Security
of Infocommunication Systems
and Internet of Things

2024 Vol 2, No 1

https://doi.org/10.31861/sisiot2024.1.01003

Received 17 June 2024; revised 07 August 2024; accepted 22 August 2024; published 30 August 2024

Investigation of the Influence of Computation Accuracy in the Implementation of Chaotic Systems in Python for Secure Telecommunication Systems

Mykola Kushnir, Hryhorii Kosovan* and Vladyslav Melnyk

Dept. of Radio Engineering and Information Security, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: g.kosovan@chnu.edu.ua)

ABSTRACT This article focuses on the use of the Python programming language to visualize chaotic models and for the investigation of the influence of initial conditions in physical systems, in particular, the Chua, Lorenz, and Rössler models. Chaotic systems are dynamic and sensitive to initial conditions, making them unpredictable as to how they will behave and react. This means that in the long run, very different outcomes can result from even small changes in initial conditions. Chaotic systems are studied in a variety of scientific fields, including physics, mathematics, biology, engineering and economics. Python, the world's most popular scientific programming language, transforms complex models into intuitive visualizations. The paper reveals the capabilities of various Python algorithms and libraries used to visualize these models, taking into account their specifics. The article focuses on three chaotic models: the Chua system, which is a universal example of a chaotic system; the Lorenz attractor, which is famous for its chaotic properties; and the Rössler rotational oscillator, which is widely used in such fields as biology, chemistry, physics, and engineering. Each model is studied in detail, its key characteristics and parameters are presented, and graphs of these models are displayed by means of Python simulation. Python, due to its ease of use and high performance, makes it possible to solve such tasks quickly and efficiently. Finally, the authors share their conclusions on the importance of initial conditions for Lorenz, Rössler and Chua systems, as well as their impact on telecommunication systems. This study provides insight into how Python, a programming language with a high level of abstraction, allows for the rapid and efficient development of complex algorithms and models needed to deal with chaotic systems. It also allows researchers and engineers to develop efficient algorithms for signal processing and control of telecommunication systems.

KEYWORDS visualization of chaotic models, Python, Chua, Lorenz and Rössler models, dynamic systems, chaotic systems.

I. INTRODUCTION

In the beginning, according to ancient Greek theology and philosophy, was chaos [1]. Greek philosophers believed that our ordered universe, the cosmos, was formed from this chaos. The ancient Greeks did not give a specific definition of chaos, although it was associated with infinity, disorder, and unpredictability. The Greeks believed that disorder could lead to order under certain conditions, which modern science discovered centuries later.

Chaos is not just a disorder. It is a dynamic and self-organizing world, where by chance amazing patterns are born, and by instability - stable structures [2]. Chaotic systems exist in all areas of life, from weather and climate to the evolution of human life and behavior. Imagine a swinging pendulum. It would seem that its motion is simple and straightforward. However, add a little vibration and the pendulum is already swinging chaotically from side to side, defying any predictions. This example illustrates the essence of chaos: even in simple systems there is unpredictability, which makes their behavior complex and fascinating.

Chaotic systems have fascinated scientists and mathematicians for decades due to their complex and unpredictable behavior. The history of the study of chaotic systems begins in the XIX century with the works of Poincaré, who studied the three-body problem and discovered unpredictable trajectories in its solutions that remained stable for a long time [3].

It was discovered that even small changes in the initial conditions can cause significant differences in the subsequent trajectories, sometimes with unpredictable results. This was a defining discovery that created the calm before the storm that soon erupted in the heart of science. At this point in history, although the first indications of chaos in dynamical systems had appeared, science was not yet ready to accept such a radical idea. In fact, with his work, Poincaré discovered the concept of "deterministic chaos," the idea that even in a deterministic system, where everything is determined by the laws of physics, complex, unpredictable behaviors can occur. This is due to the sensitivity to initial conditions that characterizes chaotic systems. However, this great idea waited almost a century to be fully realized. It was when Edward Lorenz made radical discoveries in meteorology in the 1960s that drew our attention to what we can see not only in complex systems but also in very simple mathematical models [4].

When delving into the world of chaotic systems, it is impossible to avoid the models of Chua, Lorenz, and Rössler [3, 5-10]. They demonstrate how simple dynamical systems can create complex, unpredictable patterns by combining elements of order and chaos in their trajectories.

