Reappraising double pendulum dynamics across multiple computational platforms

Sandy Herho, Faiz Fajary, Katarina Herho, Iwan Anwar, Rusmawan Suwarman, Dasapta Erwin Irawan

Reappraising double pendulum dynamics across multiple computational platforms

Sandy H. S. Herho[1*], Faiz R. Fajary[2, 3], Katarina E. P. Herho[4], Iwan P. Anwar[5], Rusmawan Suwarman[3], Dasapta E. Irawan[6]

[1*]Department of Earth and Planetary Sciences, University of California, 900 University Ave., Riverside, 92521, CA, USA.
[2]Coastal Hazards and Energy System Science (CHESS) Lab, Hiroshima University, 1-5-1 Kagamiyama, Higashihiroshima, 739-8529, Hiroshima, Japan.
[3]Atmospheric Science Research Group, Bandung Institute of Technology (ITB), Jalan Ganesha 10, Bandung, 40132, West Java, Indonesia.
[4]Department of Geological Engineering, Trisakti University, Jalan Letjen S. Parman 1, West Jakarta, 1440, DKI Jakarta, Indonesia.
[5]Oceanography Research Group, Bandung Institute of Technology (ITB), Jalan Ganesha 10, Bandung, 40132, West Java, Indonesia.
[6]Applied Geology Research Group, Bandung Institute of Technology (ITB), Jalan Ganesha 10, Bandung, 40132, West Java, Indonesia.

*Corresponding author(s). E-mail(s): sandy.herho@email.ucr.edu;

Abstract

This study presents the complexity and sensitivity of chaotic system dynamics in the case of the double pendulum. It applied detailed numerical analyses of the double pendulum in multiple computing platforms in order to demonstrate the complexity in behavior of the system of double pendulums. The equations of motion were derived from the Euler-Lagrange formalism, in order to capture the system's dynamics, which is coupled nonlinearly. These were solved numerically using the efficient Runge-Kutta-Fehlberg method, implemented in Python, R, GNU Octave, and Julia, while runtimes and memory usage were extensively benchmarked across these environments. Time series analyses, including the calculation of Shannon entropy and the Kolmogorov-Smirnov test, quantified the system's unpredictability and sensitivity to infinitesimal perturbations of the initial conditions. Phase space diagrams illustrated the intricate trajectories and strange attractors, as further confirmation of the chaotic nature of the double

pendulum. All the findings have a clear indication of the importance of accurate measurements of the initial condition in a chaotic system, contributing to an increased understanding of nonlinear dynamics. Future research directions are faster simulations using Numba and GPU computing, stochastic effects, chaotic synchronization, and applications in climate modeling. This work will be useful for understanding chaos theory and efficient computational approaches in complex systems of dynamical nature.

Keywords: Chaotic Dynamics, Computational Platforms, Nonlinear Dynamics, Numerical Simulations, Sensitivity Analysis

1 Introduction

The double pendulum forms an interesting physics system consisting of two pendulum hanging in a rigidly connected line. It does not only rely on the observation of immediate effects but also the study of the underlying structures that make things change. Though at first glance this machine may look mundane, its behavior will prove to be exciting and the underlying idea of balance between order and disorder that has intrigued scientists and mathematicians for centuries will be inspirationally brought to life. It is an example of the fascinating ways simple systems can exhibit complex behavior. From people like Henri Poincaré, who first took a look at non-linear systems and laid the foundation for chaos theory, to today's powerful computational models, the double pendulum remains a symbol that the secret of nonlinear dynamics is a profound one [1].

A double pendulum consists of two massless rods, each with a concentrated point mass at its end, connected by a frictionless hinge. This seemingly simple setup actually displays a remarkable sensitivity to initial conditions, a characteristic of chaotic systems that was first discovered by British mathematician and physicist Mary Cartwright in the early 20th century [1, 2]. Even tiny changes in the starting position can cause significant divergences in the paths the system takes - a concept that challenges the traditional idea put forth by French scholar Pierre-Simon Laplace known as Laplace's demon, which suggests that knowing all initial conditions of a system guarantees the ability to predict its future evolution with total certainty [3].

The double pendulum is a prominently studied dynamical system in climate science due to its exquisite sensitivity to initial conditions, making it a valuable model for studying chaos theory and its applications in understanding Earth's complex climate systems [4]. Gaining insights into chaotic systems like the double pendulum could prove vital for tackling one of our most pressing global issues - anthropogenic climate change driven by human activities. Several studies have used simplified models like the double pendulum to gain insights into the non-linear dynamics underlying atmospheric-oceanic flows and long-range climate predictions [e. g. 5–9].

The double pendulum exemplifies how deterministic systems can exhibit unpredictable, chaotic behavior, bridging the gap between the simple, ordered world of classical physics and the apparent randomness we observe in complex phenomena like

weather and climate patterns [10]. This chaotic complexity represents a major frontier in our scientific understanding - we have robust theories for simple systems and stochastic models for randomness, but lack a unified framework to explain the rich dynamical behaviors that emerge in between the two extremes [11]. Unraveling the chaos inherent in multi-scale systems like the double pendulum may unlock deeper insights into the fundamental laws governing our climate and the universe at large.

In this paper, our main goal is to thoroughly investigate and analyze the complex movements of a double pendulum system using detailed numerical simulations. We utilized open-source computing platforms to develop engaging visuals that can be used as effective educational resources. This graphical representation will help students, especially those with limited abstract mathematics knowledge, better understand the concepts of calculus of variations. The calculus of variations is a branch of mathematical analysis concerned with optimizing functionals, which are mappings from a space of functions to the real numbers. It addresses problems where the goal is to find a function that extremizes a given functional, either minimizing or maximizing it [12]. This field is particularly important in areas such as physics, engineering, and economics, where one seeks to optimize quantities that depend on functions, such as energy, action, or cost [13]. By presenting this information visually and in a practical context, we hope to make it easier for students to connect theoretical concepts with real-world applications.

Furthermore, allowing students to freely access the simulation code enables them to explore the computational side of the project, enhancing their comprehension of how theory translates into practice. This also gives them the opportunity to improve and perfect the models, promoting a practical approach to learning physics and applied mathematics through coding. To ensure optimal performance and efficiency, we thoroughly evaluated the free and open-source computing environments utilized in this study. This study will offer helpful information for teachers, scholars, and professionals, helping them choose the best tools for their individual computing requirements and improving resource management and promoting excellence in scientific computing.

2 Methods

In order to predict the positional paths of two point masses in a double pendulum system, we need a classical mechanics framework represented by the Euler-Lagrange equation. This is because the Euler-Lagrange equation provides a systematic approach to deriving the equations of motion for complex systems like the double pendulum, taking into account the constraints and forces involved. On the other hand, using Newtonian mechanics alone for the double pendulum can be challenging due to the system's nonlinear nature and the presence of constraints such as the lengths of the pendulum arms. While it's possible to analyze simpler pendulum systems using Newton's laws directly, the double pendulum's motion involves coupled, nonlinear differential equations that are more conveniently handled using the Euler-Lagrange formalism [14].

The Euler-Lagrange equation is essential in classical mechanics and field theory, describing the motion of particles or fields by minimizing a functional known as the action. To derive this equation from scratch, we started with the action S, defined as

the integral of the Lagrangian ($\mathcal{L}$) over time:

$$S = \int_{t_1}^{t_2} \mathcal{L}(q, \dot{q}, t)\, dt \tag{1}$$

Here, q represents the generalized coordinates, $\dot{q}$ is the derivative of q with respect to time, and t is time. The Lagrangian ($\mathcal{L}$) has a simple and concise definition:

$$\mathcal{L} \equiv T - V \tag{2}$$

The kinetic energy (T) and potential energy (V) together make up the classical Lagrangian, which is just the difference between these energies in the system. This applies to classical mechanics with conservative systems, where the total energy is the sum of kinetic and potential energies. Our next step was to find the path $q(t)$ that keeps the action constant, even if the path changes a little. This basic idea is called the principle of least action.

To find this stationary path, we used the calculus of variations. Let $\delta q(t)$ be a small variation in the path $q(t)$, such that $q(t)$ becomes $q(t) + \delta q(t)$. The variation in the action is then:

$$\delta S = \int_{t_1}^{t_2} \left[\mathcal{L}(q + \delta q, \dot{q} + \delta \dot{q}, t) - \mathcal{L}(q, \dot{q}, t)\right]\, dt \tag{3}$$

We expanded $L(q + \delta q, \dot{q} + \delta \dot{q}, t)$ in a Taylor series around q and $\dot{q}$:

$$\mathcal{L}(q + \delta q, \dot{q} + \delta \dot{q}, t) = \mathcal{L}(q, \dot{q}, t) + \frac{\partial \mathcal{L}}{\partial q}\delta q + \frac{\partial \mathcal{L}}{\partial \dot{q}}\delta \dot{q} + \mathcal{O}(\delta q^2, \delta \dot{q}^2) \tag{4}$$

Substituting the expression back into equation 3 we would consider terms up to second order or higher in δq and $\delta \dot{q}$ in the variation of the action S, $\mathcal{O}(\delta q^2, \delta \dot{q}^2)$:

$$\delta S = \int_{t_1}^{t_2} \left[\frac{\partial \mathcal{L}}{\partial q}\delta q + \frac{\partial \mathcal{L}}{\partial \dot{q}}\delta \dot{q}\right]\, dt + \mathcal{O}(\delta q^2, \delta \dot{q}^2) \tag{5}$$

Integrating the first term by parts with respect to t and assuming that variations $\delta q(t_1)$ and $\delta q(t_2)$ vanish (boundary conditions), we got:

$$\delta S = \int_{t_1}^{t_2} \left[\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q}\right] \delta q\, dt \tag{6}$$

For the action to be stationary, δS must be zero for all possible variations δq. This leads to the Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = 0 \tag{7}$$

This equation governs the dynamics of the system and provides the equations of motion for the generalized coordinates $q(t)$.

