University of Utah, Mathematics Department

CHAOTIC MOTION
IN
DOUBLE PENDULUM SYSTEMS

A Final Computing Project

Charles Godfrey Tannon Warnick Henry Whiting

Abstract

This project explores the chaotic dynamics of a double pendulum system, a classic example of how simple physical laws can yield complex, unpredictable behavior. Starting from the Lagrangian formulation, we derive the coupled nonlinear equations of motion governing the system. We implement the fourth-order Runge-Kutta (RK4) numerical method to simulate the system's trajectory over time. The study demonstrates the system's extreme sensitivity to initial conditions the "butterfly effect" where infinitesimal differences in starting states lead to exponentially diverging outcomes.

December 11, 2025

Contents

List of Figures

1. Introduction

A single pendulum is a classical example of simple harmonic motion. When constrained to small angles the pendulum will swing periodically and consistently. They are so predictable that some have been used to keep time. By simply adding a second pendulum at the end of the first, the system transforms into a classical example of chaotic motion. Even though these two systems are governed by the same physical laws of motion and only being acted upon by one force (gravity), a double pendulum is *heavily* dependent on initial conditions. We will use an approximation to solve this equation to view the behavior; the approximation we will use is the fourth order Runge-Kutta method.

2. Mathematical Model

To simulate the double pendulum, we must first derive the equations of motion. Because the system involves multiple degrees of freedom, the Lagrangian formulation of mechanics is significantly more efficient than the Newtonian approach.

2.1. System Setup and Geometry

We consider a system of two point masses, m_1 and m_2, connected by massless rigid rods of length l_1 and l_2. The system is confined to a 2D plane and acted upon by a uniform gravitational field g pointing downwards. The system can be visualized in Figure 2.1 on this page. Note: unlike in Figure 2.1, the mathematical (and numeric) system will not intersect with itself or other objects.

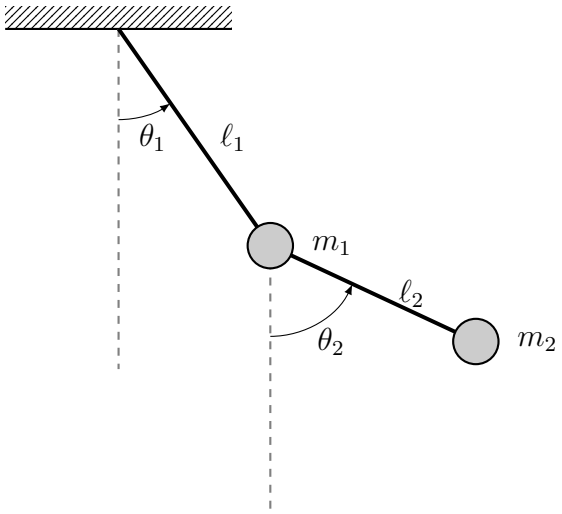

Figure 2.1: Double pendulum system

The system we will construct mathematically and depicted by Figure 2.1 is a simple pendulum, which means that there is a point mass at the end of a massless rod. A compound

pendulum can be similarly derived by just substituting the position of the point masses by the position of the centers of mass.

We define the generalized coordinates as the angles θ_1 and θ_2, measured from the vertical axis. The origin $(0, 0)$ is the pivot point of the first pendulum.

$$x_1 = \ell_1 \sin\theta_1 \tag{2.1}$$
$$y_1 = -\ell_1 \cos\theta_1 \tag{2.2}$$
$$x_2 = \ell_1 \sin\theta_1 + \ell_2 \sin\theta_2 \tag{2.3}$$
$$y_2 = -\ell_1 \cos\theta_1 + \ell_2 \cos\theta_2 \tag{2.4}$$

2.2. The Lagrangian Formulation

The Lagrangian $\mathcal{L}$ is defined as the difference between the kinetic energy (T) and the potential energy (V) of the system:

$$\mathcal{L} = T - V$$

2.2.1. Kinetic Energy (T)

The total kinetic energy is the sum of the kinetic energies of the individual masses:

$$T = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2$$

Calculating the velocities squared ($v^2 = \dot{x}^2 + \dot{y}^2$):

$$\begin{aligned} v_1^2 &= (l_1\dot{\theta}_1)^2 \\ v_2^2 &= \dot{x}_2^2 + \dot{y}_2^2 \\ &= l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1\dot{\theta}_2(\sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_2) \\ &= l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1 l_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) \end{aligned}$$