These classical chaos models have the ability to display striking patterns of order and chaos, even in their simplest mathematical form. The Lorenz model is a model of the atmosphere developed to study convection, i.e. heat transfer in liquids or gases. The Rössler model represents one of the simplest three-dimensional systems that can show chaotic behavior. The Chua model is designed to show chaos in a simple electrical oscillating circuit. Using Python to study these models opens up new possibilities for science. Python, with its powerful libraries such as NumPy for computation and Matplotlib for visualization, provides researchers with the tools to model, analyze, and visualize these complex systems.

II. DESCRIPTION OF THE CHUA CHAOTIC SYSTEM

A major advance in chaos theory, was developed by Leon O. Chua in 1992 [5]. They developed a Chua circuit, a this self-contained electronic circuit that is known for its ability to generate a variety of dynamic phenomena, making it a classic example of true chaos. At the center of the system is the Chua diode, a nonlinear resistor that allows the circuit to exhibit chaotic behavior. The Chua system, with resistors, capacitors, and Chua diodes, is a simple but powerful model for studying chaotic theory. Its behavior can be described for three combinations of first-order nonlinear differential equations. The mathematical model of the Chua circle is as follows:

$$\begin{cases} \frac{dx}{dt} = \alpha * \left(y - x - h(x)\right), \\ \frac{dy}{dt} = x - y + z, \\ \frac{dz}{dt} = -\beta * y. \end{cases} \quad (1)$$

Where α, β are system parameters that determine the behavior of the system. And $h(x)$ is a piecewise linear function that introduces nonlinearity:

$$h(x) = m_1 z + \frac{1}{2}(m_0 - m_1)(|x+1| - |x-1|) \quad (2)$$

Diagram of Chua's system shows chaotic behavior. The attractors are seen in phase space and are often referred to as "Chua attractors". Chua's system can exhibit time-doubling bifurcations, chaos synchronization, and other types of complex behavior, which can be seen in the simulation diagram (see Fig. 1).

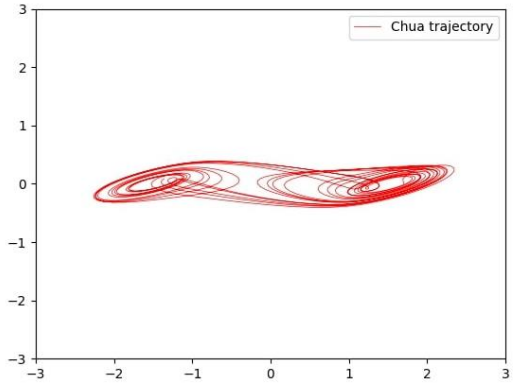

FIG. 1. Output image showing the simulation results for the Chua system.

Chua's scheme has had practical applications in a variety of industries:

- Random number generation: A Pseudo Random Number Generator Based on the Chaotic System of Chua's Circuit and its Real Time FPGA Implementation [11];

- Secure communication: Secure communication using a chaos based signal encryption scheme [12].

Fig. 1, shows the result of the implementation of the Chua chaotic system in Python.

We can gain insight into the underlying dynamics of chaotic systems and explore the rich behavior of this iconic circuit by using Python to model Chua's system.

III. DESCRIPTION OF THE LORENZ CHAOTIC SYSTEM

In 1963, a meteorologist and mathematician Edward N. Lorenz introduced a set of three ordinary differential equations (ODEs) known as the Lorenz system [6, 10]. This system is a paradigmatic example of nonlinear continuous dynamic systems that exhibit deterministic chaos, characterized by sensitivity to initial conditions, often referred to as the "butterfly effect".

The Lorenz system was originally derived as a simplified model of atmospheric convection, intended to capture the main features of fluid flow that lead to chaotic behavior. To create this model, Lorenz built upon earlier concepts proposed by Saltzman B., developing a two-layer model that approximates atmospheric motion. In this model, the fluid, which represents a gas, encompasses the upper and lower atmosphere [13-19]. Different constant temperatures are considered in each layer, generating an external force that drives convection. When the temperature gradient is sufficiently large, convection becomes turbulent.

For ease of simulation, Lorenz reduced the partial differential equations to a simple system with three differential equations. Although the quadratic polynomial vector field is relatively simple, the associated dynamics are highly complex and the mechanisms that lead to chaos have been the subject of considerable research.

The equations describing the Lorenz system are as follows:

$$\begin{cases} \frac{dx}{dt} = \sigma * (y - x), \\ \frac{dy}{dt} = x * (\rho - z) - y, \\ \frac{dz}{dt} = x * y - \beta * z. \end{cases} \quad (3)$$

Here, x, y and z are state variables representing the state of the system at any given time, and [σ, ρ, β] are parameters. The values of these parameters are often used to demonstrate chaotic behavior.