To apply the Euler-Lagrange equation to the double pendulum system, we must first establish its Lagrangian, which encapsulates both the system's kinetic and potential energies. This process started by delineating the geometric relationships governing the vertical and horizontal positions within the double pendulum system, as illustrated in Fig. 1. These relationships were formalized through the following equations, denoted as equations 8 and 9:

$$\begin{aligned} x_1 &= L_1 \sin(\theta_1) \\ x_2 &= x_1 + L_2 \sin(\theta_2) \end{aligned} \tag{8}$$

$$\begin{aligned} y_1 &= L_1 \cos(\theta_1) \\ y_2 &= y_1 + L_2 \cos(\theta_2) \end{aligned} \tag{9}$$

Here, x_1 and x_2 represent the horizontal positions, while y_1 and y_2 represent the vertical positions. These positions are determined by the lengths of the pendulum arms (L_1 and L_2) and the angles (θ_1 and θ_2).

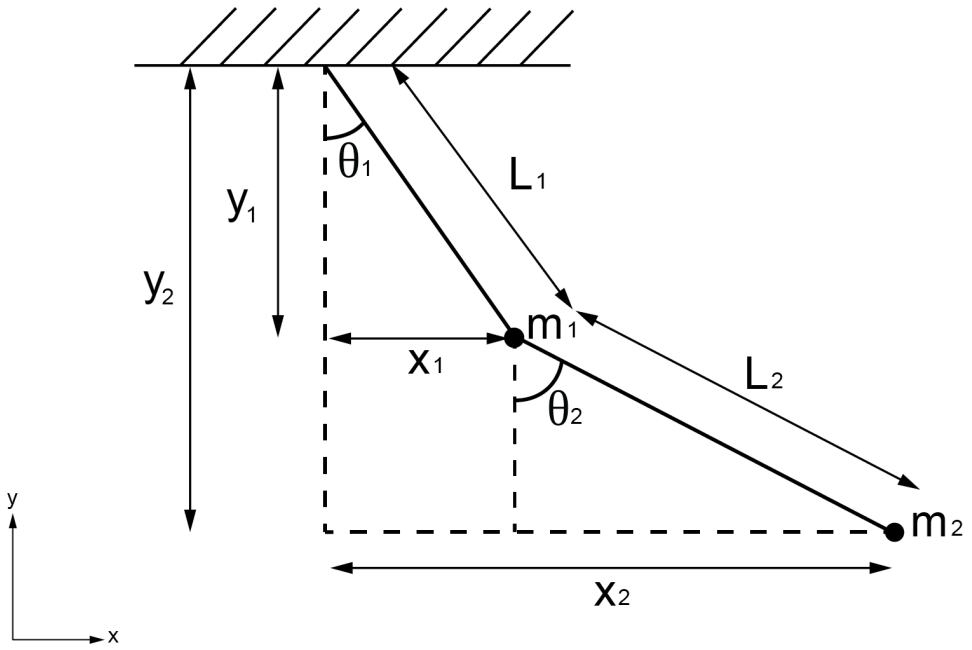


Fig. 1: Free-body diagram of a double pendulum. Point masses (m_1, m_2) connected by massless rods (L_1, L_2). Angles θ_1 and θ_2 represent deviation from the vertical axis.

Since the positions x and y are functions of the angles θ_1 and θ_2, respectively, obtaining their first derivatives requires applying the chain rule. This application results in the following expressions:

$$\begin{aligned} \dot{x}_1 &= L_1\dot{\theta}_1 \cos(\theta_1) \\ \dot{x}_2 &= L_1\dot{\theta}_1 \cos(\theta_1) + L_2\dot{\theta}_2 \cos(\theta_2) \end{aligned} \tag{10}$$

$$\begin{aligned} \dot{y}_1 &= -L_1\dot{\theta}_1 \sin(\theta_1) \\ \dot{y}_2 &= -L_1\dot{\theta}_1 \sin(\theta_1) - L_2\dot{\theta}_2 \cos(\theta_2) \end{aligned} \tag{11}$$

These derivatives $\dot{x}_1, \dot{x}_2, \dot{y}_1$, and $\dot{y}_2$ represent the rates of change of the horizontal and vertical positions with respect to time, taking into account the angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$.

Starting from the general formula for kinetic energy $T = \frac{1}{2}mv^2$ (where m is the total of two-point masses of $m_1 + m_2$ and v is velocity), we substituted the velocities $\dot{x}_1, \dot{x}_2, \dot{y}_1$, and $\dot{y}_2$ from the systems of equations 10 and 11 into equation 12. This substitution yields the expanded form:

$$\begin{aligned} T &= \frac{1}{2}\left(m_1(\dot{x}_1^2 + \dot{y}_1^2) + m_2(\dot{x}_2^2 + \dot{y}_2^2)\right) \\ &= \frac{1}{2}\left(m_1\left(L_1\dot{\theta}_1 \cos(\theta_1)\right)^2 + m_1\left(-L_1\dot{\theta}_1 \sin(\theta_1)\right)^2 \right. \\ &\quad \left. + m_2\left(L_1\dot{\theta}_1 \cos(\theta_1) + L_2\dot{\theta}_2 \cos(\theta_2)\right)^2 + m_2\left(-L_1\dot{\theta}_1 \sin(\theta_1) - L_2\dot{\theta}_2 \cos(\theta_2)\right)^2\right) \end{aligned} \tag{12}$$

Expanding and simplifying each term step by step, we obtained expressions for the kinetic energy components. These components involve terms related to the masses m_1 and m_2, the lengths L_1 and L_2 of the pendulum arms, and the angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$. After combining and simplifying the terms, we arrived at the final form of the kinetic energy:

$$T = \frac{1}{2}(m_1 + m_2)L_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2L_2^2\dot{\theta}_2^2 + m_2L_1L_2\dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{13}$$

This expression captures the kinetic energy T of the double pendulum system comprehensively, incorporating the masses m_1 and m_2, the lengths L_1 and L_2 of the pendulum arms, and the angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$ in a way that reflects the system's dynamics and interactions.

Beginning with equation 14, which defines potential energy as a function of the vertical positions y_1 and y_2, and the gravitational constant g, we can express it as:

$$V = m_1gy_1 + m_2gy_2 \tag{14}$$

Substituting the expressions for y_1 and y_2 from the system of equations 9 into the above equation yields:

$$V = m_1 g(L_1 \cos(\theta_1)) + m_2 g(L_1 \cos(\theta_1) + L_2 \cos(\theta_2)) \tag{15}$$

We then simplified this expression to:

$$V = g((m_1 + m_2)L_1 \cos(\theta_1) + m_2 L_2 \cos(\theta_2)) \tag{16}$$

Finally, to express potential energy V solely in terms of the angles θ_1 and θ_2, we derived:

$$V = -g((m_1 + m_2)L_1 \cos(\theta_1) + m_2 L_2 \cos(\theta_2)) \tag{17}$$

Here, g represents the gravitational acceleration, and the potential energy V is derived from the vertical positions of the pendulum components and their respective masses, articulated in terms of the angles θ_1 and θ_2.

After deriving the expressions for kinetic energy T and potential energy V, we can now formulate the Lagrangian $\mathcal{L}$ for the double pendulum system. The Lagrangian is defined in the definition 2. Substituting the expressions we derived for T and V into this equation, we obtained:

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}(m_1 + m_2)L_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 L_2^2\dot{\theta}_2^2 + m_2 L_1 L_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) \\
&\quad - (-g((m_1 + m_2)L_1 \cos(\theta_1) + m_2 L_2 \cos(\theta_2))) \\
&= \frac{1}{2}(m_1 + m_2)L_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 L_2^2\dot{\theta}_2^2 + m_2 L_1 L_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) \\
&\quad + g((m_1 + m_2)L_1 \cos(\theta_1) + m_2 L_2 \cos(\theta_2))
\end{aligned} \tag{18}$$

This expression for the Lagrangian $\mathcal{L}$ encapsulates the dynamic behavior of the double pendulum system. It incorporates the masses m_1 and m_2, the lengths L_1 and L_2 of the pendulum arms, the angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$, and the gravitational constant g, as well as the angles θ_1 and θ_2 that describe the positions of the pendulum components. The Lagrangian $\mathcal{L}$ serves as a fundamental quantity in the analysis of the system's motion and dynamics, providing a comprehensive representation of its energy and interactions.

