ICCD412– MÉTODOS NUMÉRICOS

FACULTAD DE INGENIERÍA DE SISTEMAS

INGENIERÍA EN COMPUTACIÓN

“Movimiento de un Nanodron”

DOCENTE: Ing. Jonathan A. Zea

Grupo: 4

Nombres: José Luis Andino Padilla

Fabián Alexander Simbaña Pinduisaca

Oscar Paul Albán Campana

Roberth Alexander Gancino Toalombo

Fecha de entrega: 16-08-2024

Índice de Contenido

Índice de Figuras

1. INTRODUCCIÓN

El presente informe detalla el desarrollo de una simulación que modela el movimiento de un NANODRON en el espacio utilizando el sistema de Lorenz, un conjunto de ecuaciones diferenciales que describe sistemas caóticos. El sistema de Lorenz es un modelo matemático compuesto por tres ecuaciones diferenciales no lineales, que inicialmente fue concebido para modelar las convecciones atmosféricas (Argote, 2013). En este contexto, el sistema se adapta para simular el movimiento de un NANODRON, con el fin de analizar cómo pequeños cambios en las condiciones iniciales pueden afectar significativamente las trayectorias futuras del dron. Este fenómeno, conocido como sensibilidad a las condiciones iniciales, es una característica fundamental de los sistemas caóticos. En este proyecto nos centraremos en la simulación del movimiento de un NANODRON utilizado para aplicaciones meteorológicas. Se estudia su comportamiento bajo las ecuaciones de movimiento conocidas como el sistema de Lorenz, con parámetros específicos para α , β y γ. Además, se proporciona una visualización interactiva para observar cómo varían las trayectorias de dos configuraciones iniciales diferentes del dron.

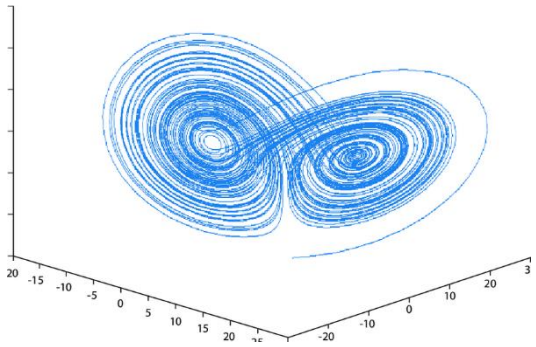

Figura 1. Atractor de Lorenz

En la Figura 1 se tiene la representación gráfica de las ecuaciones diferenciales de Lorenz.

2. METODOLOGÍA

Para lograr graficar el movimiento del nanodron se deben resolver las ecuaciones diferenciales de Lorenz las cuales son ODEs, de modo que la solución planteada para la simulación del movimiento de un nanodron fue la utilización del algoritmo “Odeint” de Scipy, la cual tiene como propósito resolver sistemas de ecuaciones diferenciales ordinarias.

Además, se implementó una interfaz gráfica mediante las librerías Matplotlib y PyQt5 donde se deben ingresar los valores de las coordenadas en el espacio del nanodron (x, y, z), constantes positivas (α, β, γ) y el tiempo en segundos del movimiento del nanodron.

2.1 Desarrollo Matemático

Las siguientes ecuaciones son las correspondientes al movimiento del nanodron en los ejes x, y, z.

$$\frac{\delta x}{\delta t} = \alpha\,(y - x)$$

$$\frac{\delta y}{\delta t} = x(\beta - z) - y$$

$$\frac{\delta z}{\delta t} = xy - \gamma z$$

Se puede apreciar que se tratan de ODEs de primer orden.

La función Odeint de Scipy utiliza LSODA (Livermore Solver for Ordinary Differential Equations) desde la librería de FORTRAN "odepack", la cual emplea diferentes métodos matemáticos para resolver ODEs dependiendo de la naturaleza del problema.

Al realizar una investigación sobre qué tipo de problema es el correspondiente a las ecuaciones del atractor de Lorenz se llegó a la conclusión de que se trata de un sistema no rígido de modo que mediante la librería odepack se utilizaron el método de Adams-Bashforth-Moulton o Runge-Kutta.