Lorenz system can be used to model and study various phenomena in different fields, including:

- Meteorology and Climatology: Effect of averaging timescale on a forced Lorenz model [13];

- Physics: A Nonlinear Oscillator Derived from the Lorenz Chaotic System [14];

- Engineering: Chaos control and synchronization of modified Lorenz system using active control and backstepping scheme [15].

Fig. 2, shows the result of the implementation of the Lorenz chaotic system in Python.

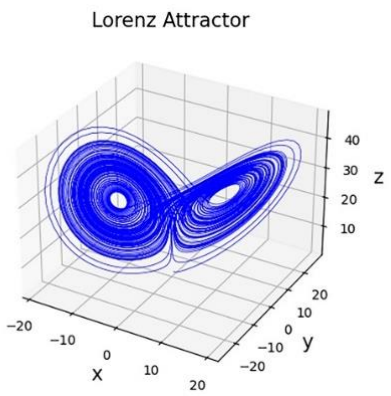

FIG. 2. Output image showing the simulation results for the Lorenz system.

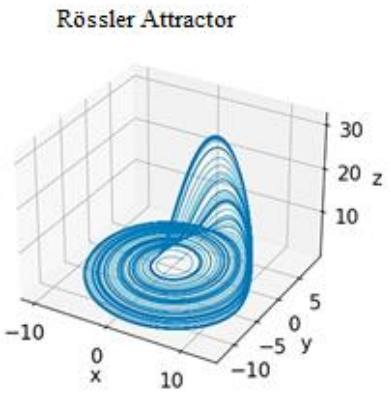

FIG. 3. Output image showing the simulation results for the Rössler system.

IV. DESCRIPTION OF THE RÖSSLER CHAOTIC

The next mathematical model of chaotic behavior is the Rössler system, which is considered together with the Lorenz system classic examples of chaotic systems [7]. Rössler system was proposed in 1976 by the German scientist Otto Rössler (Otto E. Rössler) for description of chemical reactions occurring in the reactor with mixing. The system consists of three differential equations of the first order and contains only one nonlinearity:

$$\begin{cases} \frac{dx}{dt} = -y - z, \\ \frac{dy}{dt} = x + ay, \\ \frac{dz}{dt} = b + z * (x - c), \end{cases} \quad (4)$$

where x, y and z are state variables, and $[a, b, c]$ are parameters. The system exhibits chaotic behavior at certain values of the parameters, which makes it an interesting object for studying chaos theory [8]. The Rössler system has been used to model a wide range of phenomena, including chemical reactions, population dynamics, and electrical circuits. It is also used as a teaching tool to introduce students to the concepts of chaos theory and nonlinear dynamics. The Rössler system differs from the Lorenz system by the type of the vector field of phase trajectories. In the Lorenz system the divergence is constant at any point in the phase space, negative and independent of the control parameter, while in the Rössler system the divergence is not constant and depends on the variable x and the control parameter c.

The Rössler system is used in:

- Chemical systems: Chaos control for Willamowski–Rössler model of chemical reactions [16];
- Engineering and Electronics: Chaos control for the family of Rössler systems using feedback controllers [17];
- Biology and Medicine: Chaos Control Applied to Heart Rhythm Dynamics [18].

Fig. 3, shows the result of the implementation of the Rössler chaotic system in Python.

V. DEPENDENCE OF INITIAL CONDITIONS FOR THE LORENZ, RÖSSLER AND CHUA SYSTEMS

Python offers a plethora of data types, including float, int, str, dict, list, tuple, set, and others [9]. Each data type has its own specific applications and usage. This discussion will focus on the first two, float and int. It is obvious that chaotic signals are pretty dependent on the initial conditions. Let us consider the same data, however represented with different “int” and “flow” data kinds, For example we can call “initial_conditions_int” and “initial_conditions_float” and show how minor changes in the data representation can significantly affect the computational results, especially in cases with high sensitivity to initial conditions, as is observed in chaotic systems.