Starting with the Lagrangian $\mathcal{L}$ derived earlier (equation 18), we applied the Euler-Lagrange equation (equation 7) to derive the equations of motion for a system with two point masses (the double pendulum in this case). The Euler-Lagrange equation for a variable q_i is given by:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = 0 \tag{19}$$

Applying this equation to the variables θ_1 and θ_2 separately, we obtained two set of equations:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta_1}}\right) - \frac{\partial \mathcal{L}}{\partial \theta_1} = 0 \\ \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta_2}}\right) - \frac{\partial \mathcal{L}}{\partial \theta_2} = 0 \end{aligned} \tag{20}$$

For θ_1, we first calculated the partial derivative of $\mathcal{L}$ with respect to $\dot{\theta}_1$:

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (m_1 + m_2)L_1^2\dot{\theta}_1 + m_2 L_1 L_2 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{21}$$

Then, we took the derivative of this with respect to time t:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) = (m_1 + m_2)L_1^2\ddot{\theta}_1 + m_2 L_1 L_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{22}$$

Next, we calculated the partial derivative of $\mathcal{L}$ with respect to θ_1:

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -m_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - g(m_1 + m_2)L_1 \sin(\theta_1) \tag{23}$$

Finally, substituting these derivatives into the Euler-Lagrange equation (equation 20) for θ_1:

$$(m_1 + m_2)L_1^2\ddot{\theta}_1 + m_2 L_1 L_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) = 0 \tag{24}$$

To derive the equation of motion for θ_2 using the Euler-Lagrange equation (equation 20), we started by calculating the partial derivative of the Lagrangian $\mathcal{L}$ with respect to the derivative of θ_2, denoted as $\dot{\theta}_2$. This yields:

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = m_2 L_2^2 \dot{\theta}_2 + m_2 L_1 L_2 \dot{\theta}_1 \cos(\theta_1 - \theta_2) \tag{25}$$

Taking the time derivative of this expression gave us:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right) = m_2 L_2^2 \ddot{\theta}_2 + m_2 L_1 L_2 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - m_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) \tag{26}$$

Next, we calculated the partial derivative of $\mathcal{L}$ with respect to θ_2, denoted as $\frac{\partial \mathcal{L}}{\partial \theta_2}$, which was given by:

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = -m_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_1 - \theta_2) - m_2 L_2 g \sin(\theta_2) \tag{27}$$

Substituting the expressions for $\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}$ and $\frac{\partial \mathcal{L}}{\partial \theta_2}$ into the Euler-Lagrange equation and simplifying this equation further results in the equation of motion for θ_2, given as:

$$m_2 L_2 \ddot{\theta}_2 + m_2 L_1 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - m_2 L_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + m_2 g \sin(\theta_2) = 0 \tag{28}$$

Combining the final forms of the Euler-Lagrange equations for θ_1 (equation 24) and θ_2 (equation 28) together form the complete system of equations of motion for the double pendulum system. They are coupled second-order ordinary differential equations (ODEs) because the acceleration terms $\ddot{\theta}_1$ and $\ddot{\theta}_2$ are dependent on each other due to the cosine and sine terms involving θ_1 and θ_2. This coupling reflects the interdependence of the pendulum's motions and positions, making the system dynamically rich and challenging to analyze without numerical or advanced analytical techniques.

For the numerical analysis, we substituted the following expressions ($\dot{\theta}_1 = \omega_1, \quad \dot{\theta}_2 = \omega_2, \quad \Delta\theta = \theta_1 - \theta_2$) to simplify the system of equations for computational purposes. Substituting these into the given equations (equations 24 and 28), we obtained the following system:

$$\begin{aligned}
&(m_1 + m_2)L_1\dot{\omega}_1 + m_2L_2\dot{\omega}_2\cos(\Delta\theta) + m_2L_2\omega_2^2\sin(\Delta\theta) + (m_1 + m_2)g\sin\theta_1 = 0 \\
&m_2L_2\dot{\omega}_2 + m_2L_1\dot{\omega}_1\cos(\Delta\theta) - m_2L_1\omega_1^2\sin(\Delta\theta) + m_2g\sin(\theta_2) = 0
\end{aligned} \tag{29}$$

This transformed system allows us to numerically solve for the angular frequencies ω_1 and ω_2 given initial conditions and system parameters.

To simplify the numerical analysis, we first rewrote the original system of equations in terms of the defined variables:

$$\begin{cases}
\alpha = (m_1 + m_2)L_1 \\
\beta = m_2L_2\cos(\Delta\theta) \\
\gamma = m_2L_1\cos(\Delta\theta) \\
\delta = m_2L_2 \\
\varepsilon = -m_2L_2\omega_2^2\sin(\Delta\theta) - m_2g\sin(\theta_2) \\
\zeta = m_2L_2\omega_1^2\sin(\Delta\theta) - m_2g\sin(\theta_2)
\end{cases} \tag{30}$$

We then rewrote the system of equations in matrix form:

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{pmatrix} = \begin{pmatrix} \varepsilon \\ \zeta \end{pmatrix} \tag{31}$$

Finally, solving for $\dot{\omega}_1$ and $\dot{\omega}_2$, we obtained:

$$\begin{aligned}
\dot{\omega}_1 &= \tfrac{\varepsilon\delta - \beta\zeta}{\alpha\delta - \beta\gamma} \\
\dot{\omega}_2 &= \tfrac{\alpha\zeta - \gamma\varepsilon}{\alpha\delta - \beta\gamma}
\end{aligned} \tag{32}$$

These derived equations were expressed in terms of the defined variables, providing a more organized and manageable representation for the numerical experiments.

For the numerical simulation needs in this study, we used parameters consistent with the physical properties of the system. These include an acceleration due to gravity (g) of $9.8\,\mathrm{m/s^2}$, the lengths of the pendulum arms L_1 and L_2 set to 2 meters and 1 meter respectively, and the masses of the pendulums (m_1 and m_2) set at 1 kilogram and 2 kilograms respectively. The simulation time (t) is chosen to be 10 seconds with 10,000 linearly time spacing, providing sufficient duration to observe the system's behavior.

With these parameters established, we set the initial conditions for the simulation. The initial angles (θ_1 and θ_2) are set to 3.14 radians (which is equivalent to 180°) for θ_1 and 1.57 radians (equivalent to 90°) for θ_2. Additionally, the initial angular velocities (ω_1 and ω_2) are initialized to 0 radians per second, representing a starting point where the pendulums are at rest. For the subsequent simulation, we changed the both angular velocities to 0.001 radians per seconds fo the sensitivity test with the initial conditions.

We used the Runge-Kutta-Fehlberg (RKF) method for approximating the solution of our problem (system of equations 32). RKF is a powerful numerical integration technique used to solve systems of ODEs with high accuracy and computational efficiency [13]. Its adaptability makes it particularly suitable for dynamic systems like the double pendulum, where the motion can be complex and highly nonlinear.

We started with the initial conditions and set the initial step size h. Then, we proceeded to define the predictor step by using the 4th order Runge-Kutta formulas to predict the solution at the next time step. For a general ODE of the form $\frac{dy}{dt} = f(t, y)$, the 4th order Runge-Kutta formulas are:

$$\begin{aligned} k_1 &= hf(t_n, y_n) \\ k_2 &= hf(t_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2}) \\ k_3 &= hf(t_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2}) \\ k_4 &= hf(t_n + h, y_n + k_3) \end{aligned} \tag{33}$$

The predicted solution at t_{n+1} is then given by:

$$y_{n+1}^{(4)} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{34}$$

We used the 5th order Runge-Kutta formulas to compute a more accurate estimate of the solution at the next time step. The 5th order Runge-Kutta formulas are similar

to the 4th order ones but include an additional evaluation point:

$$
\begin{aligned}
k_1 &= hf(t_n, y_n) \\
k_2 &= hf(t_n + \tfrac{h}{4}, y_n + \tfrac{k_1}{4}) \\
k_3 &= hf(t_n + \tfrac{3h}{8}, y_n + \tfrac{3k_1}{32} + \tfrac{9k_2}{32}) \\
k_4 &= hf(t_n + \tfrac{12h}{13}, y_n + \tfrac{1932k_1}{2197} - \tfrac{7200k_2}{2197} + \tfrac{7296k_3}{2197}) \\
k_5 &= hf(t_n + h, y_n + \tfrac{439k_1}{216} - 8k_2 + \tfrac{3680k_3}{513} - \tfrac{845k_4}{4104}) \\
k_6 &= hf(t_n + \tfrac{h}{2}, y_n - \tfrac{8k_1}{27} + 2k_2 - \tfrac{3544k_3}{2565} + \tfrac{1859k_4}{4104} - \tfrac{11k_5}{40})
\end{aligned}
\tag{35}
$$

The corrected solution at t_{n+1} is then given by:

$$
y_{n+1}^{(5)} = y_n + \frac{1}{90}(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6) \tag{36}
$$

Then, we calculated the local truncation error by comparing the 4th and 5th order solutions. The error estimate ϵ_n is given by:

$$
\epsilon_n = |y_{n+1}^{(5)} - y_{n+1}^{(4)}| \tag{37}
$$

We compared the error estimate ϵ_n to a predefined tolerance. If ϵ_n is within the tolerance, accept the step and update the solution. If ϵ_n exceeds the tolerance, reduce the step size h and we repeated the process until the error is acceptable. Finally, we continued integrating until reaching the desired end time or number of steps.

