Drop Angles and Chaos: A Lyapunov Exponent Analysis of the Double Pendulum

How can the Lyapunov exponent show how drop angles influence the onset of chaos in a double pendulum?

Subject: Mathematics

Word Count: 3988

Contents

1 Introduction

The double pendulum, a system governed by classical mechanics, exhibits chaotic behavior, prompting a thorough examination of its dynamics in this essay. The system is sensitive to initial conditions, hence, a microscopic change in its initial conditions makes the difference between periodic and chaotic trajectories. To find out how different drop angles influence the chaotic behavior, we first derive the simulation for the double pendulum. Then, we will examine the Lyapunov exponent, an indicator of chaos, and derive the Lyapunov exponent for different initial conditions of the double pendulum. The result is a heatmap that outlines the relationship between the drop angles and chaos in a double pendulum. Notably, it also uncovers stable, periodic orbits at high-angle drops of the double pendulum. Ultimately, there is a trend that higher drop angles yield more chaos, but the irregular trends that emerge from the analysis serve as a testament to the double pendulum's inherently chaotic nature.

2 Chaos Theory

We begin by learning the background of chaos theory. This branch of mathematics deals with the irregular and unpredictable motions of nonlinear systems. At its core is the butterfly effect, which states that small changes in initial conditions lead to large changes in results. To understand its origins, we explore the story of Edward Lorenz, the founder of chaos theory, and how he discovered chaos, as recounted in [Bradley, 2010].

2.1 The Butterfly Effect

Edward Lorenz, a meteorologist, pioneered the study of chaos theory through his work on weather simulation. While rerunning a simulation, Lorenz truncated initial values from six decimal places to three, expecting negligible differences. However, the output diverged significantly from the original results. Initially suspecting a machine error, Lorenz discovered that the discrepancy stemmed from the minute rounding differences in initial conditions.

This finding illustrated the core concept of chaos theory: sensitivity to initial conditions. Even infinitesimal differences in the starting state of a system can yield drastically distinct outcomes over time. This phenomenon has come to be known as the "butterfly effect," which poetically suggests that the flap of a butterfly's wings in Brazil could change the initial conditions in a way that leads to a hurricane in Texas [Bradley, 2010].

2.2 Defining Chaos

The concept of sensitivity to initial conditions was a foundational element in defining chaos. Over time, a widely accepted definition of chaos emerged: "Chaos is aperiodic long-term behavior in a deterministic system that exhibits sensitive dependence on initial conditions." [Strogatz, 2018]. Let us define the three important aspects in this definition.

- **Aperiodic Long-term Behavior**: Trajectories do not settle down to fixed points or periodic orbits. It implies a lack of repetitive patterns or cycles over an extended period.
- **Deterministic**: The system is not random; given the same inputs, it will always yield the same outputs.
- **Sensitive Dependence on Initial Conditions**: Differences in the initial conditions, no matter how small, will lead to drastically different results.

The third condition here is the key to identifying chaos. The crucial takeaway from this section is that the degree of sensitivity is directly tied to the degree of chaos exhibited by a system. Moving forward, the measure of chaos will be based on this concept. Building upon this understanding, we will now shift our focus to a system that exhibits chaotic behavior: the double pendulum.

3 Derivation of A Double Pendulum

The double pendulum consists of two simple pendulums connected in series. This section derives the simulation of the double pendulum. The double pendulums in this essay will strictly be simple

double pendulum systems, which are systems with point masses connected by rigid, massless rods, ignoring friction. Figure 1 shows the variables involved in the double pendulum.

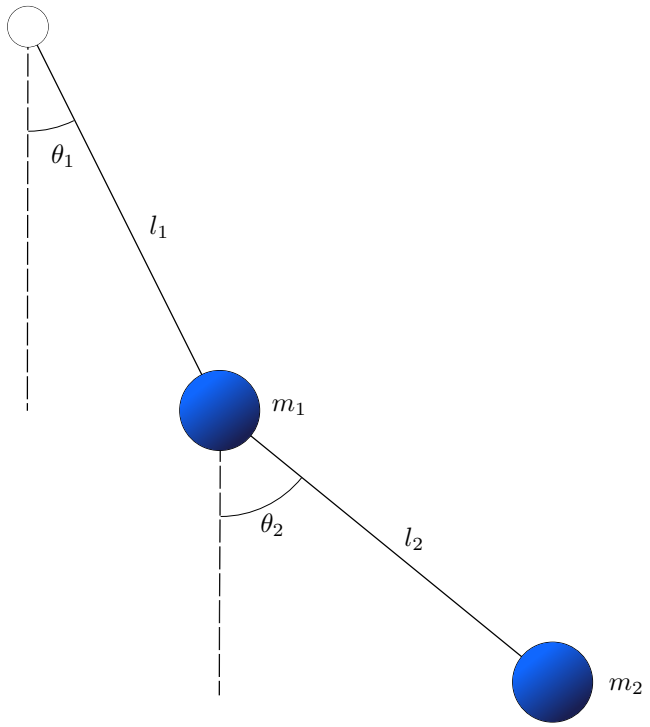


Figure 1: The double pendulum, and the variables involved.

The next subsection aims to derive the equations of motions that are necessary to simulate a double pendulum, based on the methodology presented in [González, 2006].

3.1 Equations of Motion

First, we use trigonometric principles with l_1 as the origin, we express the x and y coordinates of each mass as functions of θ_1 and θ_2. Note that the positive direction along the y-axis is defined as

upward.

$$x_1 = l_1 \sin(\theta_1) \tag{3.1}$$

$$y_1 = -l_1 \cos(\theta_1) \tag{3.2}$$

$$x_2 = l_1 \sin(\theta_1) + l_2 \sin(\theta_2) \tag{3.3}$$

$$y_2 = -l_1 \cos(\theta_1) - l_2 \cos(\theta_2) \tag{3.4}$$

Differentiate with respect to time, use the chain rule, to derive the angular velocity in each direction. The notation $\dot{x}$ represents the time derivative of x, that is, $\dot{x} = \frac{dx}{dt}$.

$$\dot{x}_1 = l_1 \cos(\theta_1)\dot{\theta}_1 \tag{3.5}$$

$$\dot{y}_1 = l_1 \sin(\theta_1)\dot{\theta}_1 \tag{3.6}$$

$$\dot{x}_2 = l_1 \cos(\theta_1)\dot{\theta}_1 + l_2 \cos(\theta_2)\dot{\theta}_2 \tag{3.7}$$

$$\dot{y}_2 = l_1 \sin(\theta_1)\dot{\theta}_1 + l_2 \sin(\theta_2)\dot{\theta}_2 \tag{3.8}$$

Next, derive the Lagrangian, denoted by L. This value is the difference between kinetic energy (T) and potential energy (U).

$$L = T - U \tag{3.9}$$

The formula for the kinetic energy of an object with mass m and velocity v is $\frac{1}{2}mv^2$. Mass m is known, and v can be expressed as the sum of the $\dot{x}$ and $\dot{y}$ components. Thus, we substitute the expressions for $\dot{x}_1, \dot{y}_1, \dot{x}_2$, and $\dot{y}_2$ from equations (3.5) through (3.8) into the kinetic energy equation for both masses. Sum them to get the kinetic energy of the entire system. Lastly, expand and collect like terms to obtain:

$$\begin{aligned} T &= \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 \\ &= \frac{1}{2}m_1(\dot{x}_1^2 + \dot{y}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + \dot{y}_2^2) \\ &= \frac{1}{2}m_1 \ell_1^2 \dot{\theta}_1^2 + \frac{1}{2}m_2 \left(\ell_1^2 \dot{\theta}_1^2 + \ell_2^2 \dot{\theta}_2^2 + 2\ell_1 \ell_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2)\right) \end{aligned} \tag{3.10}$$

As for the potential energy of the system, it is the sum of the gravitational potential energies of each mass, given by mgh, where h is the y component. Expand by referring to angular expressions, and collect the like terms of this expression.

$$\begin{aligned} U &= -m_1 g y_1 - m_2 g y_2 \\ &= -m_1 g l_1 \cos\theta_1 - m_2 g (l_1 \cos\theta_1 + l_2 \cos\theta_2) \\ &= -(m_1 + m_2) g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 \end{aligned} \tag{3.11}$$