Fig. 4 shows a code snippet that demonstrates the implementation of testing initial conditions for a chaotic Lorenz system using numbers of type Int and float. For other chaotic systems, the code structure remains similar, but the corresponding equation will be adapted to the specifics of each particular system.

```python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
sigma, beta, rho = 10, 2.667, 28
def lorenz(X, t, sigma, beta, rho):
    x, y, z = X
    return sigma * (y - x), x * (rho - z) - y, x * y - beta * z
t = np.linspace(0, 100, 10000)
def run_simulation(initial_conditions):
    return odeint(lorenz, initial_conditions, t, args=(sigma, beta, rho)).T
# Initial conditions
x_float, y_float, z_float = run_simulation((0, 1, 1.015))
x_int, y_int, z_int = run_simulation((0, 1, int(1.015)))
# Plot setup
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
line_float, = ax.plot([], [], [], 'b-', lw=0.5, label='Float ICs')
line_int, = ax.plot([], [], [], 'r-', lw=0.5, label='Int ICs')
ax.legend()
ax.set_xlim([-20, 20])
ax.set_ylim([-30, 30])
ax.set_zlim([0, 30])
# Animation functions
def init():
    for line in [line_float, line_int]:
        line.set_data([], [])
        line.set_3d_properties([])
    return line_float, line_int
def update(frame):
    line_float.set_data(x_float[:frame], y_float[:frame])
    line_float.set_3d_properties(z_float[:frame])
    line_int.set_data(x_int[:frame], y_int[:frame])
    line_int.set_3d_properties(z_int[:frame])
    return line_float, line_int
ani = FuncAnimation(fig, update, init_func=init, frames=len(t), interval=5, blit=True)
plt.show()
```

FIG. 4. Code for testing initial conditions with Int and float type.

Figures 4 and 11 used the following set of Python libraries: the NumPy library provided efficient work with large multidimensional arrays and mathematical functions, while the SciPy library extended the capabilities of NumPy to perform complex scientific and technical calculations, including the integration of differential equations using the odeint function. To visualize the results, the Matplotlib library, which provides a complete toolkit for creating various static, animated, and interactive visualizations, was used. This set of libraries ensured effective modeling and visualization of chaotic systems.

As noted earlier, chaotic systems are sensitive to initial conditions. When the initial conditions of a chaotic system are switched from floating point to integer numbers, this causes a modification of the initial point in phase space. Therefore, a small change can lead to essentially different trajectories over time.

When executing the code, we first observed that the trajectories of the floating-point system and the integer initial conditions coincided. However, over time, the behavior of the system began to differ significantly, which makes long-term prediction of the system's behavior impossible. The differences between integer modeling and floating-point modeling illustrate the sensitivity of the chaotic system to the initial conditions and their smallest difference during repeated solutions.

Since even the smallest deviations in quantification or rounding can cause drastically different results, we can assume that rounding leads to a significant loss of accuracy, which has more profound consequences in chaotic systems than in more stable systems.

In this example, we illustrated the "Butterfly Effect", a concept that describes how small reasons can have large outcomes, which includes how a flap of a butterfly's wings in China can trigger a tornado in Africa. Converting a floating point number to an integer is one such small reason that can completely change the behavior of a chaotic system.

Figures 5 and 6 show the beginning and end of the simulation result for the Lorenz system for the initial conditions given with Int and float types.

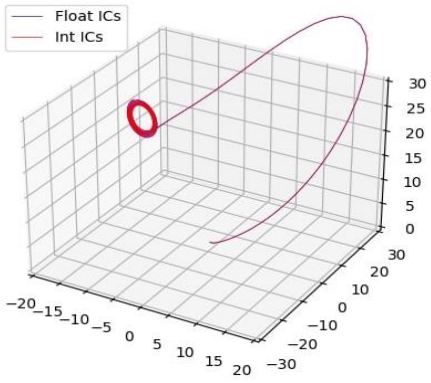

FIG. 5. Output image showing the output of the initial simulation for the Lorenz system for the initial conditions given with Int and float types.

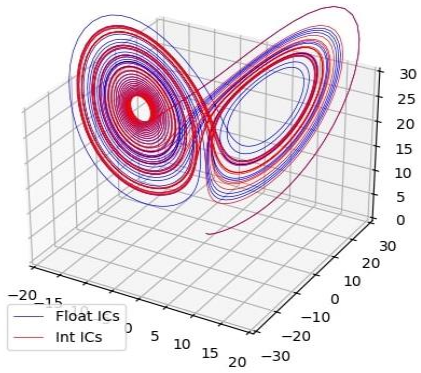

FIG. 6. Output image showing the end of simulation result for the Lorenz system for the initial conditions given with Int and float types.

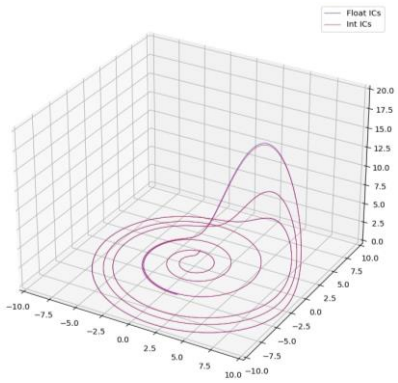

FIG. 7. Output image showing the start of simulation result for the Rössler system for the initial conditions given with Int and float types.