Nota: Los métodos de Adams-Bashforth-Moulton y Runge-Kutta son equivalentes al método de Euler ya que estos tienen formulas similares y utilizan una aproximación inicial.

$$y_{n+1} = y_n + hf(x_n, y_n; h) \rightarrow Método\ de\ Runge\ Kutte - ODE\ primer\ orden$$

$$z_{n+1} = z_n + hf(t_n, z_n) \rightarrow Método\ de\ Adams\ Moulton\ -\ ODE\ primer\ orden$$

$$x_{n+1} = x_n + hf(x_n, y_n) \rightarrow Método\ de\ Euler - ODE\ primer\ orden$$

Con respecto a las anteriores ecuaciones se empleará el método de Euler para la explicación del desarrollo matemático mediante un ejemplo.

$$\frac{\delta x}{\delta t} = 0.1\,(y - x)$$

$$\frac{\delta y}{\delta t} = x(0.1 - z) - y$$

$$\frac{\delta z}{\delta t} = xy - 0.1z$$

Aproximación inicial $x(0) = 1$
Aproximación inicial $y(0) = 1$

Aproximación inicial z(0) = 1
$x(10) = ?$
$y(10) = ?$
$z(10) = ?$

Como el intervalo del movimiento en cada uno de los ejes va de cero a diez se asumirá un valor de h igual a 0.5.

n	$t_n = 0 + nh$
0	0
1	0,5
2	1
3	1,5
4	2
5	2,5
6	3
7	3,5
8	4
9	4,5
10	5
11	5,5
12	6
13	6,5
14	7
15	7,5
16	8
17	8,5
18	9
19	9,5
20	10

Figura 2. Tablas de n valores de t

Los valores presentes en la Figura 2 corresponden a los puntos de malla, los cuales tienen una longitud entre sí de 0.5. Además, estos puntos no son utilizados en el cálculo debido a que no existe la variable t en las ecuaciones.

A continuación, se presentan las tablas de los cálculos realizados para la obtención de los puntos en los respectivos ejes coordenados.

n	t	$x = x_{n-1} + h\frac{\delta x}{\delta t}$	$\frac{\delta x}{\delta t} = 0{,}1\,(y - x)$
0	0	x_0 = 1 (valor de la condicion inicial)	0,1(1-1) = 0
1	0,5	x_1 = 1+0,5(0) = 1	0,1(0,05 - 1) = -0,095
2	1	x_2 = 1 + 0,5(-0,095) = 0,9525	0,1(-0,65 - 0,9525) = -0,16025
3	1,5	x_3 = 0,9525 + 0,5(-0,16025) = 0,872375	0,1(-0,945315625 - 0,872375) = -0,181769063
4	2	x_4 = 0,872375 + 0,5 (-0,181769063) = 0, 781490469	0,1(-0,87517709 - 0, 781490469) = -0,165666756
5	2,5	x_5 = 0, 781490469 + 0,5 (-0,165666756) = 0,698657091	0,1(-0,617072295 - 0,698657091) = -0,131572939
6	3	x_6 = 0,698657091 + 0,5(-0,131572939) = 0,632870622	0,1(-0,339765731 - 0,632870622) = -0,097263635
7	3,5	x_7 = 0,632870622 + 0,5(-0,097263635) = 0,584238805	0,1(-0,126964104 - 0,584238805) = -0,071120291
8	4	x_8 = 0,584238805 + 0,5(-0,071120291) = 0,54867866	0,1(0,007025143 - 0,54867866) = -0,054165352
9	4,5	x_9 = 0,54867866 + 0,5(-0,054165352) = 0,521595984	0,1(0,077964086 - 0,521595984) = -0,04436319
10	5	x_10 = 0,521595984 + 0,5(-0,04436319) = 0,499414389	0,1(0,107021175 - 0,499414389) = -0,039239321
11	5,5	x_11 = 0,499414389 + 0,5(-0,039239321) = 0,479794729	0,1(0,111570248 - 0,479794729) = -0,036822448
12	6	x_12 = 0,479794729 + 0,5(-0,036822448) = 0,461383505	0,1(0,103563433 - 0,461383505) = -0,035782007
13	6,5	x_13 = 0,461383505 + 0,5(-0,035782007) = 0,443492502	0,1(0,090408277 - 0,443492502) = -0,035308423
14	7	x_14 = 0,443492502 + 0,5(-0,035308423) = 0,425838291	0,1(0,076287386 - 0,425838291) = -0,034955091
15	7,5	x_15 = 0,425838291 + 0,5(-0,034955091) = 0,408360746	0,1(0,063293364 - 0,408360746) = -0,034506738
16	8	x_16 = 0,408360746 + 0,5(-0,034506738) = 0,391107377	0,1(0,052262661 - 0,391107377) = -0,033884472
17	8,5	x_17 = 0,391107377 + 0,5(-0,033884472) = 0,374165141	0,1(0,043339607 - 0,374165141) = -0,033082553
18	9	x_18 = 0,374165141 + 0,5(-0,033082553) = 0,357623865	0,1(0,036332899 - 0,357623865) = -0,032129097
19	9,5	x_19 = 0,357623865 + 0,5(-0,032129097) = 0,341559317	0,1(0,030924806 - 0,341559317) = -0,031063451
20	10	x_20 = 0,341559317 + 0,5(-0,031063451) = 0,326027592	