Substituting equations 3.10 and 3.11 into equation 3.9, we obtain the Lagrangian:

$$\begin{aligned} L &= T - U \\ &= \frac{1}{2}(m_1 + m_2) l_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_2 - \theta_1) \\ &\quad + (m_1 + m_2) g l_1 \cos(\theta_1) + m_2 g l_2 \cos(\theta_2) \end{aligned} \tag{3.12}$$

We can now use the Lagrangian we have obtained to solve for the angular accelerations $\ddot{\theta_1}$ and $\ddot{\theta_2}$ using the Euler-Lagrangian. The Euler-Lagrangian states that the rate of change of a partial derivative of the Lagrangian with respect to a generalized coordinate (an angle in our case) and its time derivative, minus the partial derivative of the Lagrangian with respect to the generalized coordinate, is equal to zero. This can be expressed mathematically as:

$$0 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1} \tag{3.13}$$

Let us first solve $\frac{\partial L}{\partial \dot{\theta}_1}$. Substitute the values obtained for L and $\dot{\theta_1}$, then use the term-wise differentiation.

$$\frac{\partial L}{\partial \dot{\theta}_1} = (m_1 + m_2) l_1^2 \dot{\theta}_1 + m_2 l_1 l_2 \dot{\theta}_2 \cos(\theta_2 - \theta_1) \tag{3.14}$$

Take the time derivative to solve for $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right)$. Use term-wise differentiation for the first term, and the product rule followed by the chain rule for the second term.

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) &= (m_1 + m_2) l_1^2 \ddot{\theta}_1 + m_2 l_1 l_2 \ddot{\theta}_2 \cos(\theta_2 - \theta_1) \\ &\quad - m_2 l_1 l_2 \dot{\theta}_2^2 \sin(\theta_2 - \theta_1) + m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2 - \theta_1) \end{aligned} \tag{3.15}$$

Lastly, we take the partial derivative of the Lagrangian with respect to θ_1.

$$\frac{\partial L}{\partial \theta_1} = m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2 - \theta_1) - (m_1 + m_2) g l_1 \sin(\theta_1) \tag{3.16}$$

Substituting the expressions for $\frac{\partial L}{\partial \dot{\theta}_1}$, $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right)$ and $\frac{\partial L}{\partial \theta_1}$ into (3.13), we obtain one of the equations of motion.

$$\begin{aligned} 0 = {} & (m_1 + m_2) l_1^2 \ddot{\theta}_1 + m_2 l_1 l_2 \ddot{\theta}_2 \cos(\theta_2 - \theta_1) \\ & - m_2 l_1 l_2 \dot{\theta}_2^2 \sin(\theta_2 - \theta_1) + (m_1 + m_2) g l_1 \sin(\theta_1) \end{aligned} \tag{3.17}$$

Take a similar approach for θ_2 to get the other equation of motion. Begin with the Euler-Lagrangian for θ_2.

$$0 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2} \tag{3.18}$$

Again, to solve for the first term, take the partial derivative of the Lagrangian with respect to $\dot{\theta}_2$. Then, take the time derivatives, followed by the chain rule.

$$\frac{\partial L}{\partial \dot{\theta}_2} = m_2 l_2^2 \dot{\theta}_2 + m_2 l_1 l_2 \dot{\theta}_1 \cos(\theta_2 - \theta_1) \tag{3.19}$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) = {} & m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 \ddot{\theta}_1 \cos(\theta_2 - \theta_1) \\ & + m_2 l_1 l_2 \dot{\theta}_1^2 \sin(\theta_2 - \theta_1) - m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2 - \theta_1) \end{aligned} \tag{3.20}$$

The term $\frac{\partial L}{\partial \theta_2}$ is again the partial derivative of the Lagrangian with respect to θ_2.

$$\frac{\partial L}{\partial \theta_2} = -m_2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2 - \theta_1) - m_2 g l_2 \sin(\theta_2) \tag{3.21}$$

Combining these equations together, we obtain the second equation of motion.

$$\begin{aligned} 0 = {} & m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 \ddot{\theta}_1 \cos(\theta_2 - \theta_1) \\ & + m_2 l_1 l_2 \dot{\theta}_1^2 \sin(\theta_2 - \theta_1) + m_2 g l_2 \sin(\theta_2) \end{aligned} \tag{3.22}$$

We have now derived both equations of motion for the double pendulum in equations 3.17 and 3.22. However, they are not useful in the form they are currently in. They are second-order differential equations, describing the angular accelerations of the system. To simulate the double pendulums, additional work needs will be done with numerical integrators.

3.2 Numerical Integration Methods

Numerical integrators approximate the evolution of a system, given initial conditions and differential equations [Cheever, 1997]. Understanding these play a crucial role in the construction of the simulation, as well as evaluating the correctness of it. This subsection will first cover Euler's method to provide an understanding of numerical integrators, followed by the fourth-order Runge Kutta, which will eventually be implemented for its greater accuracy.

Euler's Method

Euler's method is the simplest form of a numerical integrator. It uses the slope of the differential equation at the current time step to predict the state at the next time step [Cheever, 1997]. So for a differential equation described by $\frac{dy}{dt} = f(y(t), t)$, the Euler's method is

$$y^*(t+h) = y^*(t) + f(y(t), t) \cdot h \tag{3.23}$$

Where, $y^*(t)$ represents a function that estimates the state of the system at the time t, and h is the step size used to advance the time by h units. Figure 2a illustrates how Euler's method approximates the exact solutions.

Fourth-Order Runge-Kutta (RK4)

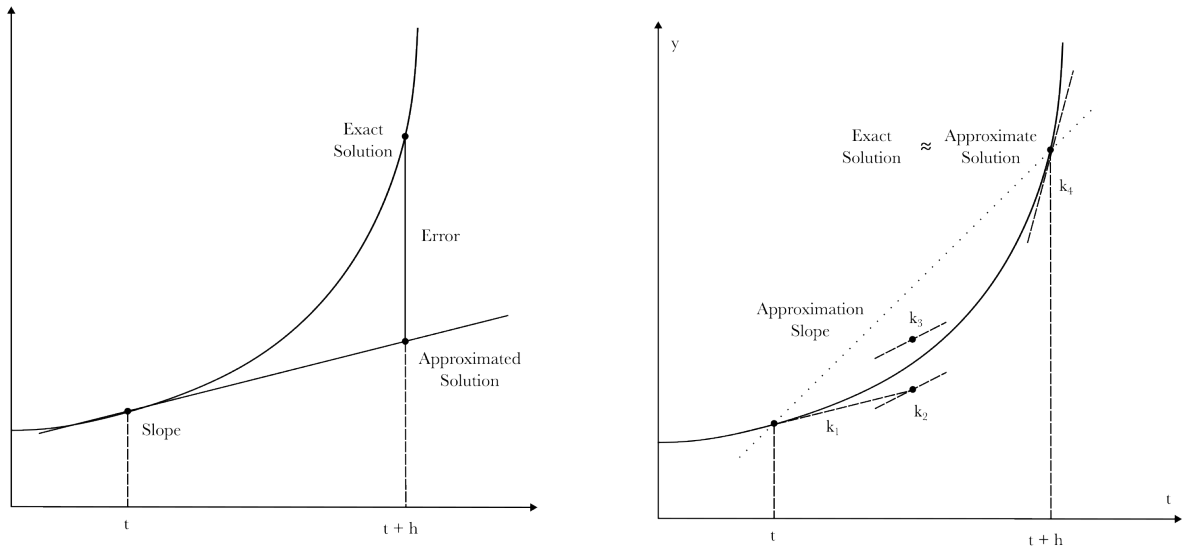
RK4 is a more accurate numerical integrator, estimating the slope by taking a weighted average of four slopes at different points. This is described by

$$y^*(t+h) = y^*(t) + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} h \tag{3.24}$$

Where

$$k_1 = f(y^*(t), t)$$
$$k_2 = f\left(y^*(t) + \frac{k_1 h}{2}, t_0 + \frac{h}{2}\right)$$
$$k_3 = f\left(y^*(t) + \frac{k_2 h}{2}, t_0 + \frac{h}{2}\right)$$
$$k_4 = f\left(y^*(t) + k_3 h, t_0 + h\right)$$

The RK4 is a popular choice due to its balance between accuracy and computational demands [D'Alessio, 2023]. Figure 2b outlines how it works. Notice that there is less error, hence, it will be the numerical integrator we will use.



(a) Euler's method of numerical integration.

(b) Fourth-Order Runge-Kutta integration.

Figure 2: Comparison of Euler's integration and the Fourth-Order Runge-Kutta.