The adaptive step-size control implemented in the RKF method ensures accurate integration while minimizing computational costs. By dynamically adjusting the step size based on error estimation, the RKF method provides accurate numerical approximations of the system's behavior over time, rendering it an effective tool for analyzing complex dynamic systems such as the double pendulum. However, in the present study, we did not employ the RKF method directly from the beginning. Instead, we utilized several platform-specific tools, including SciPy in Python [15], deSolve in R [16], DifferentialEquations.jl in Julia [17], and the built-in function ode45 in GNU Octave [18].

The motivation behind using multiple programming languages and their respective computing environments was twofold. First, it allowed us to leverage the strengths and unique features of each language and environment, enabling a comprehensive exploration of the double pendulum problem from diverse perspectives. Second, it facilitated a comparative analysis of the performance and accuracy of different numerical solvers across these platforms.

In Python, the SciPy library provided a robust and well-established collection of scientific computing tools, including numerical solvers for ODEs. The flexibility and ease of use of Python, combined with the power of SciPy, made it a suitable choice for

implementing and testing numerical methods for the double pendulum problem [19, 20]. However, the interpreted nature of Python may introduce performance overhead compared to compiled languages.

The R programming language, with its deSolve package, offered a specialized environment for solving ODEs and differential algebraic equations (DAEs) [21]. R's strong emphasis on statistical analysis and data visualization made it an attractive option for exploring the double pendulum problem, enabling efficient data analysis and visual representation of the results. Nonetheless, the high-level nature of R may lead to performance limitations for computationally intensive simulations.

Julia, a relatively new language designed for scientific computing, provided a high-performance computing environment through its DifferentialEquations.jl package. The combination of Julia's dynamic programming capabilities and its efficient just-in-time (JIT) compilation made it a promising choice for solving the double pendulum problem with potentially improved computational performance [22, 23]. However, the relatively young ecosystem of Julia may present challenges in terms of package maturity and community support.

Finally, GNU Octave, a high-level language primarily intended for numerical computations, offered the built-in ode45 function, which implements a versatile Runge-Kutta method for solving ODEs. GNU Octave's compatibility with MATLAB® syntax and its open-source nature made it an accessible option for researchers and students alike [18]. However, its performance may be limited compared to lower-level languages or specialized numerical libraries. By employing these various computing environments, we aimed to evaluate the trade-offs between performance, accuracy, and ease of use for each approach.

When conducting scientific research involving computational simulations, it is crucial to ensure reproducibility and transparency in the methods used. In this particular study, we aim to benchmark the performance of four different computing environments: Python, R, Octave/MATLAB, and Julia, by running a script that simulates the motion of a double pendulum. The script was executed 1,000 times in each computing environment, and the runtime and memory usage for each run will be measured and recorded. The benchmarking was performed on a Fedora Linux 39 (Budgie) x86_64 system with a 20LB0021US ThinkPad P52s laptop equipped with an Intel i7-8550U (8) @4.000GHz CPU.

The rationale behind this benchmarking approach is multifaceted. Firstly, it promotes reproducibility by providing the scripts and the benchmarking script, allowing other researchers to easily replicate the computational experiments and verify the results. Additionally, it facilitates a direct comparison of the runtime and memory usage across different computing environments for the same task, providing valuable insights into the relative performance of each environment. This information can guide researchers in choosing the most suitable tool for their specific computational needs.

Furthermore, running the scripts 1,000 times in each environment helps to account for potential variability and ensures that the performance measurements are consistent and reliable. This is particularly important when dealing with computationally intensive simulations like the double pendulum simulation, where minor fluctuations in hardware or software configurations can impact the results. With a large number

of runs, it becomes possible to perform statistical analysis on the collected data, such as calculating confidence intervals, identifying outliers, and determining the statistical significance of any observed performance differences.

Measuring memory usage alongside runtime can provide insights into the scalability and resource requirements of each computing environment. This information is crucial when working with large-scale simulations or data-intensive applications, where efficient memory management is essential. By including the benchmarking methodology and results in the scientific paper, researchers demonstrate transparency and allow others to critically evaluate the computational approaches used in the study [24]. This aligns with the principles of open science and facilitates future replication and extension of the research.

After conducting the benchmarking process and collecting the runtime and memory usage data for each computing environment, we performed statistical analyses to determine if there were significant differences in performance among the platforms. Specifically, we employed the Kruskal-Wallis (K-W) test and Dunn's test with Bonferroni adjustment, which are commonly used in various scientific fields for comparing multiple groups or treatments.

The K-W test is a non-parametric alternative to the one-way analysis of variance (ANOVA) and is used when the assumptions of normality and homogeneity of variances are violated [25]. It is a rank-based test that evaluates whether the populations from which the samples were drawn have the same distribution. The test statistic for the K-W test was calculated as:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1) \tag{38}$$

, where N is the total number of observations across all groups, k is the number of groups, R_i is the sum of ranks for group i, and n_i is the number of observations in group i. The null hypothesis for the K-W test is that the populations have the same distribution, while the alternative hypothesis is that at least one population has a different distribution from the others.

If the K-W test indicates significant differences among the groups, post-hoc tests are typically performed to determine which specific groups differ from each other. In our case, we used Dunn's test with Bonferroni adjustment, which is a multiple comparison procedure that adjusts the significance level to control the family-wise error rate (FWER) [26]. The Dunn's test statistic for comparing groups i and j was calculated as:

$$Z = \frac{R_i - R_j}{\sqrt{\frac{N(N+1)}{12}\left(\frac{1}{n_i} + \frac{1}{n_j}\right)}} \tag{39}$$

, where R_i and R_j are the average ranks for groups i and j, respectively, N is the total number of observations across all groups, and n_i and n_j are the number of observations in groups i and j, respectively. The Bonferroni adjustment was applied by dividing the desired significance level (α) by the number of pairwise comparisons made, resulting in an adjusted significance level of $\alpha/\binom{k}{2}$, where k is the number of groups.

The K-W test and Dunn's test with Bonferroni adjustment are widely used in various scientific fields [e. g. 27–29]. These tests are particularly useful when dealing with non-normal data or when the assumptions of parametric tests (such as ANOVA) are violated. They provide a robust and reliable way to compare multiple groups or treatments, ensuring that any observed differences are statistically significant and not due to chance alone. By applying these statistical tests to our benchmarking data, we aimed to determine if there were significant differences in performance among the four computing environments (Python, R, GNU Octave, and Julia) for the double pendulum simulation task. These non-parametric procedures were implemented in Python using the SciPy stats module [15] and the scikit-posthoc library [30].

After conducting the benchmarking process to measure the runtime and memory usage of the double pendulum simulation across different computing environments, we further analyzed the obtained time series data to gain insights into the underlying dynamics of the system. One of the analysis techniques we employed was the calculation of Shannon entropy [31], which is a measure derived from information theory that quantifies the amount of information or uncertainty present in a random variable or time series. The Shannon entropy was calculated using the following equation:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{40}$$

, where $H(X)$ is the Shannon entropy, n is the number of unique values in the time series, and $p(x_i)$ is the probability of the occurrence of the value x_i in the time series. A higher Shannon entropy value indicates a higher degree of uncertainty or unpredictability in the time series, while a lower entropy value suggests a more predictable or regular pattern. This measure can provide valuable insights into the complexity and dynamics of the double pendulum system.

In our analysis, we calculated the Shannon entropy for each variable (e.g., x_1, y_1, θ_1, θ_2, ω_1, ω_2) in the time series data obtained from the double pendulum simulation. To interpret the entropy scores, we established thresholds based on the following criteria: low entropy (predictable) for entropy scores ≤ 0.5, medium entropy (some predictability) for entropy scores between 0.5 and 1.0, and high entropy (unpredictable) for entropy scores > 1.0. These thresholds are commonly used in various applications and provide a convenient way to interpret the degree of predictability or uncertainty present in the time series data. By calculating and analyzing the Shannon entropy of the time series data, we can gain insights into the predictability and complexity of the double pendulum system's dynamics. A high entropy score for a particular variable suggests that the corresponding time series is highly unpredictable or complex, while a low entropy score indicates a more regular or predictable pattern.

The choice of Shannon entropy as an analysis technique was motivated by its strong theoretical foundation in information theory and its widespread use in various scientific fields for quantifying the complexity and uncertainty of dynamical systems. Furthermore, the interpretation of entropy scores based on predefined thresholds provides a convenient and standardized way to categorize the time series data into different levels of predictability or complexity.

After conducting the entropy measurement, we employed the Kolmogorov-Smirnov (K-S) test to assess the sensitivity of the system to initial conditions, which is a characteristic feature of chaotic systems. The K-S test is a non-parametric statistical test that compares the cumulative distribution functions (CDFs) of two samples to determine if they are drawn from the same underlying distribution [32]. In the context of dynamical systems analysis, the K-S test can be used to compare the time series obtained from the original system with slightly perturbed initial conditions, allowing us to quantify the divergence between the trajectories. Let $F(x)$ and $G(x)$ be the empirical CDFs of the original time series and the perturbed time series, respectively. The K-S statistic is defined as the maximum absolute difference between these two CDFs:

$$D_{n,m} = \sup_x |F(x) - G(x)| \tag{41}$$

, where $\sup_x$ represents the supremum (least upper bound) of the set of absolute differences between the CDFs over all possible values of x.