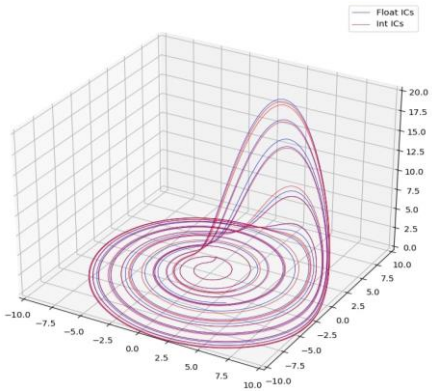

FIG. 8. Output image showing the end of simulation result for the Rössler system for the initial conditions given with Int and float types.

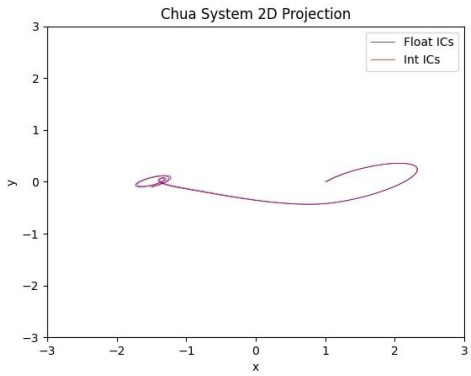

FIG. 9. Output image showing the initial simulation results for the Chua system for initial conditions with different decimal places.

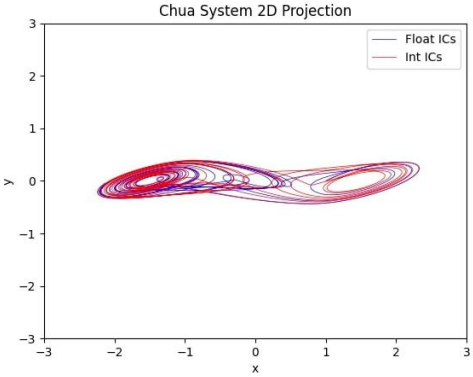

FIG. 10. Output image showing the final simulation results for the Chua system for initial conditions with different decimal places.

Figures 7 and 8 display the start and end of the simulation results for the Rössler system, using integer and floating-point initial conditions. Figures 9 and 10 present the initial and final simulation results for the Chua system, with integer and floating-point initial conditions.

The results of the Lorenz, Rössler and Chua systems simulation (Figures 6, 8, 10) show that after a certain number of iterations, trajectories with different accuracies begin to diverge. Over time, this difference will only increase, indicating that without proper attention to the accuracy of calculations, the implementation of such systems in real hardware can lead to significant deviations in the operation of communication systems built on the basis of chaotic systems.

The next step in the study is to check how the initial conditions change the modeling results with different accuracy. Fig. 11 shows the code for testing the initial conditions in the Lorenz system using different numbers of decimal places. This allows us to assess the impact of numerical precision on the dynamics of a chaotic system. For other chaotic systems, the structure of the code remains the same, but the system function must be adapted to the specifics of each particular system. To do this, we need to create a function that will round the initial conditions for each level of accuracy to 1, 2, and 3 decimal places, respectively.

```python
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

sigma, beta, rho = 10, 2.667, 28
def lorenz(X, t, sigma, beta, rho):
    x, y, z = X
    return sigma * (y - x), x * (rho - z) - y, x * y - beta * z
t = np.linspace(0, 100, 10000)
original_ic = (0, 1, 1.015)
precisions = [None, 1, 2, 3]
colors = ['black', 'red', 'green', 'blue']
def run_simulation(ic):
    return odeint(lorenz, ic, t, args=(sigma, beta, rho)).T
def adjust_precision(ic, precision):
    return tuple(np.round(c, precision) for c in ic) if precision is not None else ic
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
lines = []
for precision, color in zip(precisions, colors):
    ic = adjust_precision(original_ic, precision)
    x, y, z = run_simulation(ic)
    line, = ax.plot(x, y, z, lw=0.5, color=color, label=f'{precision} decimal: ICs={ic}')
    lines.append((line, (x, y, z)))
ax.set_xlim([-20, 20])
ax.set_ylim([-30, 30])
ax.set_zlim([0, 50])
ax.legend()
def init():
    for line, _ in lines:
        line.set_data([], [])
        line.set_3d_properties([])
    return [line[0] for line in lines]
def update(frame):
    for line, (x, y, z) in lines:
        line.set_data(x[:frame], y[:frame])
        line.set_3d_properties(z[:frame])
    return [line[0] for line in lines]
ani = FuncAnimation(fig, update, init_func=init, frames=len(t), interval=10, blit=True)
plt.show()
```

FIG. 11. Code for testing initial conditions with different decimal places.