3.3 Implementation

With the equations of motion derived and numerical methods understood, we now combine the pieces to formulate the final simulation in this subsection. We represent the state of the double

pendulum using a state vector.

$$\mathbf{Y} = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2] \tag{3.25}$$

The time derivative of this vector, $\dot{\mathbf{Y}}$, represents the slope.

$$\dot{\mathbf{Y}} = [\dot{\theta}_1, \dot{\theta}_2, \ddot{\theta}_1, \ddot{\theta}_2] \tag{3.26}$$

Variables $\dot{\theta}_1$ and $\dot{\theta}_2$ are already known from the state vector, but we will need to derive $\ddot{\theta}_1$ and $\ddot{\theta}_2$ from the Lagrangian equations of motion. We proceed as follows.

Isolate $\ddot{\theta}_1$ in equation (3.17). Do the same for $\ddot{\theta}_2$ in equation (3.22), which were the equations of motion. This yields the following two equations for the second-order derivatives.

$$\ddot{\theta}_1 = -\frac{g m_1 \sin(\theta_1) + g m_2 \sin(\theta_1) + l_2 m_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) + l_2 m_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2)}{l_1 (m_1 + m_2)} \tag{3.27}$$

$$\ddot{\theta}_2 = \frac{-g \sin(\theta_2) - l_1 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) + l_1 \dot{\theta_1}^2 \sin(\theta_1 - \theta_2)}{l_2} \tag{3.28}$$

However, the expressions for $\ddot{\theta}_1$ and $\ddot{\theta}_2$ contain additional second-order derivatives, which are not directly available from the state vector. To resolve this, we can substitute the expressions for $\ddot{\theta}_1$ and $\ddot{\theta}_2$ into each other, and then isolate the terms to obtain equations that express $\ddot{\theta}_1$ and $\ddot{\theta}_2$ solely in terms of $\dot{\theta}_1$, $\dot{\theta}_2$, θ_1, and θ_2. This yields

$$\begin{aligned} \ddot{\theta}_1 = &\frac{2 g m_1 \sin(\theta_1) + g m_2 \sin(\theta_1) + g m_2 \sin(\theta_1 - 2\theta_2)}{2 l_1 (-m_1 + m_2 \cos^2(\theta_1 - \theta_2) - m_2)} \\ &+ \frac{l_1 m_2 \dot{\theta}_1^2 \sin(2\theta_1 - 2\theta_2) + 2 l_2 m_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2)}{2 l_1 (-m_1 + m_2 \cos^2(\theta_1 - \theta_2) - m_2)} \end{aligned} \tag{3.29}$$

$$\begin{aligned} \ddot{\theta}_2 = &\frac{-g m_1 \sin(\theta_2) + g m_1 \sin(2\theta_1 - \theta_2) - g m_2 \sin(\theta_2) + g m_2 \sin(2\theta_1 - \theta_2)}{2 l_2 (m_1 - m_2 \cos^2(\theta_1 - \theta_2) + m_2)} \\ &+ \frac{2 l_1 m_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + 2 l_1 m_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + l_2 m_2 \dot{\theta}_2^2 \sin(2\theta_1 - 2\theta_2)}{2 l_2 (m_1 - m_2 \cos^2(\theta_1 - \theta_2) + m_2)} \end{aligned} \tag{3.30}$$

Now, each variable in the slope approximation $\dot{\mathbf{Y}}$ can be expressed in values from the state vector, with $\ddot{\theta}_1$ and $\ddot{\theta}_2$ coming from equations 3.29 and 3.30. Combining this with the RK4, we can simulate the double pendulum's motion at each discrete time interval.

For example, we can plot the angular displacements over time for some initial conditions, shown in Figure 3.

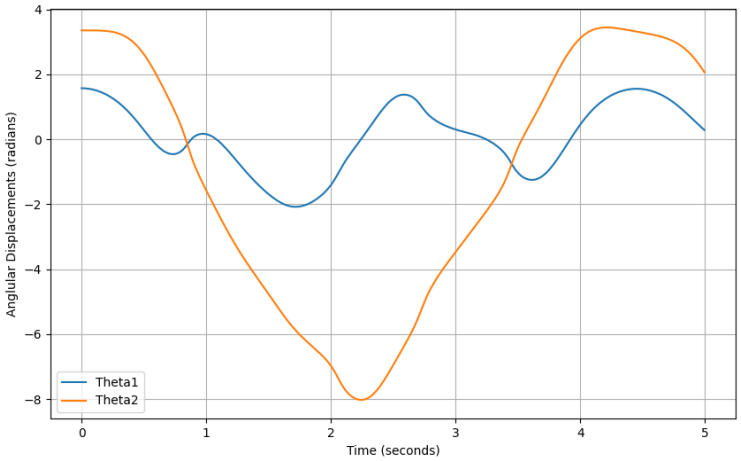


Figure 3: Evolution of some θ_1 and θ_2.

To visualize the pendulum's motion in the xy plane, we can plug the angles back into the trigonometric equations (equations 3.1 to 3.4). This gives a more visually realistic representation, as shown in Figure 4. The gray trajectory shows the where paths that the second mass on the pendulum has traversed.

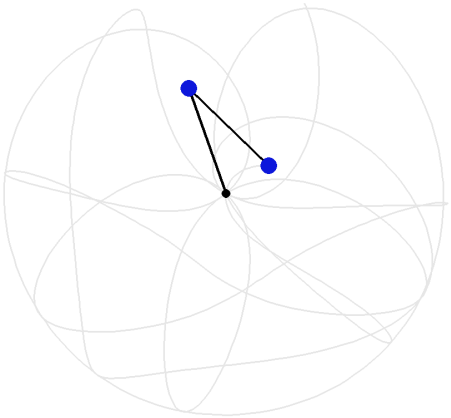


Figure 4: Simulation of the double pendulum in the xy plane.

To verify the correctness of our simulation, we evaluate the total energy of the system at each time step by computing the sum of kinetic energy (Equation 3.10) and potential energy (Equation 3.11) using the state vector variables. As exhibited in Figure 5, the total energy remains constant, indicating the reliability of our simulation.

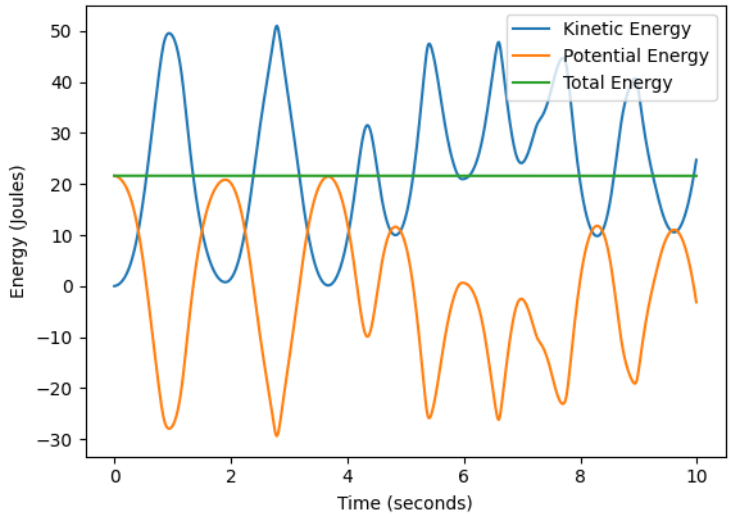


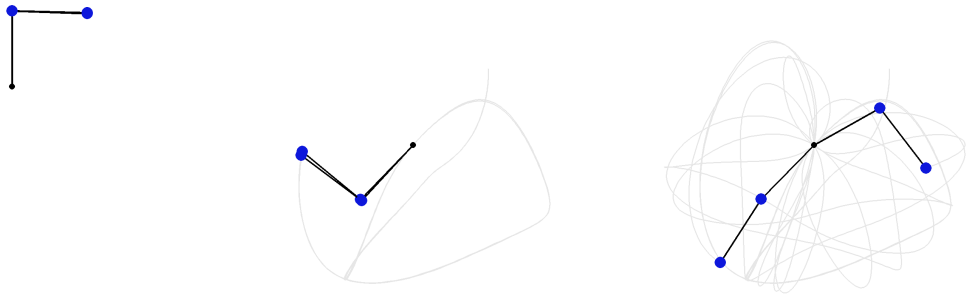
Figure 5: Total energy of the system over time.

The simulation is now complete, and the code can be found in appendix 9.

