis on a four-wing chaotic attractor and its fpga implement," *Nonlinear Dyn*, vol. 83, pp. 623–30, 2016.

[23] E. Tlelo-Cuautle, V. H. Carbajal-Gomez, P. J. Obeso-Rodelo, J. J. Rangel-Magdaleno, and J. C. Nunez-Perez, "Fpga realization of a chaotic communication system applied to image processing," *Nonlinear Dyn*, vol. 82, pp. 1879–92, 2015.

[24] P. Chen, S. Yu, X. Zhang, J. He, Z. Lin, C. Li, and J. Lü, "Arm-embedded implementation of a video chaotic secure communication via wan remote transmission with desirable security and frame rate," *Nonlinear Dyn*, vol. 86, pp. 725–40, 2016.

[25] M. S. Azzaz, C. Tanougast, S. Sadoudi, R. Fellah, and A. Dandache, "A new autoswitched chaotic system and its fpga implementation," *Commun Nonlinear Sci Numer Simul*, vol. 18, pp. 1792–804, 2013.

[26] Z. Zhang, G. Chen, and S. Yu, "Hyperchaotic signal generation via dsp for efficient perturbations to liquid mixing," *Int J Circuit Theory Appl*, vol. 37, pp. 31–41, 2009.

[27] E. Zambrano-Serrano, J. Muñoz-Pacheco, and E. Campos-Cantón, "Chaos generation in fractional-order switched systems and its digital implementation," *Int. J. Electron. Commun*, vol. 79, pp. 43–52, 2017.

[28] V. Vujovic and M. Maksimovic, "Raspberry pi as a sensor web node for home automation," *Comput Electr Eng*, vol. 44, pp. 153–71, 2015.

[29] M. Jutila, E. Strömmer, M. Ervasti, M. Hillukkala, P. Karhula, and J. Laitakari, "Safety services for children: a weareable sensor vest with wireless charging," *Pers Ubiquit Comput*, vol. 19, pp. 915–27, 2014.

[30] K. Takeshi, T. Koide, and T. Fujino, "Secure data processing with massive-parallel simd matrix for embedded soc in digital-convergence mobile devices," *IEEJ Trans Electr Electron Eng*, vol. 12, pp. 96–104, 2017.

[31] I. Petras, *Fractional-Order Nonlinear Systems, Modeling, Analysis and simulation.* Slovak Republic: Springer, 2011.

[32] O. I. Tacha, J. M. Muñoz-Pacheco, E. Zambrano-Serrano, I. N. Stouboulos, and V. T. Pham, "Determining the chaotic behavior in a fractional order finance system with negative parameters," *Nonlinear Dynamics*, vol. 94, pp. 1303–1317, 2018.

[33] Y. Luo and Y. Chen, *Fractional Order Motion Controls.* United Kingdom: John Wiley and Sons, Ltd., Publication, 2013.

[34] D. Matignon, "Generalized fractional differential and difference equations: Stability properties and modeling issues," *Proceedings of Math. Theory of Networks and Systems Symposium*, 1998.

[35] D. Matignon, "Stability result of fracional differential equations with applications to control processing," *Proceedings IMACS-SMC*, pp. 963–968, 1996.

[36] Z. Odibat, N. Corson, M. A. Aziz-Alaoui, and A. Alsaedi, "Chaos in fractional order cubic chua system and synchronization," *Int. J. Bifurc. Chaos*, vol. 27, p. 1750161, 2017.

[37] E. Ahmed, A. M. A. El-Sayed, and H. A. A. El-Saka, "Equilibrium points, stability and numerical solutions of fractional-order predator-prey and rabies models," *J. Math. Anal. Appl*, vol. 325, pp. 542–553, 2007.

[38] M. S. Tavazoei and M. Haeri, "Unreliabilitu of frequency-domain approximation in recognising chaos in fractiona-order systems," *IET. Signal Proc.*, vol. 1, pp. 171–181, 2007.

