Non-Linearity in Chua’s Circuit

Abstract:

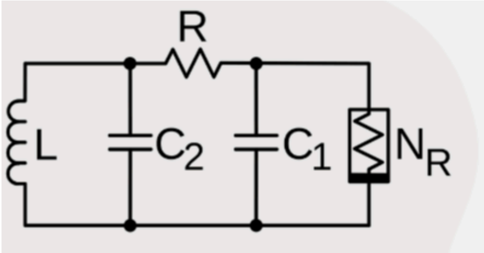
Chua’s Circuit is a fundamental example in nonlinear dynamics and chaos theory, demonstrating how simple electrical systems can exhibit complex, chaotic behavior. This report elaborates on the key aspects of Chua’s Circuit, including its governing equations, bifurcation characteristics, attractors, and the profound implications of chaos.

1. Introduction

Chua's Circuit is a simple electronic circuit that has linear passive components—resistors, capacitors, and inductors—paired with a single nonlinear component called Chua's diode. Chua's diode inserts a piecewise-linear nonlinearity into the circuit. In spite of its simple structure, Chua's Circuit can exhibit an extensive variety of nonlinear behavior such as bifurcations, limit cycles, and chaos. The chaotic dynamics were initially discovered by Professor Leon Chua in 1983 and have since been a seminal example used to study chaos theory, motivating many research investigations of nonlinear systems and attractors.

2. Theoretical Background

The dynamics of Chua's Circuit are determined by a system of three coupled, first-order, nonlinear differential equations describing the evolution of two capacitor voltages and and the inductor current :

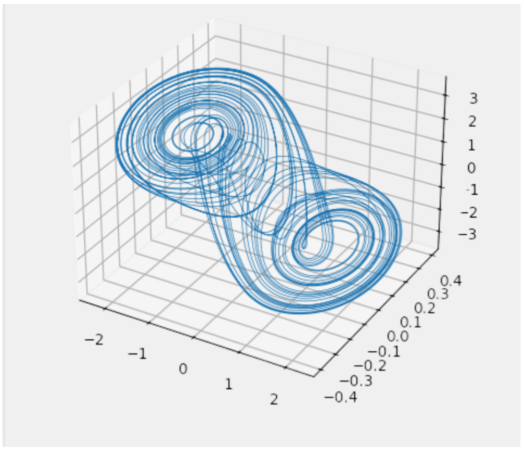


$$C_1\frac{dv_1}{dt} = \frac{1}{R}(v_2 - v_1) - g(v_1)$$
$$C_2\frac{dv_2}{dt} = \frac{1}{R}(v_1 - v_2) + i_L$$
$$L\frac{di_L}{dt} = -v_2$$

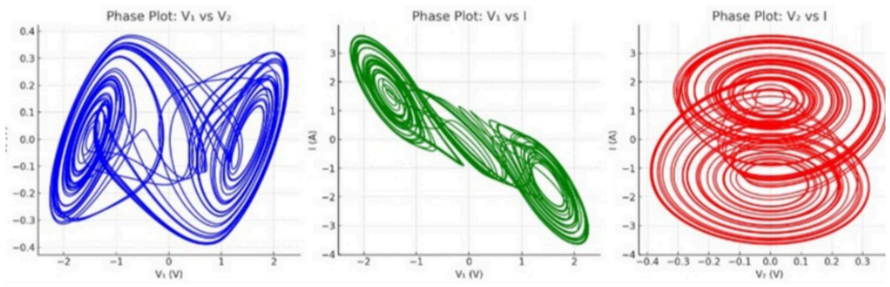
$$g(v_1) = \begin{cases} m_1 v_1 + (m_0 - m_1)E, & \text{if } v_1 \leq -E \\ m_0 v_1, & \text{if } -E < v_1 < E \\ m_1 v_1 + (m_0 - m_1)(-E), & \text{if } v_1 \geq E \end{cases}$$

Here, is the piecewise-linear nature of Chua's diode, introducing nonlinearity in the form of a segmented linear function that changes slope depending on the voltage. The constants are dimensionless constants that are functions of the circuit elements (resistances, capacitances, and inductance). These equations collectively form the basis for the emergence of complex dynamical behaviors ranging from stable equilibria to fully developed chaos.

3. Chua Attractor and Phase Space



The Chua attractor is a good instance of a "strange attractor," a bounded orbit in three-dimensional phase space. If the system is seeded with the conditions and simulated for 10,000 time steps, it yields a characteristic double-scroll attractor. The attractor takes the form of two intertwined scroll-like patterns, indicating the sensitive dependence on initial conditions and aperiodic, bound motion. Visualization of phase space reveals fundamental insights into the evolution of trajectories, stretch, and folding, similar to stretch and fold dynamics of chaotic systems.



4. Bifurcations and Fixed Points

When the parameters of the circuit (and the slopes) vary, the system experiences bifurcations—qualitative transitions from one state to another. A bifurcation diagram graphs how the circuit behavior evolves as parameters change, showing transitions from fixed-point to periodic oscillations and then to chaotic oscillations. Fixed points of the system are the states at which all the derivatives are zero (steady states). The stability of the fixed points can be studied using linearization methods, like Jacobian matrix analysis, to see whether trajectories converge or diverge to or from these points. Saddle points, stable nodes, and unstable foci are all possible according to the parameter regime.