4 Lyapunov exponent

To investigate the relationship between drop angles and chaos, a quantitative measurement of chaos is needed. This section aims to intuitively introduce the Lyapunov exponent, an indicator of chaos, that measures the sensitivity of the system to initial conditions.

First, recall that chaotic systems are sensitive to initial conditions. We can use the simulation we have derived to visually demonstrate how the double pendulum exhibits this trait. This can be done by dropping two double pendulums with very close initial conditions.



(a) Two double pendulums separated by $\theta_1 \pm 0.01$ radians.

(b) After three seconds.

(c) After ten seconds.

Figure 6: Two double pendulums' trajectories over time.

The results are shown in Figure 6. From initial conditions of θ_1 with separations of 0.01 radians, indiscernible to the eye, to quickly being on different ends of the plane in only ten seconds. This underscores the double pendulum's pronounced sensitivity to initial conditions. A natural follow-up question arises: Could we grasp a more quantitative measurement of how sensitive the system is to compare the chaos between different initial conditions?

To further investigate, we plot the time evolution of the variable θ_1 from the two double pendulums, as shown in Figure 7. The angle trajectories appear to coincide initially, only to diverge significantly over time. Is there a way to quantify this separation? Perhaps, how fast they diverge?

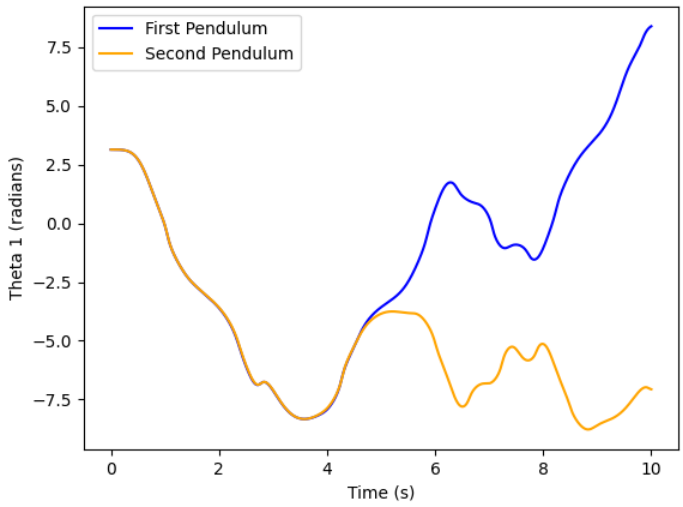


Figure 7: Divergence of θ_1 for two double pendulums with separation of initial conditions of 0.01 radians.

The answer is yes. We have stumbled upon the Lyapunov exponent. The Lyapunov exponent serves as the quantitive indicator of how sensitive a system is to its initial conditions. It does this by measuring the rate of divergence of variables of close initial conditions. However, this analysis needs to be done on all variables, not just one (like our θ_1). Phase space is the concept that enables us to perform this multidimensional calculation. Let us grasp an understanding of phase space first before continuing further with the Lyapunov exponent.

4.1 Phase Space

Phase space is a mathematical representation of all possible states of a physical system. It provides a way to analyze the behavior of complex systems, such as the double pendulum. In this context, the state vector $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ forms a four-dimensional phase space, where each dimension corresponds to a system variable. Each point in this space represents a specific system state, with each axis specifying a particular value for each variable.

To measure the separation of two points in phase space, we will need to find the Euclidean distance.

The Euclidean distance between two points $\mathbf{P} = (x_1, y_1, \ldots, z_1)$ and $\mathbf{Q} = (x_2, y_2, \ldots, z_2)$ in an n-dimensional space is given by:

$$d(\mathbf{P}, \mathbf{Q}) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \ldots + (z_2 - z_1)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

This calculation takes into account the multidimensional nature of our phase space. With the tools we have learned so far, we can begin to define the Lyapunov exponent.

4.2 Defining the Lyapunov Exponent

Consider a dynamical system described by the differential equation $\frac{dx}{dt} = f(x)$. Let x_t be the state of the system at time t, and let $f^t(x)$ denote the trace of iterating f t times. This evolution is called a trajectory, or an orbit.

Suppose we have an initial condition x_0 and a nearby initial condition $x_0 + u_0$, where u_0 is a small perturbation. We can iterate f to evolve these initial conditions, resulting in two trajectories that diverge or converge over time. Let u_t denote the Euclidean distance between these trajectories at time t. Figure 8 illustrates the separation of trajectories and the change in u_t.

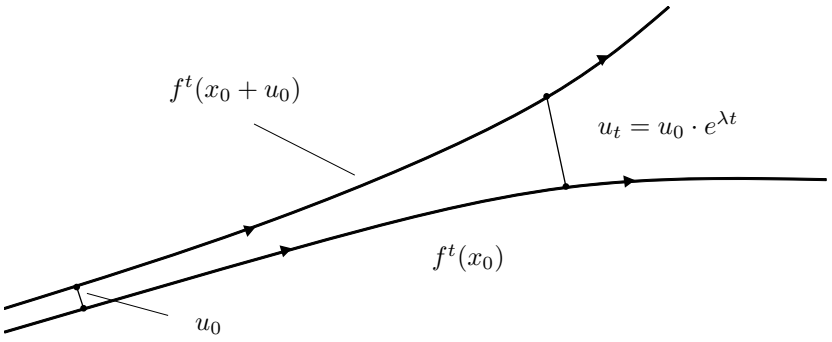


Figure 8: Separation of two trajectories with closeby initial conditions.

This growth or decay of u_t can be modeled by the exponential equation

$$u_t = u_0 \cdot e^{\lambda t} \tag{4.1}$$

In this equation, λ is the Lyapunov exponent. The larger the value of the exponent is, the faster

the rate of divergence. We can isolate λ by dividing both sides by $d(t_0)$, taking the natural logarithm, and dividing by t:

$$u_t = u_0 \cdot e^{\lambda t} \tag{4.2}$$

$$\frac{u_t}{u_0} = e^{\lambda t} \tag{4.3}$$

$$\ln\left(\frac{u_t}{u_0}\right) = \lambda t \tag{4.4}$$

$$\lambda = \frac{1}{t}\ln\left(\frac{u_t}{u_0}\right) \tag{4.5}$$

In this form, the Lyapunov exponent can be calculated using the initial separation and final separation.

Furthermore, we can categorize the values of the Lyapunov exponent and describe what they reveal about the initial conditions of a dynamical system [Elert, 2010].

- $\boldsymbol{\lambda > 0}$: Since there is exponential growth, initial conditions are extremely sensitive, with even infinitesimally small variations leading to drastically different outcomes. This extreme sensitivity is a hallmark of chaotic behavior, and even minute errors in initial angle, angular velocity, or other parameters can result in drastically different trajectories.

- $\boldsymbol{\lambda < 0}$: Since there is exponential decay, the system is not sensitive to initial conditions, with trajectories starting from slightly different initial conditions converging to a fixed point or orbit. This characteristic is typical of dissipative or non-conservative systems, where energy is lost over time.

- $\boldsymbol{\lambda = 0}$: There is neither growth or decay. The system exhibits stable, periodic behavior, with the separation between initially close trajectories remaining constant. This scenario arises when the system's frequencies are amplitude-independent, as observed in ideal harmonic oscillators.

The magnitude of the exponent reveals how fast the separation or convergence is. To provide context, below are the Lyapunov exponents for different systems. The Lorenz system that kickstarted

the branch of chaos had a Lyapunov exponent of 0.9 [Viswanath, 2007], and another famous chaotic system, the Henon map, has a Lyapunov exponent of 0.69 [Viswanath, 2007].

5 Numerical Simulation

Finally, with an understanding of the Lyapunov exponent as well as having derived the double pendulum, this section combines both to outline the methodology for finding the Lyapunov exponent for different initial conditions.

Following the definition of the Lyapunov exponent outlined in the previous section, the intuitive method to find the Lyapunov exponent goes as follows: Start with two nearby initial conditions, evolve them over time, and find how the separation changes over time to plug into the Lyapunov equation. However, this method has one key limitation. Due to the physically constrained environment that the double pendulums are in, there is an upper bound on the divergence of the variables. As a result, their growth apart will be limited, leading to an underestimation of the Lyapunov exponent [Wolf et al., 1985].

To account for these limitations, [Sprott, 1997] came up with the methodology of drawing the trajectories closer together immediately after measuring the separation, preventing the trajectories from running into a limitation. Let us formally define this with this procedure.