Figura 3. Tabla aproximaciones eje x

n	t	$y = y_{n-1} + h\frac{\delta y}{\delta t}$	$\frac{\delta y}{\delta t} = x(0{,}1 - z) - y$
0	0	y_0 = 1 (valor de la condicion inicial)	1(0,1 - 1) -1 = -1,9
1	0,5	y_1 = 1+0,5(-1,9) = 0,05	1(0,1 - 1,45) - 0,05 = -1,4
2	1	y_2 = 0,05 + 0,5(-1,4) = -0,65	0,9525(0,1 - 1,4025) - (-0,65) = -0,59063125
3	1,5	y_3 = -0,65 + 0,5 (-0,59063125) = -0,945315625	0,872375(0,1 - 1,0228125) - (-0,945315625) = 0,14027707
4	2	y_4 = -0,945315625 + 0,5(0,14027707) = -0,87517709	0,781490469(0,1 - 0,559337016) - (-0,87517709) = 0,51620959
5	2,5	y_5 = -0,87517709 + 0,5(0,51620959) = -0,617072295	0,698657091(0,1 - 0,189398888) - (-0,617072295) = 0,554613128
6	3	y_6 = -0,617072295 + 0,5(0,554613128) = -0,339765731	0,632870622(0,1 - (-0,035632024)) - (-0,339765731) = 0,425603254
7	3,5	y_7 = -0,339765731 + 0,5(0,425603254) = -0,126964104	0,584238805(0,1 - (-0,141364298)) - (-0,126964104) = 0,267978493
8	4	y_8 = -0,126964104 + 0,5(0,267978493) = 0,007025143	0,54867866(0,1 - (-0,171384762)) - 0,007025143 = 0,141877885
9	4,5	y_9 = 0,007025143 + 0,5(0,141877885) = 0,077964086	0,521595984(0,1 - (-0,160888251)) - 0,077964086 = 0,058114178
10	5	y_10 = 0,077964086 + 0,5(0,058114178) = 0,107021175	0,499414389(0,1 - (-0,132510962)) - 0,107021175 = 0,009098145
11	5,5	y_11 = 0,107021175 + 0,5(0,009098145) = 0,111570248	0,479794729(0,1 - (-0,099161457)) - 0,111570248 = -0,016013631
12	6	y_12 = 0,111570248 + 0,5(-0,016013631) = 0,103563433	0,461383505(0,1 - (-0,067437976)) - 0,103563433 = -0,026310313
13	6,5	y_13 = 0,103563433 + 0,5(-0,026310313) = 0,090408277	0,443492502(0,1 - (-0,040174848)) - 0,090408277 = -0,028241783
14	7	y_14 = 0,090408277 + 0,5(-0,028241783) = 0,076287386	0,425838291(0,1 - (-0,018118409)) - 0,076287386 = -0,025988045
15	7,5	y_15 = 0,076287386 + 0,5(-0,025988045) = 0,063293364	0,408360746(0,1 - (-0,000969444)) - 0,063293364 = -0,022061407
16	8	y_16 = 0,063293364 + 0,5(-0,022061407) = 0,052262661	0,391107377(0,1 - 0,012002291) - 0,052262661 = -0,017846108
17	8,5	y_17 = 0,052262661 + 0,5(-0,017846108) = 0,043339607	0,374165141(0,1 - 0,021622333) - 0,043339607 = -0,014013416
18	9	y_18 = 0,043339607 + 0,5(-0,014013416) = 0,036332899	0,357623865(0,1 - 0,028649302) - 0,036332899 = -0,010816187
19	9,5	y_19 = 0,036332899 + 0,5(-0,010816187) = 0,030924806	0,341559317(0,1 - 0,033713593) - 0,030924806 = -0,008284066
20	10	y_20 = 0,030924806 + 0,5(-0,008284066) = 0,026782773	