[39] M. S. Tavazoei and M. Haeri, "A necessary condition for double scroll attractor existance in fractional-order systems," *Phys. Lett. A*, vol. 367, pp. 102–113, 2007.

[40] M. Danca, "Hidden chaotic attractors in fractional-order systems," *Nonlinear Dyn.*, vol. 89, pp. 577–586, 2017.

[41] M. S. Tavazoei and M. Haeri, "Chaotic attractors in inconmmensurate fractional order systems," *Phys. D.*, vol. 237, pp. 2628–2637, 2008.

[42] G. C. Layel, *An Introduction to Dynamical Systems and Chaos*. New York, USA: Springer Berlin Heidelberg, 2015.

[43] J. L. Moiola and G. Chen, *Hopf Bifurcation Analysis: A Frecuency Domain Approach*. World Scientific Publishing, 1996.

[44] C. Li and Y. Ma, "Fractional dynamical systems and its linearization theorem," *Nonlinear Dyn.*, vol. 71, pp. 621–633, 2013.

[45] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*. Colorado, USA: Studies in Nonlinearity. Westview Press, 2003.

[46] H. Sira-Ramirez and C. Cruz-Hernández, "Synchronization of chaotic systems: A generalized Hamiltonian systems approach," *Int. J. Bifurcation and Chaos*, vol. 11, pp. 1381–1395, 2001.

[47] L. Gamez-Guzman, C. Cruz-Hernandez, and R. M. Lopez-Gutierrez, "Synchronization of multi-scroll chaos generators: application to private communication," *Revista Mexicana de Física*, vol. 54, pp. 299–305, 2008.

[48] J. He, "Nonlinear science as a fluctuating research frontier," *Chaos Solitons and Fractals*, vol. 41, pp. 2533–2537, 2009.

[49] J. M. Muñoz-Pacheco and E. Tlelo-Cuautle, *Electronic design automation of multiscroll chaos generators*. Dubai, UAE: Bentham Sciences Publishers, 2010.

[50] G. A. Leonov, N. V. Kuznetsov, and V. I. Vagaitsev, "Localization of hidden Chua's attractors," *Physical Review Letters*, vol. 375, pp. 2230–2233, 2011.

[51] S. T. Kingni, S. Jafari, H. Simo, and P. Woafo, "Three-dimensional chaotic autonomous system with only one stable equilibrium: Analysis, circuit design, parameter estimation, control, synchronization and its fractional-order form," *Eur. Phys. J. Plus*, vol. 139, pp. 76–92, 2014.

[52] X. Wang, A. Ouannas, V. T. Pham, and H. R. Abdolmohammadi, "A fractional-order form of a system with stable equilibria and its synchronization," *Adv. Differ. Equ*, vol. 20, pp. 1–13, 2018.

[53] V. T. Pham, A. Ouannas, C. Volos, and T. Kapitaniak, "A simple fractional-order chaotic system without equilibrium and its synchronization," *Int. J. Electron. Commun. (AEÜ)*, vol. 86, pp. 69–76, 2018.

[54] V. T. Pham, S. T. Kingni, C. Volos, S. Jafari, and T. Kapitaniak, "A simple three-dimensional fractional-order chaotic system without equilibrium: Dunamics, circuitry implementation, chaos control and synchronization," *Int. J. Electron. Commun. (AEÜ)*, vol. 78, pp. 220–227, 2017.

[55] D. Cafagna and G. Grassi, "Elegant chaos in fractional-order system without equilibria," *Math. Probl. Eng.*, vol. 2013, p. 380436, 2013.

[56] D. Cafagna and G. Grassi, "Chaos in a new fractional-order system without equilibrium points," *Commun. Nonlinear Sci. Numer. Simul*, vol. 19, pp. 2919–2927, 2014.

[57] S. T. Kingni, V. T. Pham, S. Jafari, G. R. Kol, and P. Woafo, "Three-dimensional chaotic autonomous system with a circular equilibrium: Analysis, circuit implementation and its fractional-order form," *Circuits Syst. Signal Process*, vol. 35, pp. 1943–1948, 2016.