The initial and final simulation results for the Lorenz system with initial conditions with different decimal places are shown in Figures 12 and 13. Figures 14 and 15 illustrate the initial and final stages of the Rössler system simulation, with the initial conditions set to different decimal places. Figures 16 and 17 illustrate the initial and final stages of the Chua system simulation, with the initial conditions set to different decimal places.

The simulation results of the Lorenz, Rössler and Chua systems (Figures 13, 15, 17) demonstrate the impact of varying numerical precision on the dynamics of chaotic systems. In particular, the simulation results indicate that the trajectories of systems with differing numbers of decimal places begin to diverge after a specific number of iterations. With increasing time this divergence increases.

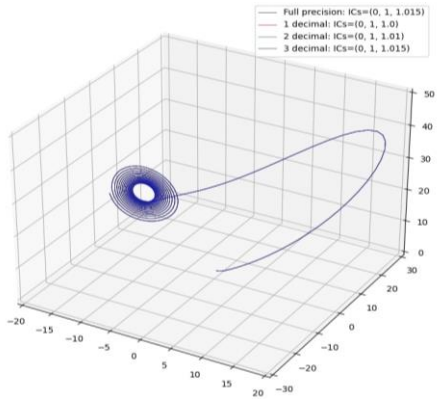

FIG. 12. Output image showing the initial simulation results for the Lorenz system for initial conditions with different decimal places.

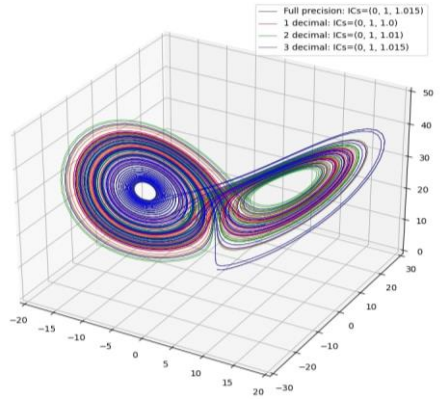

FIG. 13. Output image showing the final simulation results for the Lorenz system for initial conditions with different decimal places.

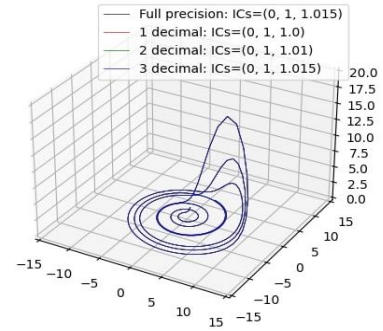

FIG. 14. Output image showing the start of simulation result for the Rössler system for initial conditions with different decimal places.

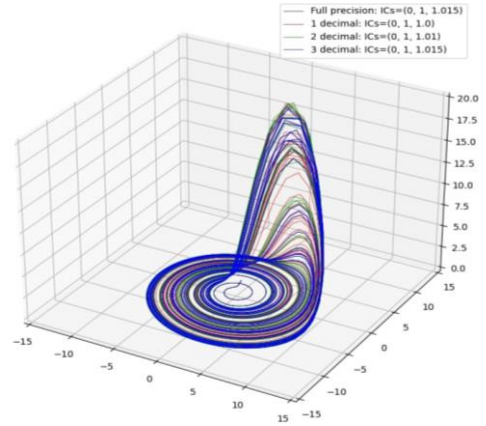

FIG. 15. Output image showing the end of simulation result for the Rössler for initial conditions with different decimal places.

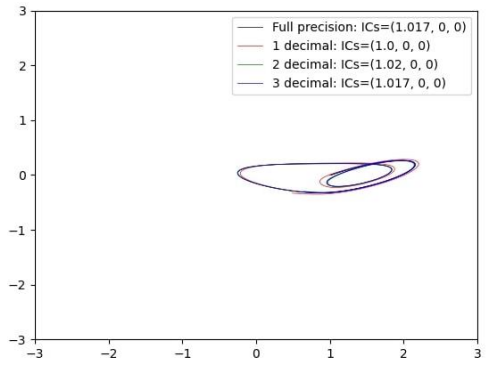

FIG. 16. Output image showing the start of simulation result for the Chua system for initial conditions with different decimal places.