Figura 4. Tabla aproximaciones eje y

n	t	$z = z_{n-1} + h\frac{\delta z}{\delta t}$	$\frac{\delta z}{\delta t} = xy - \gamma z$
0	0	z_0 = 1 (valor de la condicion inicial)	1(1) - 0,1(1) = 0,9
1	0,5	z_1 = 1+0,5(0,9) = 1,45	1(0,05) - 0,1(1,45) = -0,095
2	1	z_2 = 1,45 + 0,5 (-0,095) = 1,4025	0,9525(-0,65) - 0,1(1,4025) = -0,759375
3	1,5	z_3 = 1,4025 + 0,5(-0,759375) = 1,0228125	0,872375(-0,945315625) - 0,1(1,0228125) = -0,926950968
4	2	z_4 = 1,0228125 + 0,5(-0,926950968) = 0,559337016	0,781490469(-0,87517709) - 0,1(0,559337016) = -0,739876256
5	2,5	z_5 = 0,559337016 + 0,5(-0,739876256) = 0,189398888	0,698657091(-0,617072295) - 0,1(0,189398888) = -0,450061823
6	3	z_6 = 0,189398888 + 0,5(-0,450061823) = -0,035632024	0,632870622(-0,339765731) - 0,1(-0,035632024) = -0,211464547
7	3,5	z_7 = -0,035632024 + 0,5(-0,211464547) = -0,141364298	0,584238805(-0,126964104) - 0,1(-0,141364298) = -0,060040927
8	4	z_8 = -0,141364298 + 0,5(-0,060040927) = -0,171384762	0,54867866(0,007025143) - 0,1(-0,171384762) = 0,020993022
9	4,5	z_9 = -0,171384762 + 0,5(0,020993022) = -0,160888251	0,521595984(0,077964086) - 0,1(-0,160888251) = 0,056754579
10	5	z_10 = -0,160888251 + 0,5(0,056754579) = -0,132510962	0,499414389(0,107021175) - 0,1(-0,132510962) = 0,066699011
11	5,5	z_11 = -0,132510962 + 0,5(0,066699011) = -0,099161457	0,479794729(0,111570248) - 0,1(-0,099161457) = 0,063446963
12	6	z_12 = -0,099161457 + 0,5(0,063446963) = -0,067437976	0,461383505(0,103563433) - 0,1(-0,067437976) = 0,054526257
13	6,5	z_13 = -0,067437976 + 0,5(0,054526257) = -0,040174848	0,443492502(0,090408277) - 0,1(-0,040174848) = 0,044112878
14	7	z_14 = -0,040174848 + 0,5(0,044112878) = -0,018118409	0,425838291(0,076287386) - 0,1(-0,018118409) = 0,034297931
15	7,5	z_15 = -0,018118409 + 0,5(0,034297931) = -0,000969444	0,408360746(0,063293364) - 0,1(-0,000969444) = 0,02594347
16	8	z_16 = -0,000969444 + 0,5 (0,02594347) = 0,012002291	0,391107377(0,052262661) - 0,1(0,012002291) = 0,019240083
17	8,5	z_17 = 0,012002291 + 0,5(0,019240083) = 0,021622333	0,374165141(0,043339607) - 0,1(0,021622333) = 0,014053937
18	9	z_18 = 0,021622333 + 0,5(0,014053937) = 0,028649302	0,357623865(0,036332899) - 0,1(0,028649302) = 0,010128582
19	9,5	z_19 = 0,028649302 + 0,5(0,010128582) = 0,033713593	0,341559317(0,030924806) - 0,1(0,033713593) = 0,007191296
20	10	z_20 = 0,033713593 + 0,5(0,007191296) = 0,037309241	

Figura 5. Tabla de aproximaciones eje z

Mediante las Figuras 3, 4 y 5 podemos apreciar como desde las posiciones (x,y,z) = (1,1,1) se tiene una aproximación hacia las coordenadas (x,y,z) = (0.326027592, 0.026782773, 0.037309241) en un tiempo de 10 segundos.