NORMALIZED EQUATIONS

$$\dot{x} = \alpha(y - x - g(x))$$
$$\dot{y} = x - y + z$$
$$\dot{z} = -\beta y$$

5. Applications and Implications

The research on Chua's Circuit goes far beyond scholarly interest. The underlying principles of its nonlinear dynamics and chaos have applications in secure communication, random number generation, biological neural networks, and even the study of natural phenomena such as weather patterns and cardiac rhythms. The double-scroll attractor has been implemented in many real-world systems, proving that chaos is not an abstract curiosity but a property common to many natural and engineered systems.

6. Conclusion

Chua's Circuit beautifully illustrates that chaotic, complex dynamics can arise from deterministic systems based on simple rules. With its rich phase portraits, bifurcation behaviors, and strange attractors, it is a prototype for nonlinear dynamical systems. The lessons learned from its study persist in educating a wide range of scientific and engineering fields, making it a pillar in chaos theory.

7.Code for the attractor

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from mpl_toolkits.mplot3d import Axes3D

# Chua's circuit parameters
alpha = 15.6
beta = 28
```

```
m0 = -1.143
m1 = -0.714

# Nonlinear function
def f(V):
    return m1 * V + 0.5 * (m0 - m1) * (np.abs(V + 1) - np.abs(V - 1))

# Chua system equations (using V, R, I)
def chua(V, R, I):
    dV = alpha * (R - V - f(V))
    dR = V - R + I
    dI = -beta * R
    return np.array([dV, dR, dI])

# Runge-Kutta 4th order integration
def rk4(state, dt):
    k1 = chua(*state)
    k2 = chua(*(state + 0.5 * dt * k1))
    k3 = chua(*(state + 0.5 * dt * k2))
    k4 = chua(*(state + dt * k3))
    return state + (dt / 6) * (k1 + 2*k2 + 2*k3 + k4)

# Simulation parameters
dt = 0.1
steps = 1000

# Three initial conditions
init1 = np.array([0.71, 0.0, 0.0])
init2 = np.array([0.7, 0.0, 0.0])
init3 = np.array([0.72, 0.0, 0.0])

# Precompute trajectories
traj1 = np.zeros((steps, 3))
traj2 = np.zeros((steps, 3))
traj3 = np.zeros((steps, 3))
s1, s2, s3 = init1.copy(), init2.copy(), init3.copy()

for i in range(steps):
    traj1[i] = s1
    traj2[i] = s2
    traj3[i] = s3
    s1 = rk4(s1, dt)
    s2 = rk4(s2, dt)
    s3 = rk4(s3, dt)

V1, R1, I1 = traj1.T
V2, R2, I2 = traj2.T
V3, R3, I3 = traj3.T

# Plotting
fig = plt.figure(figsize=(8, 5))
ax = fig.add_subplot(111, projection='3d')
ax.set_xlim(-2, 2)
ax.set_ylim(-2, 2)
ax.set_zlim(-30, 30)
ax.set_title("Chua's Circuit Phase Portrait (Three Trajectories)",
fontsize=14)
ax.set_xlabel('V1 (Voltage across C1)', fontsize=12)
ax.set_ylabel('V2 (Voltage across C2)', fontsize=12)
ax.set_zlabel('I (Current through L)', fontsize=12)

# Set background color
fig.patch.set_facecolor('white')
ax.set_facecolor('white')

# Set a better viewing angle (Azimuth and Elevation)
ax.view_init(elev=30, azim=60)

# Line properties
line1, = ax.plot([], [], [], color='deepskyblue', label='Init 1',
linewidth=0.7)
line2, = ax.plot([], [], [], color='tomato', label='Init 2',
linewidth=0.7)
line3, = ax.plot([], [], [], color='yellowgreen', label='Init 3',
linewidth=0.7)
point1, = ax.plot([], [], [], 'bo', markersize=5)
point2, = ax.plot([], [], [], 'ro', markersize=5)
point3, = ax.plot([], [], [], 'go', markersize=5)
ax.legend()
```

```python
def init():
    line1.set_data([], [])
    line1.set_3d_properties([])
    line2.set_data([], [])
    line2.set_3d_properties([])
    line3.set_data([], [])
    line3.set_3d_properties([])
    point1.set_data([], [])
    point1.set_3d_properties([])
    point2.set_data([], [])
    point2.set_3d_properties([])
    point3.set_data([], [])
    point3.set_3d_properties([])
    return line1, line2, line3, point1, point2, point3

def update(frame):
    if frame >= steps:
        ani.event_source.stop()  # Stop the animation after the last frame

        # Save the final image
        plt.savefig("chua_final_image.png")  # Save the image here

        return line1, line2, line3, point1, point2, point3

    line1.set_data(V1[:frame], R1[:frame])
    line1.set_3d_properties(I1[:frame])
    line2.set_data(V2[:frame], R2[:frame])
    line2.set_3d_properties(I2[:frame])
    line3.set_data(V3[:frame], R3[:frame])
    line3.set_3d_properties(I3[:frame])

    point1.set_data([V1[frame]], [R1[frame]])
    point1.set_3d_properties([I1[frame]])
    point2.set_data([V2[frame]], [R2[frame]])
    point2.set_3d_properties([I2[frame]])
    point3.set_data([V3[frame]], [R3[frame]])
    point3.set_3d_properties([I3[frame]])

    return line1, line2, line3, point1, point2, point3
```

```
# Create a 5 second delay before starting the animation
ani = FuncAnimation(fig, update, frames=steps, init_func=init,
                    interval=50, blit=False)

# Add a 5-second delay before starting the animation
ani.event_source.start()
 # Add 5000 ms (5 seconds) delay

plt.tight_layout()
plt.show()
```