1. Choose an initial condition as the fiducial trajectory starting point and a nearby initial condition as the non-fiducial trajectory starting point. Let their separation be u_0.

2. Evolve both trajectories for a selected period. For this simulation, the period used is 1 second, to balance between finding small separations and mitigating the underestimation of the Lyapunov exponent in running into physical boundaries.

3. Calculate the final separation, then use equation (4.5) to obtain the maximum Lyapunov exponent at that point.

4. Adjust the non-fiducial orbit to converge towards the fiducial orbit, eliminating the restrictive effect of physical barriers. The relocation of trajectories is as follows: Suppose x is a variable in the state vector. Let orbit a be the fiducial trajectory with $x = x_{a0}$, and orbit b be the adjusted trajectory with $x = x_{b0}$. When orbit b is drawn back, the state new variable x_{b1} is reinitialized to:

$$x_{b1} = x_{a0} + \frac{u_0}{u_t}(x_{b0} - x_{a0}).$$

This equation is derived from [Sprott, 1997]. Repeat this process for every variable in the state vector.

5. We will repeat steps 2-4 for 30 seconds for every initial condition, adjusting the non-fiducial trajectory as the fiducial trajectory evolves. The final Lyapunov exponent will be the average Lyapunov exponent over the entire trajectory.

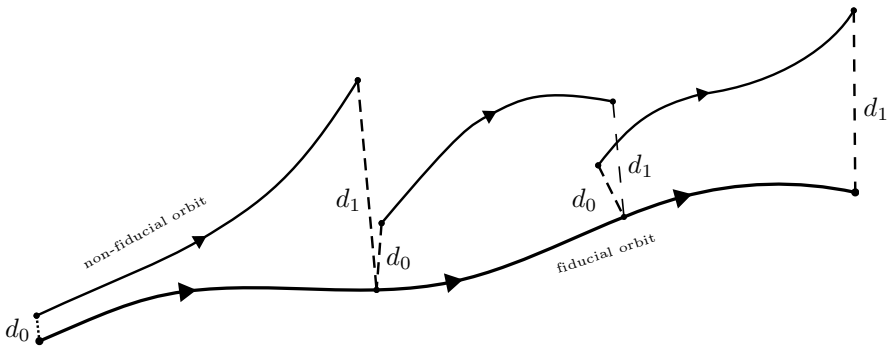


Figure 9: Schematic of the Lyapunov exponent calculation.

A schematic representing this entire process can be seen in Figure 9. By adjusting the fiducial orbit, we effectively mitigate the effects of physical barriers in calculating the Lyapunov exponent. We can incorporate the concept of the average onto the definition of the Lyapunov exponent from equation (4.5) by dividing the sum of all Lyapunov exponents by the total number of exponents.

$$\lambda = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{t}\ln\left(\frac{u_i}{u_{i-1}}\right) \tag{5.1}$$

Further simplifying, the value for the change in time, $\frac{1}{t}$, is factored out of the summation. Multiplying this value with the number of iterations, $\frac{1}{n}$, returns the reciprocal of the total time, which we will denote as $\frac{1}{T}$. This gives us

$$\begin{aligned}\lambda &= \frac{1}{n}\frac{1}{t}\sum_{i=1}^{n}\ln\left(\frac{u_i}{u_{i-1}}\right) \\ &= \frac{1}{T}\sum_{i=1}^{n}\ln\left(\frac{u_i}{u_{i-1}}\right)\end{aligned} \tag{5.2}$$

We now obtained and understand all of the tools required to find the Lyapunov exponent of each drop angle of a double pendulum.

6 Results

This section presents the results of our investigation into the double pendulum's chaotic behavior, using the methodology outlined previously.

The heatmap in Figure 10 shows the Lyapunov exponents for the double pendulum with varying initial drop angles θ_1 and θ_2, each ranging from 0 to 2π radians. The x-axis and y-axis represent the drop angles of θ_2 and θ_1, respectively, with a resolution of 1260×1260 (calculating the Lyapunov exponent for 1,587,600 values). The color scheme ranges from dark red (low, near-zero Lyapunov exponents) to white (high Lyapunov exponents, maximum of 2.358).

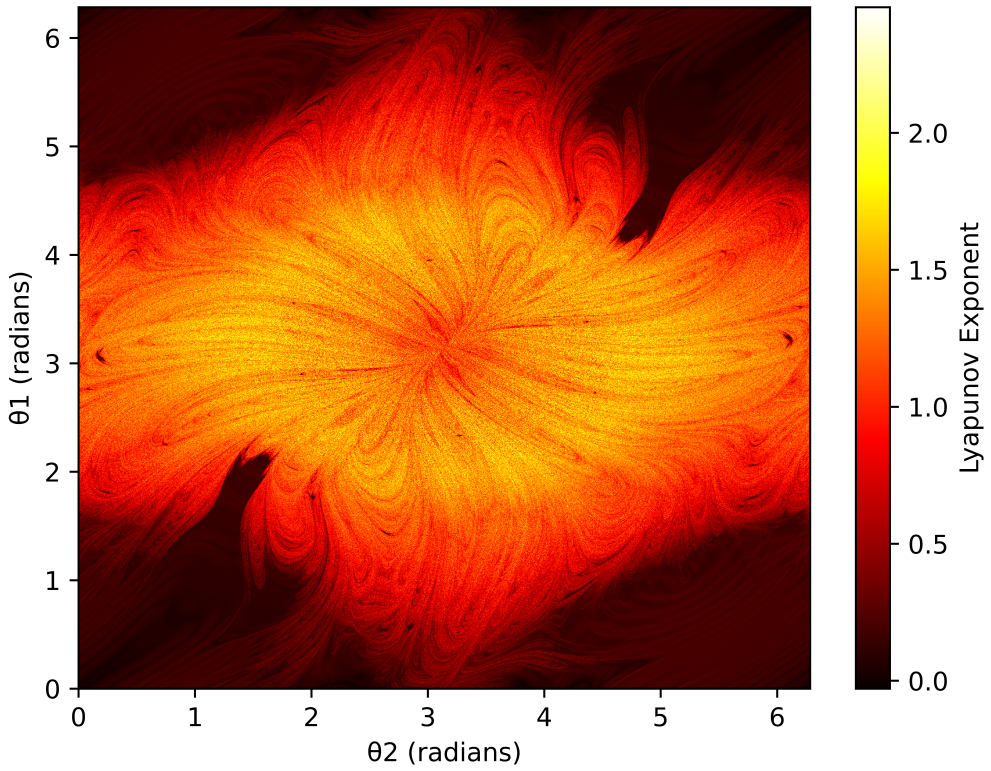
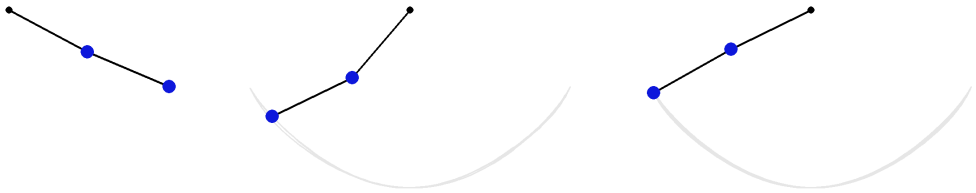


Figure 10: Heatmap of the Lyapunov exponent at different drop angles.

The heatmap uncovers the answer to the research question, of how the initial conditions affect the onset of chaos. Drop angles that yield bright spots are chaotic, while drop angles that yield dark spots are periodic. Notice how the heatmap is mirrored at the line $\theta_1 = \pi$. This is because the double pendulum is symmetric around the vertical axis, and θ_1 from π to 2π is a reflection of θ_1 from 0 to π. This means that analysis of the bottom half applies to the top half as well. We will now analyze what the heatmap of Lyapunov exponents reveals about the relationship between drop angles and chaos.

First, notice the concentration of lower Lyapunov exponents at the corners of the heatmap, where the drop angle is low (2π is a low drop angle as the pendulum does a circle to the other side). Plotting the double pendulum with parameters from these regions (Figure 11), we see it resembles a single pendulum. Notice that these orbits are periodic, corresponding to their low Lyapunov exponents.



(a) Initial conditions. (b) After ten seconds. (c) After thirty seconds.

Figure 11: Low angle drop with $\theta_1 = 1.078$ radians, $\theta_2 = 1.173$ radians, and $\lambda \approx 0.104$.