[58] D. Dudkowski, S. Jafari, T. Kapitaniak, N. Kuznetsov, G. Leonov, and A. Prasad, "Hidden attractors in dynamical systems," *Phys. Rep.*, vol. 637, pp. 1–50, 2016.

[59] D. E. Knuth, *The Art of Computer Programming Volume 2, Seminumerical Algorithms*. Boston: Addison Wesley, 1998.

[60] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications," *NIST Special Publication 800-22*, vol. Revision 1a, pp. 1–131, 2010.

[61] B. Jun and P. Kocher, "The intel random number generator," *Cryptography Research Inc.*, vol. N.D., pp. 1–8, 1999.

[62] T. Stojanovski and L. Kocarev, "Chaos-based random number generators - part i: Analysis," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, pp. 281–288, 2001.

[63] B. Blaner, B. Abali, B. M. Bass, S. Chan, R. Kalla, S. Kunkel, K. Launcella, R. Leavens, J. J. Reilly, and P. A. Sandon, "Processor on-chip accelerators for cryptography and active memory expansión," *IBM Journal of Research and Development*, vol. 57, pp. 1–16, 2013.

[64] L. G. D. la Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, and C. Mancillas-López, "Hardware implementation of pseudo-random number generators based on chaotic maps," *Nonlinear Dynamics*, vol. 90, pp. 1661–1670, 2017.

[65] J. M. M. Pacheco, "Infinitely many hidden attractors in a new fractional-order chaotic system based on a fracmemristor," *Eur. Phys. J. Special Topics*, vol. 228, pp. 2185–2196, 2019.

[66] J. A. Mauricio, *Análisis de Series Temporales*. 2007.

[67] D. Peña and I. Sánchez, "El análisis de series temporales: situación y perspectivas," *Artículos de Estadística*, vol. 23, pp. 4–8, 2007.

[68] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*. New York, USA: Springer-Verlag, 1989.

[69] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. New York, USA: Springer-Verlag, 2003.

[70] P. Palaniyandi, "On computing Poincaré map by Hénon method," *Chaos, Solitons and Fractals*, vol. 39, pp. 1877–1882, 2009.

[71] M. Hénon, "On the numerical computation of Poincaré maps," *Physica D*, vol. 5, pp. 412–414, 1982.

[72] W. Tucker, "Computing accurate Poincaré maps," *Physica D*, vol. 171, pp. 127–137, 2002.

[73] M. L. Vilches, F. Velasco, and J. J. García, "Bifurcaciones transcríticas y ciclos límites en un modelo dinámico de competición entre dos especies. una aplicación a la pesquería de engraulis encrasicholus de la región suratlántica española," *Estudios de Economía Aplicada*, vol. 20, pp. 651–677, 2002.

[74] K. T. Chau and Z. Wang, *Chaos in Electric Drive Systems Analysis, Control and Application*. China: John Wiley and Sons, 2011.

[75] A. Wolf, J. Swift, H. Swinney, and J. Vastano, "Determining Lyapunov exponents from a time series," *Physica D*, vol. 16, pp. 5–317, 1985.

[76] M. Sandri, "Numerical calculation of Lyapunov exponents," *The Mathematica Journal*, vol. 6, pp. 78–84, 1996.

[77] S. Lynch, *Dynamical Systems with Applications using MATLAB*. Manchester, UK: Birkhäuser, 2014.

[78] S. J. Johnston and S. J. Cox, *Raspberry Pi Technology*. UK: MDPI, 2017.

[79] B. Ying, *Practical Microcontroller Engineering with ARM Technology*. John Wiley and Sons, Inc, 2016.

[80] B. Stoyanov, M. Kolev, and A. Nachev, "Design of a new self-shrinking 2-adic cryptographyc sistem with application to imagen encryption," *European Journal of Scientific Research*, vol. 78, pp. 362–374, 2012.