A continuación, se presenta el pseudocódigo del algoritmo implementado en Python para la resolución de las ecuaciones diferenciales de Lorenz.

```
INICIO

1. DEFINIR parámetros iniciales

 alpha, beta, gamma, pos = [ x(0), y(0), z(0)] , t

2. DEFINIR la función que representa las ecuaciones de Lorenz:

función deriv(pos, t, alpha, beta, gamma):
        x, y, z = pos
        dxdt = alpha * (y - x)
        dydt = x * (beta - z) - y
        dzdt = x * y - gamma * z
        return [dxdt, dydt, dzdt]

3. ESTABLECER las condiciones iniciales y duración de la simulación:

pos = [#, #, #] // x(0), y(0), z(0)

t = # s


4. LLAMAR a la función odeint para resolver el sistema:

   solución = odeint(deriv, pos, t, args= (alpha, beta, gamma))

5. EXTRAER los resultados:

   x = SOLUCION[:, 0]  # Valores de x(t)
   y = SOLUCION[:, 1]  # Valores de y(t)
   z = SOLUCION[:, 2]  # Valores de z(t)

6. VISUALIZAR los resultados:
   GRAFICAR x, y, z en un gráfico 3D


FIN
```

Figura 6. Pseudocodigo del algoritmo desarrollado para la solución del proyecto

Se puede notar que en la Figura 6 se especifica solo la implementación de la resolución de las ecuaciones de Lorenz y apenas se hace referencia de la implementación grafica del movimiento del nanodron, esto se debe a que no se consideró tan relevante la descripción del desarrollo de la interfaz gráfica, pero si se desea conocer se tiene el link del repositorio en GitHub en la parte de anexos del informe para un mayor entendimiento de la implementación gráfica.

2.2 Consideraciones importantes para el uso del código

Para el desarrollo del algoritmo se utilizaron varias librerías que no son parte de Python directamente, de modo que para una mayor eficiencia en la ejecución del algoritmo por parte del usuario se creó un apartado para que al ejecutarse se instalen todos los paquetes necesarios.

```
import sys
import subprocess

# Intentar instalar 'setuptools' si no está disponible
try:
    import pkg_resources
except ImportError:
    print("pkg_resources no está disponible. Instalando setuptools...")
    python = sys.executable
    subprocess.check_call([python, '-m', 'pip', 'install', 'setuptools'], stdout=subprocess.DEVNULL)
    import pkg_resources

from tqdm import tqdm

# Lista de paquetes necesarios
required = {'matplotlib', 'ipywidgets', 'numpy', 'PyQt5', 'scipy'}
installed = {pkg.key for pkg in pkg_resources.working_set}
missing = required - installed

if missing:
    print("Faltan los siguientes paquetes:")
    for package in missing:
        print(f" - {package}")

    # Instalación de los paquetes faltantes con barra de progreso
    print("\nInstalando paquetes...")
    for package in tqdm(missing, desc="Progreso"):
        python = sys.executable
        subprocess.check_call([python, '-m', 'pip', 'install', package], stdout=subprocess.DEVNULL)
    print("\nTodos los paquetes necesarios han sido instalados.")
else:
    print("Todos los paquetes necesarios ya están instalados.")
```


Figura 7. Algoritmo que instala los paquetes necesarios para el algoritmo principal

La ejecución del algoritmo presente en la Figura 7 puede tomar un tiempo dependiendo de las características del computador y su conexión a internet.

Adicionalmente, el algoritmo principal está acotado a un máximo de 100 segundos para el tiempo de simulación del movimiento del nanodron, debido a que durante el intervalo 0 – 100 se simulan 100 puntos para tener una simulación más fluida. No obstante, dicho parámetro puede ser modificado en el código fuente.

En adición, el proyecto solicitó que se tenga el vuelo del nanodron en diferentes posiciones iniciales donde los parámetros de (α, β, γ) y el tiempo t en segundos pueden o no ser los mismos. De modo que se creó apartados específicos para ambas trayectorias.

3. RESULTADOS:

Los siguientes casos de prueba (Caso 1 y Caso 2) fueron realizados mediante el algoritmo presente en el repositorio de GitHub, el link se encuentra en el apartado de anexos del informe.