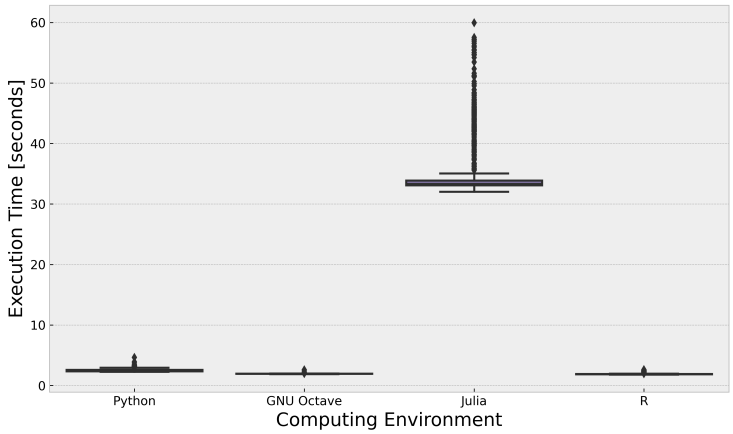
The null hypothesis for the K-S test is that the two samples are drawn from the same continuous distribution, while the alternative hypothesis is that they are drawn from different distributions. The null hypothesis is rejected if the K-S statistic $D_{n,m}$ exceeds a critical value that depends on the chosen significance level and the sample sizes n and m. In our analysis, we compared the original time series $(x_1, y_1, \theta_1, \theta_2, \omega_1, \omega_2)$ obtained from the double pendulum simulation with slightly perturbed time series, where the initial conditions for ω_1 and ω_2 were perturbed by 0.001 rad/s. By applying the K-S test to each pair of original and perturbed time series, we can assess whether the small perturbation in the initial conditions leads to a significant divergence in the trajectories over time. If the K-S test rejects the null hypothesis, indicating that the original and perturbed time series are drawn from different distributions, it suggests that the double pendulum system is sensitive to initial conditions, which is a hallmark of chaotic behavior. Conversely, if the test fails to reject the null hypothesis, it implies that the system is less sensitive to small perturbations in the initial conditions, suggesting a more predictable or regular dynamics.

The choice of the K-S test was motivated by its nonparametric nature, which means that it does not make assumptions about the underlying distribution of the data, making it suitable for analyzing complex dynamical systems where the distribution is often unknown or difficult to model parametrically. Additionally, the K-S test is widely used in various scientific fields for comparing distributions and detecting deviations from a hypothesized distribution [e. g. 33–35], further justifying its application in our analysis. We employed SciPy's stats module [15] to conduct an automated K-S test.

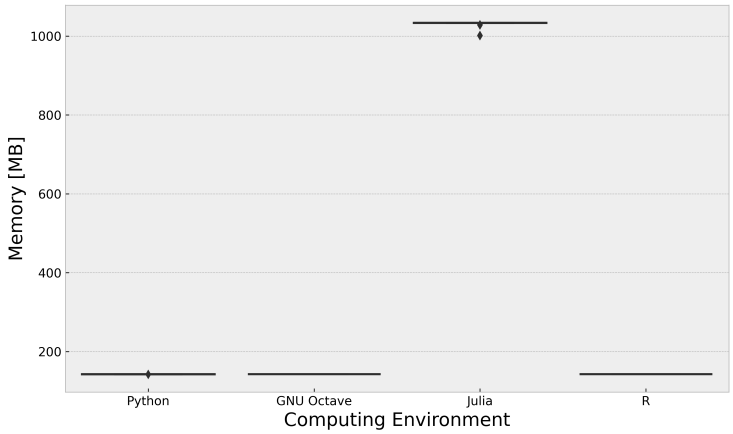
3 Results and Discussion

The K-W test, a non-parametric test for comparing multiple groups, yielded a test statistic of 3523.203 and a p-value of 0.000 for the runtime data, indicating that at least one group's median runtime significantly differs from the others. Dunn's post-hoc test, with Bonferroni adjustment for multiple comparisons, showed that all pairs of groups had p-values below 0.05, indicating significant differences in their median runtimes.

The results (Fig. 2a) indicate that R and GNU Octave exhibited the fastest runtimes, with mean values of 1.914 seconds and 1.944 seconds, respectively. The fast performance of R can be attributed to the use of the deSolve package, which provides efficient solvers for ODEs. GNU Octave, on the other hand, likely leverages optimized numerical libraries or solvers for ODE systems. Python followed closely with a mean runtime of 2.503 seconds, benefiting from the SciPy and NumPy libraries, which provide efficient numerical computations and ODE solvers. Notably, Julia had the slowest runtime performance, with a mean of 35.701 seconds, which is significantly longer than the other environments. One potential factor contributing to Julia's slower performance could be the overhead associated with its JIT compilation process. JIT compilation can introduce additional runtime overhead, particularly for computationally intensive tasks like solving ODEs, which may have impacted Julia's overall runtime performance in this specific implementation.



(a)



(b)

Fig. 2: Comparison of (a) execution time and (b) memory usage across different computing environments.

The K-W test for memory usage data yielded a test statistic of 3375.563 and a p-value of 0.000, indicating that at least one group's median memory usage significantly differs from the others. Dunn's post-hoc test showed that the pairs of groups with p-values below 0.05, indicating significant differences in their median memory usage, were GNU Octave-Julia, GNU Octave-Python, Julia-Python, and Julia-R. The results (Fig. 2b) show that Python, R, and GNU Octave exhibited similar memory usage patterns, with a mean of approximately 142.84 MB. Julia, on the other hand, had significantly higher memory usage, with a mean of 1031.703 MB, which is about seven times higher than the other environments. The differences in memory usage across the computing environments can be attributed to various factors, such as the memory management

strategies of the respective programming languages and the specific implementation of the double pendulum simulation in each environment. However, it is worth noting that the pure Julia implementation using DifferentialEquations.jl may have different memory requirements or optimization strategies compared to the packages or libraries used in the other environments.

When selecting the appropriate computing environment for the double pendulum simulation, the statistical significance of the runtime and memory usage differences should be considered alongside other factors, such as existing codebase, familiarity with the language, and potential optimizations. If runtime performance is the primary concern, R and GNU Octave would be the recommended choices based on the provided results, with R leveraging the deSolve package and GNU Octave likely utilizing optimized numerical libraries or solvers for ODE systems. Python, with the SciPy and NumPy libraries, also exhibited a relatively good runtime performance and could be a viable option, especially if existing Python code or familiarity with the language is a consideration. If memory usage is a critical factor and the higher memory footprint of Julia is not a concern, Julia could be considered, potentially with further optimization efforts or alternative implementations. It is important to note that the pure Julia implementation using DifferentialEquations.jl may have different memory requirements or optimization strategies compared to the packages or libraries used in the other environments.

Figure 3 and Fig. 4 present time series plots of a double pendulum system with slightly different initial angular velocities for the inner and outer pendulum. Each subplot displays the time evolution of the angular positions of both pendulums over 10 seconds with 1000 time steps.