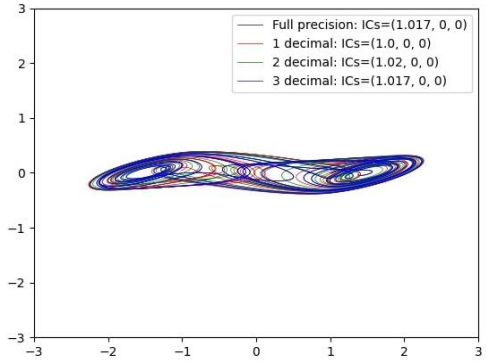

FIG. 17. Output image showing the end of simulation result for the Chua system for initial conditions with different decimal places.

The graphs showed a divergence of trajectories. As expected in chaotic systems, the modified trajectories differed over time from the initial trajectory, or in other words, the trajectory of full accuracy. This divergence is related to the sensitivity of the Lorenz system, where even small differences can quickly amplify, significantly altering the evolution of the system. An important observation was the speed of divergence of each trajectory from the reference behavior of full fidelity. Trajectories with more decimal places (more precise rounding) followed the reference more closely for a longer period of time before diverging, while trajectories with fewer decimal places (less precise rounding) diverged more quickly.

VI. CONCLUSION

Using Python simulations, we have studied the Chua, Lorenz and Rössler systems, classic examples of chaotic dynamics. By modelling these systems and visualizing their trajectories, we observed the emergence of complex behavior. Python proved to be a versatile tool for studying chaotic phenomena, offering insights into the sensitivity to initial conditions, the formation of characteristic patterns, and the dynamics of strange attractors. In microelectronics and in the context of digital systems and integrated circuits, floating point truncation to integer and rounding of numbers to 1, 2, 3 decimal places can be analogous to quantization errors in analogue-to-digital conversion, where a continuous range of values must be represented by discrete levels due to the nature of digital systems. Such changes can affect the accuracy of systems and are a critical factor that engineers must consider. For example, it can affect timing accuracy, signal processing in a way that can potentially affect the reliability of telecommunications systems. In this way, Lorenz, Chua and Rössler's attractor experiments serve as a metaphor for the importance of accuracy in telecommunications systems: they illustrate how small changes in conditions or parameters can have a significant, sometimes unpredictable, impact on system behavior. This highlights the need for a careful and meticulous approach to the design and testing of telecommunications systems to ensure their reliability.

ACKNOWLEDGMENT

We acknowledge the contributions of our colleagues and research peers who have engaged in thoughtful discussions, shared valuable perspectives, and provided constructive feedback. The diverse range of ideas has enriched the depth and breadth of our exploration. We also express our most sincere gratitude to all the colleagues whose research papers are mentioned in References.

AUTHOR CONTRIBUTIONS

M.K. – conceptualization, methodology; V.M. – writing-original draft preparation, writing the code of the software implementation; H.K. – formal analysis, investigation.

COMPETING INTERESTS

There are no competing interests.

REFERENCES

[1] Ovid, *Metamorphoses*, translated by Anthony S. Kline, 2000.
[2] S. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, 2014.
[3] T. N. Palmer, "Edward Lorenz (1917-2008)," *Nature*, vol. 453, no. 7196, pp. 583-584, 2008.
[4] S. Katzir, "Poincaré's Relativistic Physics: Its Origins and Nature," *Phys.*, 2005.
[5] L. O. Chua, "The Genesis of Chua's Circuit," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, pp. 885-902, 1992.
[6] E. N. Lorenz, *The Essence of Chaos*. University of Washington Press, 1993.
[7] O. E. Rössler, "Chaos in Abstract Kinetics: Two Prototypes," *Bulletin of Mathematical Biology*, vol. 39, no. 2, pp. 275-289, 1977.
[8] C. Letellier and V. Messager, "Influences on Otto E. Rössler's Earliest Paper on Chaos," *International Journal of Bifurcation and Chaos*, vol. 20, no. 11, pp. 3585-3616, 2010.
[9] Python Documentation, "Data Types," *Python 3.x Documentation*, 2024.
[10] E. N. Lorenz, "Deterministic Non-Periodic Flow," *Journal of the Atmospheric Sciences*, pp. 130-141, 1963.
[11] K. Murali, V. Varadan, and H. Leung, "Secure Communication Using a Chaos-Based Signal Encryption Scheme," 2001.
[12] L. Merah, A. Ali Pacha, N. Hadj Said, and M. Mamat, "A Pseudo Random Number Generator Based on the Chaotic System of Chua's Circuit, and Its Real-Time FPGA Implementation," 2013.
[13] S. Dwivedi, A. Pandey, and A. Mittal, "Effect of Averaging Timescale on a Forced Lorenz Model," 2007.
[14] P. Stamenov and T. Spassova, "A Nonlinear Oscillator Derived from the Lorenz Chaotic System," 2001.
[15] M. T. Akter, A. Tarammim, and S. Hussen, "Chaos Control