We move closer to the centre, to the intermediate point between the periodic corners and chaotic centres. Notice the dark patch of Lyapunov exponents that protrude from the bottom corner. There is a harsh border between this patch and the bright, higher Lyapunov exponents. We examine two trajectories from either side of this border, differing by only 0.01 radians in their initial drop angle of θ_2. These are shown in Figure 13a and Figure 13b.

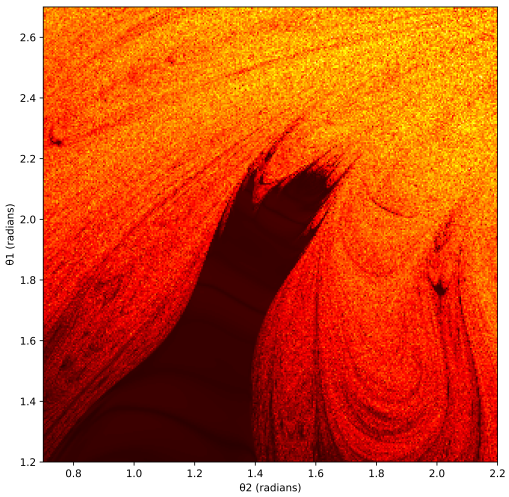


Figure 12: The dark patch of low Lyapunov exponents bordering the chaotic region.

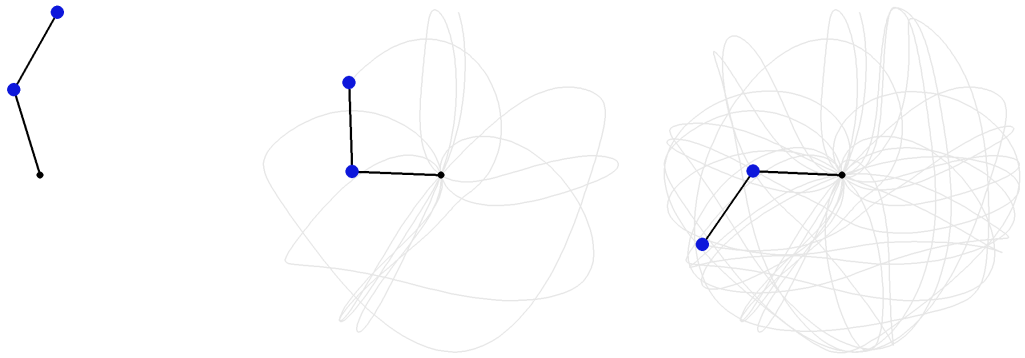
(a) Periodic orbit. $\theta_1 \approx 2.021$ radians, $\theta_2 \approx 1.637$ radians, $\lambda \approx 0.245$

(b) Chaotic orbit. $\theta_1 \approx 2.021$ radians, $\theta_2 \approx 1.647$ radians, $\lambda \approx 1.381$

Figure 13: Two double pendulums with initial conditions on the chaotic and periodic side of the dark patch.

The difference in these two trajectories only after thirty seconds demonstrates the sensitivity to initial conditions in the double pendulum. Notice the chaotic orbit initially had traces of swinging like a single pendulum, evident from the gray streaks. However, it quickly decayed into chaos. This stark contrast in behavior, triggered by a mere 0.01-radian difference in initial conditions, marks the border beyond which chaos emerges.

Moving further towards to centre, where $\theta_1 \approx \pi$ and $\theta_2 \approx \pi$, yields higher Lyapunov exponents. These drop angles produce patterns that are more chaotic, shown in Figure 14.



(a) Initial conditions. (b) After ten seconds. (c) After thirty seconds.

Figure 14: High angle drop that is chaotic, with $\theta_1 \approx 3.439$ radians, $\theta_2 \approx 2.630$ radians, and $\lambda \approx 2.161$.

We have now visually observed the trend of higher drop angles leading to higher degrees of chaos. However, due to the double pendulum's chaotic nature, there are notable regions on the map that punctuate this trend. In the high-angle drop angles, where there is expected chaos, lie regions of dark spots — low Lyapunov exponents. Some of these spots are shown in Figure 15.

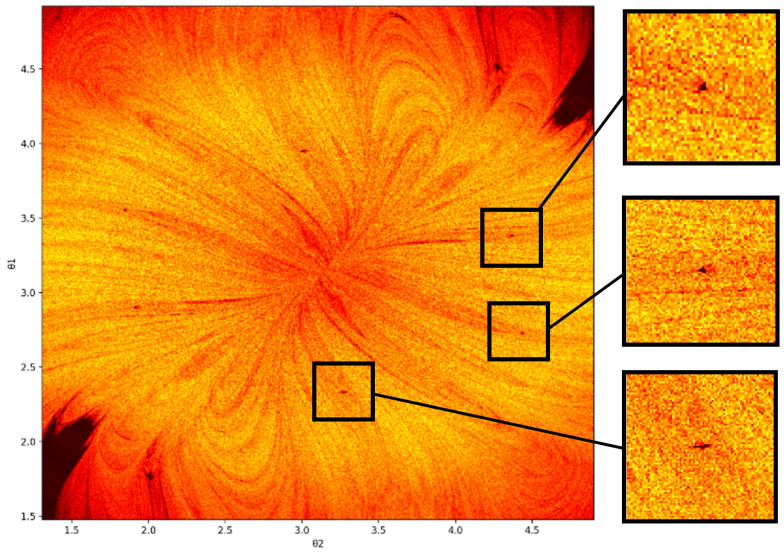
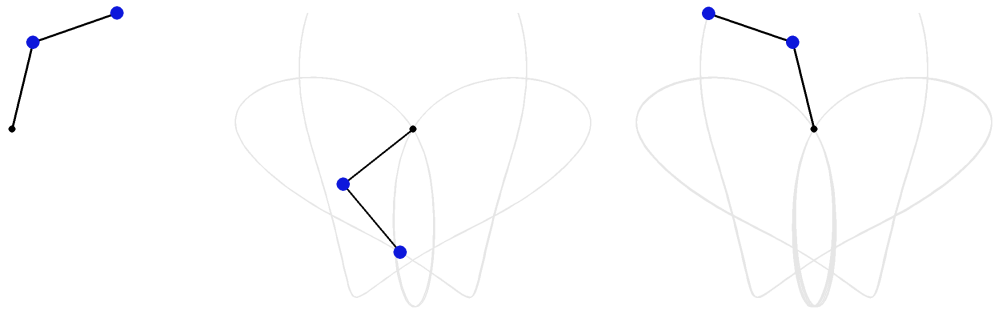


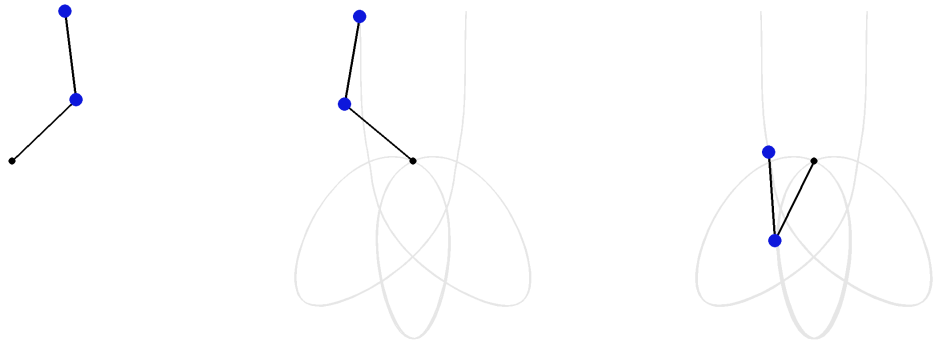
Figure 15: Heatmap origins of high angle periodic orbits, or dark spots in regions of high Lyapunov exponents.

We proceed by examining the outcomes of these initial conditions. The resulting patterns are noteworthy for their unexpected periodicity in regions where chaotic behavior is anticipated. Notice that the trajectories repeat each other, going back on the same trajectories to make the same pattern. There exists an exhaustive list of these stable orbits. A selection of the most intriguing cases is exhibited in the following figures. Figures 16 and 17 demonstrate the patterns at intervals of zero, ten and thirty seconds to showcase the evolution of the periodic orbits.



(a) Initial conditions. (b) After ten seconds. (c) After thirty seconds.