Thus, the total kinetic energy is:

$$T = \frac{1}{2}(m_1 + m_2)l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2 l_2^2\dot{\theta}_2^2 + m_2 l_1 l_2 \dot{\theta}_1\dot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{2.5}$$

2.2.2. Potential Energy (V)

Assuming potential energy is zero at $y = 0$, we have:

$$\begin{aligned} V &= m_1 g y_1 + m_2 g y_2 \\ &= -(m_1 + m_2)g l_1 \cos\theta_1 - m_2 g l_2 \cos\theta_2 \end{aligned} \tag{2.6}$$

2.3. Equations of Motion

Applying the Euler-Lagrange equation for each coordinate θ_i:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}\right) - \frac{\partial \mathcal{L}}{\partial \theta_i} = 0$$

Solving these derivatives yields a system of two coupled, non-linear second-order differential equations.

For θ_1:

$$(m_1 + m_2)l_1\ddot{\theta}_1 + m_2 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) + m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + (m_1 + m_2)g \sin\theta_1 = 0 \tag{2.7}$$

For θ_2:

$$m_2 l_2 \ddot{\theta}_2 + m_2 l_1 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - m_2 l_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + m_2 g \sin\theta_2 = 0 \tag{2.8}$$

These two equations fully describe the motion of the double pendulum. In Section 3, we will rearrange these terms to solve for $\ddot{\theta}_1$ and $\ddot{\theta}_2$ explicitly to implement the RK4 algorithm.

3. Preparing for Numerical Model

Implementing RK4 requires transforming the coupled second-order differential equations derived in Section 2 into a system of first-order differential equations.

3.1. Matrix Formulation of Motion

We begin with the coupled Euler-Lagrange equations (2.7) and (2.8). By grouping the angular acceleration terms $(\ddot{\theta}_1, \ddot{\theta}_2)$ on the left-hand side and the remaining velocity and potential terms on the right, we obtain the following linear system:

$$(m_1 + m_2)l_1\ddot{\theta}_1 + m_2 l_2 \cos(\theta_1 - \theta_2)\ddot{\theta}_2 = -m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - (m_1 + m_2)g \sin\theta_1, \tag{3.1}$$

$$m_2 l_1 \cos(\theta_1 - \theta_2)\ddot{\theta}_1 + m_2 l_2 \ddot{\theta}_2 = m_2 l_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - m_2 g \sin\theta_2. \tag{3.2}$$

This system can be written in matrix form as $M(\theta)\ddot{\theta} = b(\theta, \omega)$, where $\omega_i = \dot{\theta}_i$:

$$\underbrace{\begin{pmatrix} (m_1 + m_2)l_1 & m_2 l_2 \cos(\theta_1 - \theta_2) \\ m_2 l_1 \cos(\theta_1 - \theta_2) & m_2 l_2 \end{pmatrix}}_{M(\theta)} \underbrace{\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{pmatrix}}_{\ddot{\theta}} = \underbrace{\begin{pmatrix} -m_2 l_2 \omega_2^2 \sin(\theta_1 - \theta_2) - (m_1 + m_2)g \sin\theta_1 \\ m_2 l_1 \omega_1^2 \sin(\theta_1 - \theta_2) - m_2 g \sin\theta_2 \end{pmatrix}}_{b(\theta,\omega)}.$$

3.2. Explicit Acceleration Terms

To solve for the angular accelerations $\ddot{\theta}$, we invert the mass matrix M. For a 2×2 matrix, the inverse is explicitly given by:

$$\ddot{\theta} = M^{-1}b = \frac{1}{\det M}\begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

Defining the angle difference as $\Delta\theta = \theta_1 - \theta_2$, the determinant is $\det M = m_2 l_1 l_2 (m_1 + m_2 \sin^2 \Delta\theta)$. Carrying out the matrix multiplication yields the explicit accelerations:

$$\ddot{\theta}_1 = \frac{1}{\det M}\left[m_2 l_2 b_1 - m_2 l_2 \cos(\Delta\theta) b_2\right], \tag{3.3}$$

$$\ddot{\theta}_2 = \frac{1}{\det M}\left[-m_2 l_1 \cos(\Delta\theta) b_1 + (m_1 + m_2) l_1 b_2\right]. \tag{3.4}$$

Substituting the components of vector b (from (3.1) and (3.2)) into these expressions provides the functions necessary for the numerical solver.