and Synchronization of Modified Lorenz System Using Active Control and Backstepping Scheme," 2023.
[16] I. Bodale and V. A. Oancea, "Chaos Control for Willamowski–Rössler Model of Chemical Reactions," 2015.
[17] X. Liao and P. Yu, "Chaos Control for the Family of Rössler Systems Using Feedback Controllers," 2006.
[18] B. B. Ferreira, A. S. de Paula, and M. Savi, "Chaos Control Applied to Heart Rhythm Dynamics," 2011.
[19] B. Saltzman, "Finite Amplitude Free Convection as an Initial Value Problem–I," 1962.

Hryhorii Kosovan

Born in Ukraine in 1985. Received the PhD degree in 2019. An assistant at the Department of Radio Engineering and Information Security of Yuriy Fedkovych Chernivtsi National University. Research interests include chaos theory, secure telecommunications networks.

ORCID ID: 0000-0002-3351-3852

Mykola Kushnir

Associate Professor of the Department of Radio Engineering and Information Security. 19 Scopus documents, h-index -5.
Two Erasmus grants - Iasi - 2014-2015, Valencia - 2015. Two CRDF grants - 2022 and 2023. State Order "For Courage" (III degree).

ORCID ID: 0000-0001-9480-3856

Vladyslav Melnyk

Born in 1999 in Kamianets-Podilskyi, Ukraine. Entered Yuriy Fedkovych Chernivtsi National University in September 2016 as a student of the Department of Radio Engineering and Information Security. Since September 2023 – a PhD student of the department.

ORCID ID: 0009-0004-9847-2635

Дослідження впливу точності обчислення на реалізацію хаотичних систем на Python для захищених телекомунікаційних систем

Микола Кушнір, Григорій Косован*, Владислав Мельник

Кафедра радіотехніки та інформаційної безпеки, Чернівецького національного університету імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: g.kosovan@chnu.edu.ua)

АНОТАЦІЯ Ця стаття присвячена використанню мови програмування Python для візуалізації хаотичних моделей і дослідження впливу початкових умов у фізичних системах, зокрема, в моделях Чуа, Лоренца і Ресслера. Хаотичні системи є динамічними і чутливими до початкових умов, що робить їх непередбачуваними щодо того, як вони будуть поводитися і реагувати. Це означає, що в довгостроковій перспективі навіть невеликі зміни початкових умов можуть призвести до дуже різних результатів. Хаотичні системи вивчаються в різних наукових галузях, включаючи фізику, математику, біологію, інженерію та економіку. Python, найпопулярніша у світі мова наукового програмування, перетворює складні моделі на інтуїтивно зрозумілі візуалізації. У статті розкриваються можливості різних алгоритмів і бібліотек Python, що використовуються для візуалізації цих моделей з урахуванням їхньої специфіки. Основну увагу приділено трьом хаотичним моделям: об'єкту Чуа, який є універсальним прикладом хаотичної системи; атрактору Лоренца, який відомий своїми хаотичними властивостями; та обертальному осцилятору Ресслера, який широко використовується в таких галузях, як біологія, хімія, фізика та інженерія. Кожна модель детально розглядається, наводяться її ключові характеристики та параметри, а графіки цих моделей демонструються за допомогою симуляції на мові Python. Python, завдяки простоті використання та високій продуктивності, дозволяє вирішувати такі завдання швидко та ефективно. Насамкінець автори діляться своїми висновками щодо важливості початкових умов для систем Лоренца, Ресслера та Чуа, а також їхнього впливу на телекомунікаційні системи. Це дослідження дає уявлення про те, як Python, мова програмування з високим рівнем абстракції, дозволяє швидко і ефективно розробляти складні алгоритми і моделі, необхідні для роботи з хаотичними системами. Вона також дозволяє дослідникам та інженерам розробляти ефективні алгоритми для обробки сигналів та управління телекомунікаційними системами.

КЛЮЧОВІ СЛОВА візуалізація хаотичних моделей, Python, моделі Чуа, Лоренца та Ресслера, динамічні системи, хаотичні системи.