3.1 Caso 1

En este caso, se establecieron las constantes:
α= 0.1
β = 0.1
γ = 0.1
Posiciones Iniciales Consideradas:
Posición A: $x = 1, y = 1, z = 1$
Posición B: $x = 0.9, y = 0.9, z = 0.9$
Tiempo de vuelo:
t = 10.00 segundos

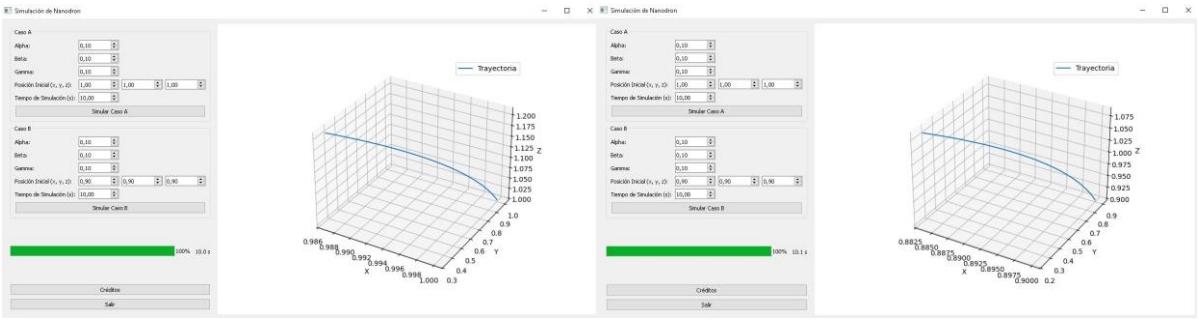

Figura 8. Gráficos del caso 1

En la Figura 8 se tiene a la izquierda el caso A correspondiente a la posición A, mientras que a la derecha se tiene el caso B correspondiente a la posición B.

```
[[1.         1.         1.        ]
 [0.9999905  0.98103128 1.00890948]
 [0.99996209 0.96216392 1.0176206 ]
 ...
 [0.32479711 0.01882839 0.04863451]
 [0.32449097 0.01880693 0.04864697]
 [0.32418513 0.01878548 0.0486593 ]]
[[0.9        0.9        0.9       ]
 [0.89999191 0.88383258 0.90713072]
 [0.89996769 0.86776308 0.91410901]
 ...
 [0.28637102 0.01126935 0.05853943]
 [0.28609578 0.01127534 0.05851314]
 [0.28582083 0.01128124 0.05848686]]
```

Figura 9. Caso 1: Puntos de la trayectoria (x, y, z)

El arreglo de datos presente en la Figura 9 corresponde a las aproximaciones de x, y, z para el tiempo t igual a diez segundos.

Descripción de Resultados:

Con estos parámetros, el sistema muestra una trayectoria estable y predecible, debido a los bajos valores de α, β y γ.

- Para la Posición A: El NANODRON sigue una trayectoria que tiende a estabilizarse, alcanzando un punto de equilibrio o un ciclo límite. Esto se refleja en una trayectoria suave y sin oscilaciones significativas.
- Para la Posición B: La trayectoria es casi idéntica a la de la Posición A, demostrando que el sistema es insensible a pequeñas variaciones en las condiciones iniciales. Ambas trayectorias convergen, reflejando un comportamiento predecible y estable del sistema bajo estos parámetros.

3.2 Caso 2

En este caso, los parámetros utilizados son:
$\alpha = 10$
$\beta = 28$
$\gamma = \frac{8}{3}$

Posición Inicial Considerada:
Posición A: $x = 1, y = 1, z = 1$
Posición B: $x = 0{,}9, y = 0{,}9, z = 0{,}9$

Tiempo de vuelo:
t = 10.00 segundos

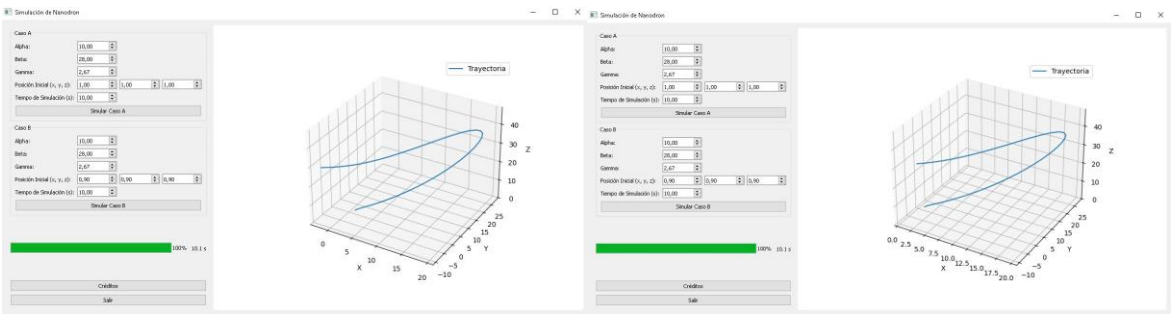

Figura 10. Gráficos del caso 2

En la Figura 10 se tiene a la izquierda el caso A correspondiente a la posición A, mientras que a la derecha se tiene el caso B correspondiente a la posición B.