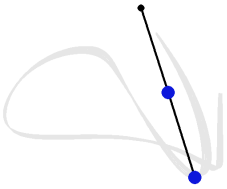
Figure 16: High angle drop that is periodic, with $\theta_1 \approx 2.900$ radians, $\theta_2 \approx 1.911$ radians, and $\lambda \approx 1.911$



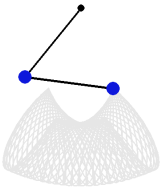
(a) Initial conditions. (b) After ten seconds. (c) After thirty seconds.

Figure 17: High angle drop that is periodic, with $\theta_1 \approx 2.331$ radians, $\theta_2 \approx 3.269$ radians, and $\lambda \approx 0.158$.

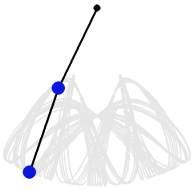
The following figure shows more periodic orbits and their final patterns.



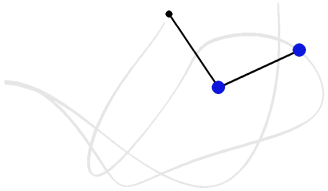
(a) Hook shape. $\theta_1 \approx 1.602$ radians, $\theta_2 \approx 6.193$ radians, $\lambda \approx 0.200$.



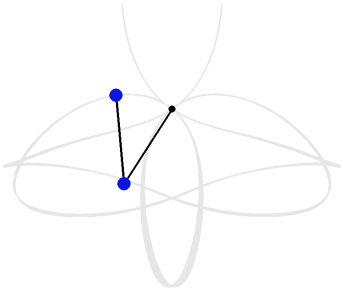
(b) Tunnel. $\theta_1 \approx 0.679$ radians, $\theta_2 \approx 4.836$ radians, $\lambda \approx 0.174$.



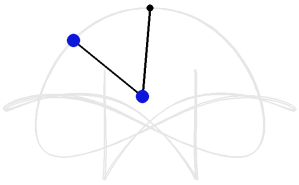
(c) Loops. $\theta_1 \approx 0.838$ radians, $\theta_2 \approx 4.801$ radians, $\lambda \approx 0.135$.



(d) Asymmetric. $\theta_1 \approx 1.048$ radians, $\theta_2 \approx 4.292$ radians, $\lambda \approx 0.178$.



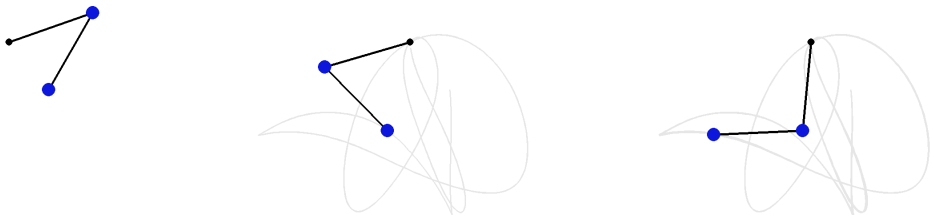
(e) Flower shape. $\theta_1 \approx 2.725$ radians, $\theta_2 \approx 4.442$ radians, $\lambda \approx 0.089$.



(f) Cat whiskers. $\theta_1 \approx 1.757$ radians, $\theta_2 \approx 5.784$ radians, $\lambda \approx 0.216$.

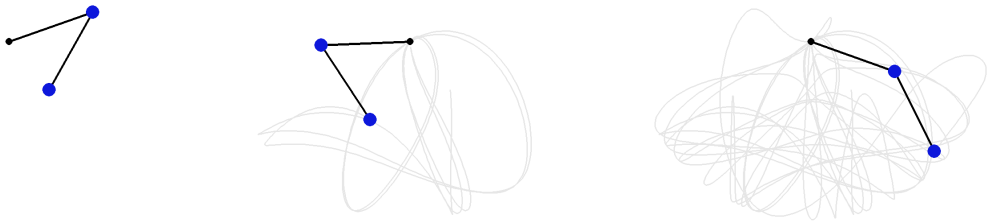
Figure 18: High-angle periodic patterns.

Moreover, some of these dark regions are just one pixel of the heatmap. That means any slight deviation of initial drop angles will yield a chaotic system.



(a) Initial conditions. (b) After ten seconds. (c) After thirty seconds.

Figure 19: High angle drop that is periodic, with $\theta_1 \approx 1.906$ radians, $\theta_2 \approx 5.764$ radians, and $\lambda \approx 0.252$.



(a) Initial conditions. (b) After ten seconds. (c) After thirty seconds.

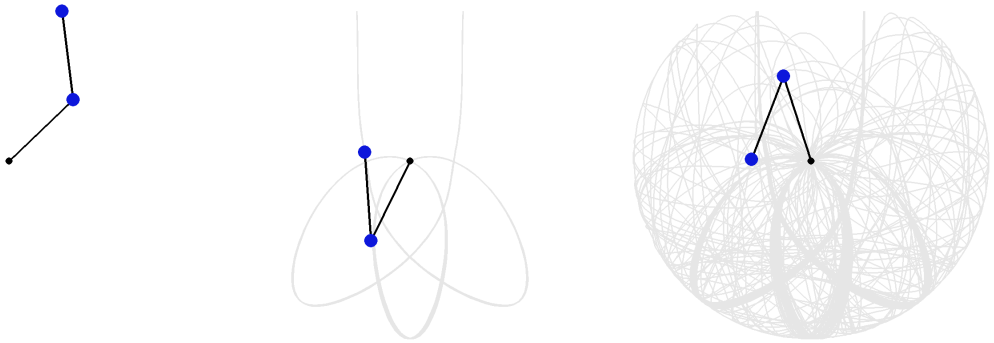
Figure 20: Nearly the same drop angle as in Figure 19, except variance in θ_1 by 0.002 radians. The result after 10 seconds is similar at first but quickly decays into chaos in the next 30 seconds. In fact, the Lyapunov exponent for this drop angle is 1.118.

Overall, this heatmap encapsulates the chaotic nature of the double pendulum, showing its sensitivity to initial conditions — tiny fluctuations in the initial conditions making the difference between chaotic and periodic trajectories.

7 Conclusion

Our investigation into the double pendulum's nonlinear dynamics has yielded a comprehensive understanding of the relationship between initial conditions and chaotic behavior. The Lyapunov exponent heatmap provides a visual representation of the system's sensitivity to initial drop angles, revealing the unpredictable dynamic between these parameters and the onset of chaos. Notably, our results show a general trend of increasing chaotic behavior with higher drop angles. Nonetheless, this pattern is punctuated by regions of periodic orbits at high-angle drops, underscoring the chaos that lies in the dynamic system of a double pendulum.

However, this study has several limitations. The heatmap's applicability is restricted to its specific pendulum condition: 1-meter lengths, 1-kg point masses, and a frictionless environment. Additionally, the orbits are not truly periodic, as they inevitably deviate into chaos. Specifically, the Lyapunov exponent calculations are only valid for the 30-second simulation duration; extending the simulation reveals behavior that is unaccounted for during the calculation of the Lyapunov exponent. We demonstrate this by allowing one of the periodic trajectories outlined in Figure 17 for a longer duration.



(a) Initial conditions. (b) After thirty seconds. (c) After two hundred seconds.

Figure 21: Triple ring periodic trajectory from Figure 17 decaying into chaos.

The trajectory's rapid descent into chaos underscores the heatmap's limited validity, which is confined to a short duration.

Furthermore, if we refer back to Figure 2, we see the numerical integrator we used may not provide perfectly accurate results, as it only provides predictions at discrete time intervals — unlike the continuous process in reality. These outlined limitations are important to consider when evaluating this study's results over extended periods.

Notwithstanding these limitations, our study yields valuable insights and meaningful conclusions. It beckons us to confront the inherent limitations of predictability in complex systems. It challenges our deterministic intuition, inviting us to grapple with the inherent uncertainties that underlie the seemingly deterministic world we inhabit. If the addition of one pendulum onto another can create such unpredictable chaos, what about the world around us?

Perhaps these mathematical ideas stemming from chaos theory provide insight into the way we should approach life. There is only so much we can know about the initial conditions and parameters of our lives. So in a world where the conditions are more complex than a double pendulum, is it worthwhile to have worries that extend beyond what is predictable? That is one of the key lessons that chaos theory can teach us. The sensitivity to initial conditions demonstrated by the Lyapunov exponent in a double pendulum reflects the valleys of unpredictableness around us.