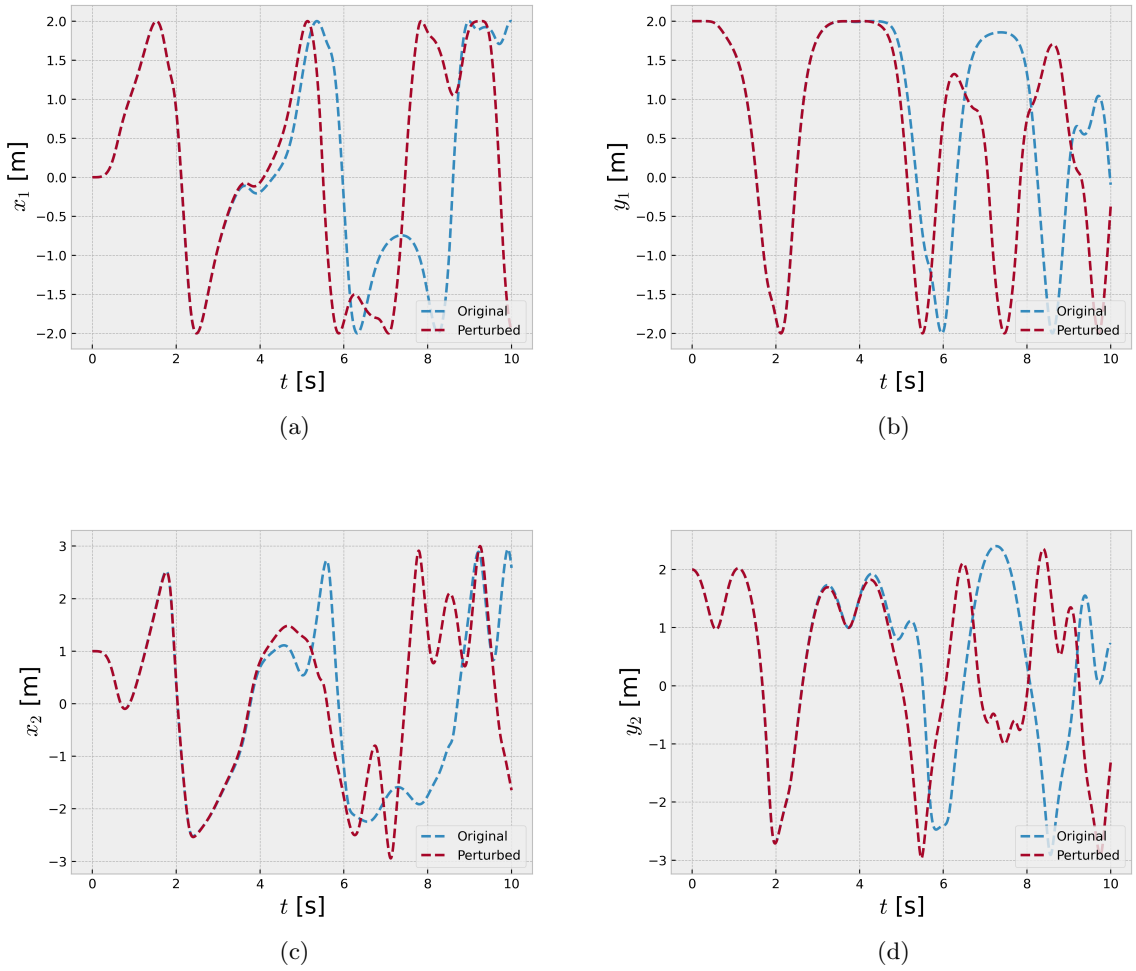


Fig. 3: Time series of a double pendulum positions with slightly different initial angular velocities for (a-b) inner and (c - d) outer pendulum.

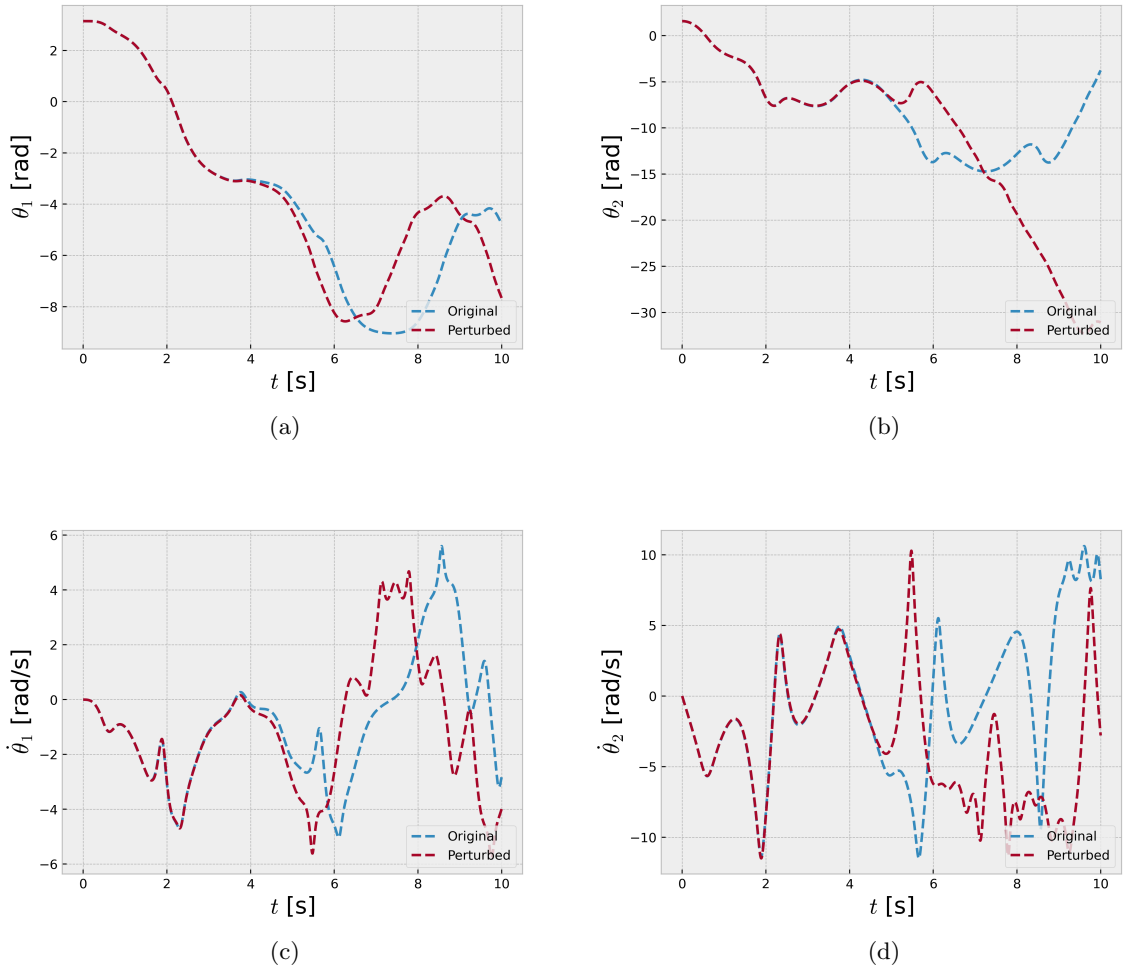


Fig. 4: Time series of a double pendulum (a - b) angles and (c - d) angular velocities with slightly different initial angular velocities for (a - c) inner and (b - d) outer pendulum.

The sensitivity to initial conditions is evident as the trajectories diverge significantly, despite only a small difference of 0.001 rad/s in the initial angular velocities (ω_1 and ω_2) of the inner and outer pendulums. This phenomenon is characteristic of chaotic systems, where minuscule changes in initial conditions can lead to vastly different long-term behaviors, making precise predictions challenging [1]. The time series exhibit intricate patterns with recurring oscillations, indicative of the underlying nonlinear dynamics. However, the trajectories quickly diverge, showcasing the system's

sensitivity to initial conditions, a hallmark of chaos theory [36]. The time series initially following similar paths but rapidly diverging due to the infinitesimal differences in the initial angular velocities, further demonstrating the sensitive dependence on initial conditions in chaotic systems.

These results highlight the importance of accurately measuring and accounting for initial conditions in chaotic systems, as even minute uncertainties can amplify over time, leading to substantial deviations in the system's behavior. The double pendulum serves as a compelling example of the intricate dynamics and unpredictability that can arise in nonlinear systems due to their extreme sensitivity to initial conditions, a fundamental concept in chaos theory [37].

To further quantify and statistically validate this divergence, we performed the K-S test on the time series data for various variables, including positions (x, y, θ) and angular velocities (ω_1, ω_2), before and after the initial perturbation. The K-S test results revealed that for each of these variables, the distributions of the time series data before and after the perturbation were significantly different, with p-values less than 0.05, leading to the rejection of the null hypothesis. This statistically confirms that the slight change in the initial angular velocity has a profound impact on the system's dynamics, causing the distributions of the position and velocity variables to diverge substantially.

This divergence can be attributed to the chaotic nature of the double pendulum system, where the nonlinear equations governing its motion exhibit extreme sensitivity to initial conditions. Even minuscule differences in the starting values can rapidly amplify over time, leading to vastly different trajectories in the phase space [36]. The K-S test results corroborate this fundamental characteristic of chaos, demonstrating that the distributions of the system's variables become statistically distinct due to the exponential divergence of initially close trajectories, a phenomenon known as the "butterfly effect" [38].

Figure 5a and Fig. 5b show the trajectories of the inner and outer pendulums in the $x - y$ plane, providing a visualization of their motion over time. However, Fig. 5c and Fig. 5d represent the phase space diagrams of the coupled pendulum system. A phase space diagram is a powerful tool for studying dynamical systems, as it provides a geometric representation of the system's state at any given time [13]. In the case of the coupled pendulum system, the phase space diagrams plot the angular position (θ) of each pendulum against its angular velocity (ω), capturing the evolution of the system's state over time.

The complex patterns observed in Fig. 5c and Fig. 5d are indicative of the chaotic nature of the coupled pendulum system. The phase space trajectories exhibit intricate, aperiodic behavior, never repeating the same pattern or revisiting the same state. The presence of chaos in the coupled pendulum system has significant consequences for its predictability and controllability [36]. Even with precise knowledge of the initial conditions and governing equations, the system's behavior becomes increasingly difficult to predict over long time scales due to the exponential divergence of nearby trajectories in the phase space.