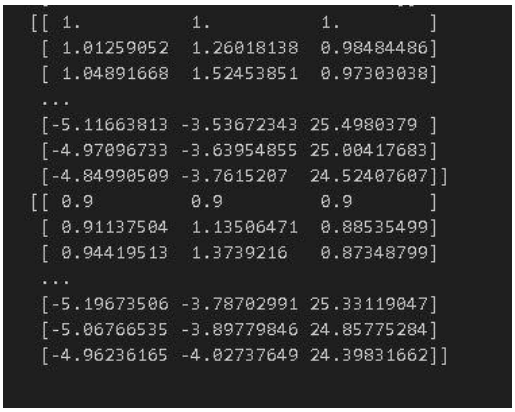

Figura 11. Caso 2: Puntos de la trayectoria (x, y, z)

El arreglo de datos presente en la Figura 11 corresponde a las aproximaciones de x, y, z para el tiempo t igual a diez segundos.

Descripción de Resultados:

Estos valores de parámetros corresponden al conocido sistema de Lorenz, que es famoso por su comportamiento caótico.

- Trayectoria Observada: A diferencia del primer caso, la trayectoria del NANODRON no sigue un camino predecible o repetitivo. En su lugar, oscila caóticamente alrededor de dos atractores extraños. Este tipo de comportamiento, caracterizado por su extrema sensibilidad a las condiciones iniciales, es una manifestación del caos. Así, pequeñas diferencias en las condiciones iniciales pueden llevar a trayectorias completamente diferentes con el tiempo.
- Además, se puede notar que el tipo de movimiento es el mismo para las posiciones A y B.

4. CONCLUSIONES

- El estudio del movimiento del NANODRON bajo las ecuaciones de Lorenz proporciona una herramienta poderosa para entender sistemas caóticos. La implementación interactiva facilita la exploración de diferentes escenarios, proporcionando una plataforma educativa valiosa.
- Al ejecutar el programa para los casos vistos en el apartado de resultados estos ilustran cómo diferentes configuraciones de parámetros α, β, γ pueden resultar en comportamientos del sistema que van desde trayectorias estables hasta complejas y caóticas. En el Caso 1, se observan trayectorias predecibles y estables, mientras que en el Caso 2, el sistema exhibe un comportamiento caótico, demostrando la naturaleza impredecible de los sistemas dinámicos no lineales.
- Los métodos numéricos de Euler, Adams-Bashforth-Moulton y Runge-Kutta son utilizados para la resolución de ODEs de primer grado, las cuales están presentes en

varios de los fenómenos naturales y científicos, de modo que su importancia es muy alta.

5. REFERENCIAS

Argote, J. I. (06 de abril de 2013). *Atractor Lorenz*. Obtenido de asociacionceat: https://www.asociacionceat.org/aw/2/atractor_lorenz.htm

6. Anexos

GitHub: https://github.com/R0berth456/Metodos_Numericos_ProyectoIIB.git

Documentación sobre LSODA: lsoda: Solver for Ordinary Differential Equations (ODE), Switching... in deSolve: Solvers for Initial Value Problems of Differential Equations ('ODE', 'DAE', 'DDE') (rdrr.io)

Documentación adicional sobre el LSODA odepack: netlib.org/odepack/opkdmain.f

Documentación adicional sobre el modelo matemático utilizado: https://www.bing.com/ck/a?!&&p=c805d2989c58ce47JmltdHM9MTcyMzY4MDAwMCZpZ3VpZD0zYTlmMmMzMy1lZjAzLTY2NTYtMmFlZS0zZmY2ZWViZDY3YTkmaW5zaWQ9NTE5NQ&ptn=3&ver=2&hsh=3&fclid=3a9f2c33-ef03-6656-2aee-3ff6eebd67a9&psq=Dialnet-MetodosNumericosRungekuttaYAdamsBashforthmoultonEn-7894523.pdf&u=a1aHR0cHM6Ly9kaWFsbmV0LnVuaXJpb2phLmVzL2Rlc2NhcmdhL2FydGljdWxvLzc4OTQ1MjMucGRm&ntb=1