8 Bibliography

[Bradley, 2010] Bradley, L. (2010). The butterfly effect. https://www.stsci.edu/ lbradley/seminar/butterfly.html.

[Cheever, 1997] Cheever, E. (1997). Approximation of differential equations by numerical integration. https://lpsa.swarthmore.edu/NumInt/NumIntIntro.html.

[D'Alessio, 2023] D'Alessio, S. (2023). An analytical, numerical and experimental study of the double pendulum. *Eur. J. Phys.*, 44.

[Elert, 2010] Elert, G. (2010). The chaos hypertextbook. https://hypertextbook.com/chaos/.

[González, 2006] González, G. (2006). Single and double plane pendulum. *Louisiana State University.* https://www.phys.lsu.edu/faculty/gonzalez/Teaching/Phys7221/DoublePendulum.pdf.

[Sprott, 1997] Sprott (1997). Numerical calculation of largest lyapunov exponent. https://sprott.physics.wisc.edu/chaos/lyapexp.htm.

[Strogatz, 2018] Strogatz, S. (2018). *Nonlinear Dynamics and Chaos.* Westview Press.

[Viswanath, 2007] Viswanath, D. (2007). Lyapunov exponents from random fibonacci sequences to the lorenz equations. *Cornell University.*

[Wolf et al., 1985] Wolf, A., Swift, J., Swinney, H. L., and Vastano, J. A. (1985). Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, 16:285–317.

9 Appendix

Code For Double Pendulum

```
import numpy as np
from math import cos, sin, pi
import pygame
import sys
def G(y,t):
    theta1, theta2, omega1, omega2 = y
    f1 = omega1
    f2 = omega2
    f3 = (-g * (2 * m1 + m2) * sin(theta1) - m2 * g * sin(theta1 - 2
        * theta2) - 2 * sin(theta1 - theta2) * m2 * (f2**2 * l2 + f1
        **2 * l1 * cos(theta1 - theta2))) / (l1*(2*m1 + m2 - m2 * cos
        (2*theta1 - 2* theta2)))
    f4 = (2 * sin(theta1-theta2) * (f1 ** 2 * l1 * (m1 + m2) + g* (m1
         + m2) * cos(theta1) + f2**2 * l2 * m2 * cos(theta1- theta2)))
         / (l2 * (2 * m1 + m2 - m2 * cos(2*theta1 - 2* theta2)))

    return np.array([f1, f2, f3, f4])

def RK4_step(y, t, dt):
    k1 = G(y,t)
    k2 = G(y+0.5*k1*dt, t+0.5*dt)
    k3 = G(y+0.5*k2*dt, t+0.5*dt)
    k4 = G(y+k3*dt, t+dt)
```

```
    return dt * (k1 + 2*k2 + 2*k3 + k4)/6


def energy(y):
    theta1, theta2, omega1, omega2 = y
    T = 0.5* (m1 + m2) * l1**2 * omega1**2 + (m2/2) * l2**2 * omega2
        **2 + m2*l1*l2*omega1*omega2*cos(theta1-theta2)
    U = - (m1+m2)*l1*g*cos(theta1) - m2*l2*g*cos(theta2)
    return T + U


def update(a1, a2):
    scale = 200 * (2 / (l1 + l2))
    x1 = l1*scale * sin(a1) + offset[0]
    y1 = l1*scale * cos(a1) + offset[1]
    x2 = x1 + l2*scale * sin(a2)
    y2 = y1 + l2*scale * cos(a2)

    return (x1, y1), (x2, y2)


def render(point1, point2):
    scale = 15
    x1, y1, = int(point1[0]), int(point1[1])
    x2, y2, = int(point2[0]), int(point2[1])

    if prev_point:
        xp, yp = prev_point[0], prev_point[1]
```

```
            pygame.draw.line(trace, TRAILCOLOUR, (xp, yp), (x2, y2), 3)

        screen.fill(WHITE)
        screen.blit(trace, (0,0))

        pygame.draw.line(screen, BLACK, offset, (x1,y1), 5)
        pygame.draw.line(screen, BLACK, (x1,y1), (x2,y2), 5)
        pygame.draw.circle(screen, BLACK, offset, 8)
        pygame.draw.circle(screen, COLOUR1, (x1, y1), int(m1*scale))
        pygame.draw.circle(screen, COLOUR2, (x2, y2), int(m2*scale))

        return (x2, y2)

# variables
m1, m2 = 1.0, 1.0
l1, l2 = 1.0, 1.0
g = 9.81

w, h = 850, 850
WHITE = (255,255,255)
BLACK = (0,0,0)
COLOUR1 = (13, 23, 219)
COLOUR2 = (13, 23, 219)
TRAILCOLOUR = (230,230,230)
offset = (h/2, w/2)
prev_point = None
```

```
screen = pygame.display.set_mode((w,h))
screen.fill(WHITE)
trace = screen.copy()
pygame.display.update()
clock = pygame.time.Clock()


delta_t = 0.01 # increment
totalTime = 100
t = 0
time = np.arange(0, totalTime, delta_t)
# initial state
initial = np.array([3.7, 2.011, 0, 0])
pygame.font.init()
myfont = pygame.font.SysFont('Comic-Sans-MS', 38)


animation_started = False
initial = np.array([3.7, 2.011, 0, 0])
while True:


    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
    point1, point2 = update(initial[0], initial[1])
    prev_point = render(point1, point2)


    t += delta_t
```

```
    initial = initial + RK4_step(initial, t, delta_t)


    clock.tick(60)
    pygame.display.update()
```

Lyapunov Exponent Calculator

```
import numpy as np
from numpy.linalg import inv
from matplotlib import pyplot as plt
from math import cos, sin, pi


def G(y,t):
    a1d, a2d = y[0], y[1]
    a1, a2 = y[2], y[3]


    m11, m12 = (m1+m2)*l1, m2*l2*cos(a1-a2)
    m21, m22 = l1*cos(a1-a2), l2
    m = np.array([[m11, m12],[m21, m22]])


    f1 = -m2*l2*a2d*a2d*sin(a1-a2) - (m1+m2)*g*sin(a1)
    f2 = l1*a1d*a1d*sin(a1-a2) - g*sin(a2)
    f = np.array([f1, f2])
    accel = inv(m).dot(f)
```

```
    return np.array([accel[0], accel[1], a1d, a2d])


def RK4_step(y, t, dt):
    k1 = G(y,t)
    k2 = G(y+0.5*k1*dt, t+0.5*dt)
    k3 = G(y+0.5*k2*dt, t+0.5*dt)
    k4 = G(y+k3*dt, t+dt)

    return dt * (k1 + 2*k2 + 2*k3 + k4)/6


def perturb(di, df):
    for i in range(4):
        perturbed[i] = initial[i] + ((di * (perturbed[i] - initial[i
            ]))/df)
    return perturbed


def lyapunov(initial, perturbed):
    distanceInitial = np.linalg.norm(perturbed-initial)
    distanceFinal = np.linalg.norm(perturbed-initial)

    return np.log(abs(distanceFinal/distanceInitial)), initial,
        perturbed


def lyapunovTime(di, df):
    return np.log(abs(df/di))
```

```
def reset(initial):
    perturb = [None]*4


# variables
m1, m2 = 1, 1
l1, l2 = 1.0, 1.0
g = 9.81


delta_t = 0.01 # increment
totalTime = 100
time = np.arange(0, totalTime, delta_t)


# initial state
initial = np.array([0,0,pi/1.3,3*pi/4])    # [velocity, displacement]
perturbed = np.array([0,0,pi/1.3 + 0.001,3*pi/4])    # [velocity,
    displacement]
lyapunovExponents = []
distanceTimeInitial = distanceInitial = np.linalg.norm(perturbed-
    initial)
divisor = 2


# time-stepping solution
counter = 0
for t in time:
    counter = counter + 1
    distanceInitial = np.linalg.norm(perturbed-initial)
```

```
        perturbed = perturbed + RK4_step(perturbed, t, delta_t)
        initial = initial + RK4_step(initial, t, delta_t)
        distanceFinal = np.linalg.norm(perturbed-initial)

        if counter % divisor == 0:
            le = lyapunovTime(distanceTimeInitial, distanceFinal)
            lyapunovExponents.append(le)
            print(le)
            perturbed = perturb(distanceTimeInitial, distanceFinal)
            distanceInitial = distanceFinal


finalExponent = np.sum(lyapunovExponents) * (1/(totalTime))
print("The final exponent is...")
print(finalExponent)
```