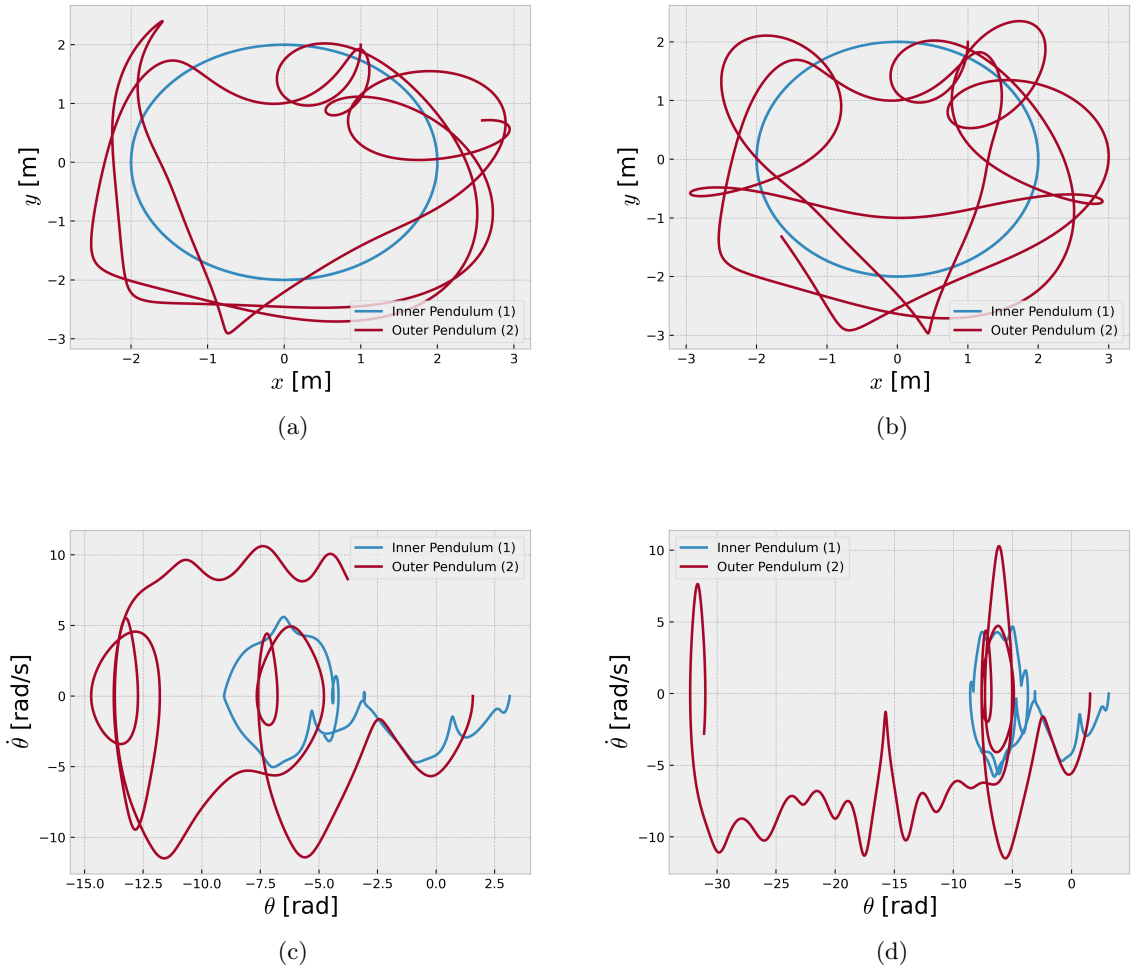


Fig. 5: Coupled inner-outer pendulum dynamics. (a - c) Original system. (b - d) System with 0.001 rad/s perturbation to initial angular velocities. (a - b) $x-y$ trajectories. (c - d) θ vs. ω phase space diagrams for inner (blue) and outer (red) pendulums.

Furthermore, chaotic systems often exhibit strange attractors in the phase space, which are geometrically complex structures that the system's trajectories are confined. The presence of strange attractors in the phase space diagrams of the coupled pendulum system suggests that the system's dynamics are governed by an underlying deterministic process, despite the apparent randomness and unpredictability of its behavior [1].

The high Shannon entropy values (greater than 1.0) mentioned for the time series data of the coupled pendulum system are consistent with the observed chaotic dynamics. For chaotic systems, the Shannon entropy is expected to be high due to the inherent unpredictability and lack of regularity in the system's behavior [39].

4 Conclusion

The double pendulum system serves as a compelling example of the intricate dynamics and unpredictability that can arise in nonlinear systems due to their extreme sensitivity to initial conditions, a fundamental concept in chaos theory. Through numerical simulations and extensive analysis, we have demonstrated the chaotic behavior of the double pendulum, characterized by the exponential divergence of trajectories, the presence of strange attractors in the phase space, and high Shannon entropy values. These findings underscore the importance of accurately measuring and accounting for initial conditions in chaotic systems, as even minute uncertainties can amplify over time, leading to substantial deviations in the system's behavior. Furthermore, our comparative study of different computing environments (Python, R, GNU Octave, and Julia) has revealed significant differences in their runtime performance and memory usage for the double pendulum simulation task. This information can guide researchers in selecting the most appropriate computing environment based on their specific needs and resource constraints.

To further enhance the computational performance of the double pendulum simulations, future work could explore the use of accelerated computing techniques. Numba, a JIT compiler for Python [40], can be leveraged to optimize numerical computations by compiling Python code to efficient machine instructions, potentially improving the runtime performance of the Python-based simulations [e. g. 41–43]. Additionally, the utilization of GPU-accelerated computing libraries like CuPy (CUDA for Python) [44] could significantly accelerate the simulations by offloading computationally intensive tasks to the highly parallel architecture of modern graphics processing units (GPUs) [e. g. 45–47]. The massive parallelism provided by GPUs can lead to substantial speedups, especially for large-scale simulations or ensemble runs. While the double pendulum system studied in this work was modeled as a deterministic system, future studies could explore the effects of incorporating stochastic elements. Real-world systems often exhibit random fluctuations or noise, which can significantly impact the system's dynamics and introduce additional complexity. By incorporating stochastic components into the double pendulum model, researchers could investigate the interplay between deterministic chaos and random noise, potentially revealing new insights into the behavior of complex dynamical systems.

Future investigations could also focus on studying chaotic synchronization, a phenomenon where two or more chaotic systems can become synchronized, exhibiting correlated behavior despite their inherent unpredictability. Exploring chaotic synchronization in coupled double pendulum systems or the potential applications of such synchronization in various fields, such as secure communication, signal processing, and control systems, could yield valuable insights. As mentioned in the introduction, the double pendulum system serves as a valuable model for studying chaos theory

and its applications in understanding Earth's complex climate systems. Future work could explore the potential use of the double pendulum as a simplified model for investigating the nonlinear dynamics underlying atmospheric-oceanic flows and developing improved long-range climate predictions. By pursuing these future directions, researchers can continue to deepen our understanding of chaotic systems, leverage advanced computational techniques for efficient simulations, and explore the potential applications of chaos theory in various scientific and engineering domains.

Acknowledgments. We extend our sincere gratitude to Ferio Brahmana (KAIST) and Michael N. Evans (UMD) for their insightful discussions on nonlinear dynamics several years ago. These earlier interactions undoubtedly played a significant role in the development of this study. Furthermore, the authors also gratefully acknowledge the financial support of the Dean's Distinguished Fellowship at the University of California, Riverside (UCR) awarded in 2023. We also recognize the support of the ITB Research, Community Services, and Innovation Program (PPMI-ITB) in 2024. The code used for the numerical simulations in this study is available on our GitHub repository: https://github.com/sandyherho/doublePendulum.

References

[1] Letellier, C., Abraham, R., Shepelyansky, D.L., Rössler, O.E., Holmes, P., Lozi, R., Glass, L., Pikovsky, A., Olsen, L.F., Tsuda, I., Grebogi, C., Parlitz, U., Gilmore, R., Pecora, L.M., Carroll, T.L.: Some elements for a history of the dynamical systems theory. Chaos: An Interdisciplinary Journal of Nonlinear Science **31**(5) (2021) https://doi.org/10.1063/5.0047851

[2] Atiyah, L.: Dame Mary Cartwright 1900–1997. The Mathematical Gazette **82**(495), 494–496 (1998) https://doi.org/10.1017/S002555720016262X

[3] Crease, R.P.: Letting demons do the teaching. Physics World **35**(4), 31 (2022) https://doi.org/10.1088/2058-7058/35/04/20

[4] Alexandrov, D.V., Bashkirtseva, I.A., Crucifix, M., Ryashko, L.B.: Nonlinear climate dynamics: From deterministic behaviour to stochastic excitability and chaos. Physics Reports **902**, 1–60 (2021) https://doi.org/10.1016/j.physrep.2020.11.002

[5] Khatiwala, S., Shaw, B.E., Cane, M.A.: Enhanced sensitivity of persistent events to weak forcing in dynamical and stochastic systems: Implications for climate change. Geophysical Research Letters **28**(13), 2633–2636 (2001) https://doi.org/10.1029/2000GL012773

[6] Budyansky, M.V., Uleysky, M.Y., Prants, S.V.: Lagrangian coherent structures, transport and chaotic mixing in simple kinematic ocean models. Communications in Nonlinear Science and Numerical Simulation **12**(1), 31–44 (2007) https://doi.org/10.1016/j.cnsns.2006.01.008

[7] Cropp, R., Moroz, I.M., Norbury, J.: Chaotic dynamics in a simple dynamical green ocean plankton model. Journal of Marine Systems **139**, 483–495 (2014) https://doi.org/10.1016/j.jmarsys.2014.08.002

[8] Tantet, A., Lucarini, V., Lunkeit, F., Dijkstra, H.A.: Crisis of the chaotic attractor of a climate model: a transfer operator approach. Nonlinearity **31**(5), 2221 (2018) https://doi.org/10.1088/1361-6544/aaaf42

[9] Shen, B.-W., Pielke Sr, R.A., Zeng, X., Baik, J.-J., Faghih-Naini, S., Cui, J., Atlas, R.: Is weather chaotic?: Coexistence of chaos and order within a generalized lorenz model. Bulletin of the American Meteorological Society **102**(1), 148–158 (2021) https://doi.org/10.1175/BAMS-D-19-0165.1

[10] Ghil, M., Lucarini, V.: The physics of climate variability and climate change. Reviews of Modern Physics **92**(3), 035002 (2020) https://doi.org/10.1103/RevModPhys.92.035002

[11] Crutchfield, J.P.: Between order and chaos. Nature Physics **8**(1), 17–24 (2012) https://doi.org/10.1038/nphys2190

[12] Rindler, F.: Calculus of Variations vol. 5. Springer, Berlin (2018)

[13] Higham, N.J., Dennis, M.R., Glendinning, P., Martin, P.A., Santosa, F., Tanner, J.: Princeton Companion to Applied Mathematics. Princeton University Press, Princeton (2015)

[14] Calvão, A.M., Penna, T.P.: The double pendulum: a numerical study. European Journal of Physics **36**(4), 045018 (2015) https://doi.org/10.1088/0143-0807/36/4/045018

[15] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020) https://doi.org/10.1038/s41592-019-0686-2

[16] Soetaert, K., Petzoldt, T., Setzer, R.W.: Solving differential equations in R: package deSolve. Journal of Statistical Software **33**, 1–25 (2010) https://doi.org/10.18637/jss.v033.i09

[17] Rackauckas, C., Nie, Q.: Differentialequations.jl–A Performant and Feature-rich Ecosystem for Solving Differential Equations in Julia. Journal of Open Research Software **5**(1), 15–15 (2017) https://doi.org/10.5334/jors.151

[18] Eaton, J.W.: GNU Octave and reproducible research. Journal of Process Control **22**(8), 1433–1438 (2012) https://doi.org/10.1016/j.jprocont.2012.04.006

[19] Herho, S.H.S.: Tutorial Pemrograman Python 2 Untuk Pemula. WCPL ITB, Bandung (2017)

[20] Herho, S.H.S., Syahputra, M.R., Trilaksono, N.J.: Pengantar Metode Numerik Terapan: Menggunakan Python. WCPL ITB, Bandung (2024)

[21] Kong, L.: Epidemic modeling using differential equations with implementation in R. International Journal of Mathematical Education in Science and Technology **55**(2), 480–491 (2024) https://doi.org/10.1080/0020739X.2023.2249902

[22] Bezanson, J., Chen, J., Chung, B., Karpinski, S., Shah, V.B., Vitek, J., Zoubritzky, L.: Julia: Dynamism and performance reconciled by design. Proceedings of the ACM on Programming Languages **2**(OOPSLA), 1–23 (2018) https://doi.org/10.1145/3276490

[23] Gao, K., Mei, G., Piccialli, F., Cuomo, S., Tu, J., Huo, Z.: Julia language in machine learning: Algorithms, applications, and open issues. Computer Science Review **37**, 100254 (2020) https://doi.org/10.1016/j.cosrev.2020.100254

[24] Wang, D., Ueda, Y., Kula, R.G., Ishio, T., Matsumoto, K.: Can we benchmark code review studies? a systematic mapping study of methodology, dataset, and metric. Journal of Systems and Software **180**, 111009 (2021) https://doi.org/10.1016/j.jss.2021.111009

[25] Kruskal, W.H., Wallis, W.A.: Use of Ranks in One-Criterion Variance Analysis. Journal of the American Statistical Association **47**(260), 583–621 (1952) https://doi.org/10.1080/01621459.1952.10483441

[26] Dunn, O.J.: Multiple comparisons using rank sums. Technometrics **6**(3), 241–252 (1964) https://doi.org/10.1080/00401706.1964.10490181

[27] Das, B., Murgaonkar, D., Navyashree, S., Kumar, P.: Novel combination artificial neural network models could not outperform individual models for weather-based cashew yield prediction. International Journal of Biometeorology **66**(8), 1627–1638 (2022) https://doi.org/10.1007/s00484-022-02306-1

[28] Borko, Š., Premate, E., Zagmajster, M., Fišer, C.: Determinants of range sizes pinpoint vulnerability of groundwater species to climate change: A case study on subterranean amphipods from the Dinarides. Aquatic Conservation: Marine and Freshwater Ecosystems **33**(6), 629–636 (2023) https://doi.org/10.1002/aqc.3941

[29] Herho, S.H.S., Anwar, I.P., Herho, K.E.P., Dharma, C.S., Irawan, D.E.: Numerical simulation of the 2D trajectory of a non-buoyant fluid parcel under the influence of inertial oscillation. EarthArXiv (2024) https://doi.org/10.31223/X5GT35

[30] Terpilowski, M.A.: scikit-posthocs: Pairwise multiple comparison tests in Python. J. Open Source Softw. **4**(36), 1169 (2019) https://doi.org/10.21105/joss.01169

[31] Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal **27**(3), 379–423 (1948) https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[32] Massey Jr, F.J.: The Kolmogorov-Smirnov test for goodness of fit. Journal of the American Statistical Association **46**(253), 68–78 (1951) https://doi.org/10.1080/01621459.1951.10500769

[33] Ghatak, G., Mohanty, H., Rahman, A.U.: Kolmogorov–Smirnov test-based actively-adaptive Thompson sampling for non-stationary bandits. IEEE Transactions on Artificial Intelligence **3**(1), 11–19 (2021) https://doi.org/10.1109/TAI.2021.3121653

[34] Kini, K.R., Harrou, F., Madakyaru, M., Sun, Y.: Enhanced Data-Driven Monitoring of Wastewater Treatment Plants using the Kolmogorov-Smirnov Test. Environmental Science: Water Research & Technology (2024) https://doi.org/10.1039/D3EW00829K

[35] Sharma, V., Biswas, R.: Statistical analysis of seismic b-value using non-parametric Kolmogorov–Smirnov test and probabilistic seismic hazard parametrization for Nepal and its surrounding regions. Natural Hazards, 1–28 (2024) https://doi.org/10.1007/s11069-024-06531-2

[36] Tél, T., Gruiz, M.: Chaotic Dynamics: an Introduction Based on Classical Mechanics. Cambridge University Press, Cambridge (2006)

[37] Hilborn, R.C.: Chaos and Nonlinear Dynamics: an Introduction for Scientists and Engineers. Oxford University Press, Oxford (2000)

[38] Lorenz, E.N.: Designing Chaotic Models. Journal of the Atmospheric Sciences **62**(5), 1574–1587 (2005) https://doi.org/10.1175/JAS3430.1

[39] Contreras-Reyes, J.E.: Mutual information matrix based on asymmetric Shannon entropy for nonlinear interactions of time series. Nonlinear Dynamics **104**(4), 3913–3924 (2021) https://doi.org/10.1007/s11071-021-06498-w

[40] Lam, S.K., Pitrou, A., Seibert, S.: Numba: A LLVM-based Python JIT compiler. In: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, pp. 1–6 (2015). https://doi.org/10.1145/2833157.2833162

[41] Bartman, P., Banaśkiewicz, J., Drenda, S., Manna, M., Olesik, M.A., Rozwoda, P., Sadowski, M., Arabas, S.: PyMPDATA v1: Numba-accelerated implementation of MPDATA with examples in Python, Julia and Matlab. Journal of Open Source Software **7**(77), 3896 (2022) https://doi.org/10.21105/joss.03896

[42] Clemente-López, D., Munoz-Pacheco, J.M., Jesus, J.R.-M.: Experimental validation of IoT image encryption scheme based on a 5-D fractional hyperchaotic system and Numba JIT compiler. Internet of Things, 101116 (2024) https://doi.org/10.1016/j.iot.2024.101116

[43] Herho, S.H.S., Kaban, S.N., Irawan, D.E., Kapid, R.: Efficient 1D Heat Equation Solver: Leveraging Numba in Python. EKSAKTA: Berkala Ilmiah Bidang MIPA **25**(02), 126–137 (2024) https://doi.org/10.24036/eksakta/vol25-iss02/487

[44] Okuta, R., Unno, Y., Nishino, D., Hido, S., Loomis, C.: CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In: Proceedings of Workshop on Machine Learning Systems (LearningSys) in The 31st Annual Conference on Neural Information Processing Systems (NIPS) (2017). http://learningsys.org/nips17/assets/papers/paper_16.pdf

[45] Mathias, S., Coulier, A., Hellander, A.: CBMOS: a GPU-enabled Python framework for the numerical study of center-based models. BMC Bioinformatics **23**(1), 55 (2022) https://doi.org/10.1186/s12859-022-04575-4

[46] Martin, D.S., Torres, C.: 2D Simplified Wildfire Spreading Model in Python: From NumPy to CuPy. CLEI electronic journal **26**(1), 5–1 (2023) https://doi.org/10.19153/cleiej.26.1.5

[47] Askar, T., Yergaliyev, A., Shukirgaliyev, B., Abdikamalov, E.: Exploring Numba and CuPy for GPU-Accelerated Monte Carlo Radiation Transport. Computation **12**(3), 61 (2024) https://doi.org/10.3390/computation12030061