3.3. State-Space Representation

Finally, we convert the two second-order equations into a system of four first-order equations. We introduce the state vector X:

$$X = \begin{pmatrix} \theta_1 & \theta_2 & \omega_1 & \omega_2 \end{pmatrix}^T.$$

The time evolution of the system is governed by $\dot{X} = F(X)$:

$$\dot{X} = \frac{d}{dt}\begin{pmatrix} \theta_1 \\ \theta_2 \\ \omega_1 \\ \omega_2 \end{pmatrix} = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \ddot{\theta}_1(\theta_1, \theta_2, \omega_1, \omega_2) \\ \ddot{\theta}_2(\theta_1, \theta_2, \omega_1, \omega_2) \end{pmatrix},$$

where $\ddot{\theta}_1$ and $\ddot{\theta}_2$ are determined by equations (3.3) and (3.4). This vector $F(X)$ is passed directly to the RK4 algorithm.

3.4. Numerical Implementation: The RK4 Algorithm

With the system reduced to the first-order form $\dot{X} = F(X)$, we can now apply the classical fourth-order Runge-Kutta (RK4) method to numerically integrate the equations of motion. This method is preferred over simpler schemes (like Euler's method) because its local truncation error is of order $O(h^4)$, providing the high precision necessary to track chaotic trajectories.

Given the state vector X_n at time t_n and a time step h, the next state X_{n+1} is computed

as a weighted average of four slope estimates:

$$X_{n+1} = X_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

where the intermediate slopes k_i are defined as:

$$\begin{aligned} k_1 &= F(X_n), \\ k_2 &= F\left(X_n + \frac{h}{2}k_1\right), \\ k_3 &= F\left(X_n + \frac{h}{2}k_2\right), \\ k_4 &= F(X_n + hk_3). \end{aligned}$$

In these equations, the function F corresponds to the system defined in Section 3.3, which calculates the derivatives $[\omega_1, \omega_2, \ddot{\theta}_1, \ddot{\theta}_2]^T$ at each step. This algorithm ensures that the non-linearities of the double pendulum are captured accurately even over extended simulation times. The full Julia code implementing this solver can be found in Appendix A on page 7.

4. Results and Discussion

Using what we have established in previous sections provides the necessary tools to simulate the double pendulum's complex dynamics. With the equations of motion derived via the Lagrangian and the Runge-Kutta solver implemented in Julia, we now turn to the simulation results.

4.1. Numerical Results

To verify the chaotic nature of the double pendulum, we simulated the system's time evolution using the RK4 integrator derived in Section 3. We specifically investigated the system's sensitivity to initial conditions.

We initialized two *almost* identical double pendulum systems (sharing identical lengths l_1, l_2 and masses m_1, m_2) with initial angles differing by a microscopic perturbation, ϵ:

$$\Theta_A = (\theta_1, \theta_2) \quad \text{and} \quad \Theta_B = (\theta_1 + \epsilon, \theta_2)$$

where $\epsilon = 10^{-5}$ degrees.

Figure 4.1 on the following page illustrates the trajectories of these two systems over time. During the initial phase of the simulation, the deviation caused by the perturbation appears negligible, and the systems move in unison. However, as the system evolves, the non-linear coupling terms in the equations of motion, specifically those derived in (3.1), cause this infinitesimal difference ϵ to propagate and grow exponentially for some time.

By the end of the simulation, the two pendulums exhibit completely uncorrelated states, occupying different regions of the configuration space despite being governed by the same deterministic laws. This confirms that while the double pendulum is deterministic (fully

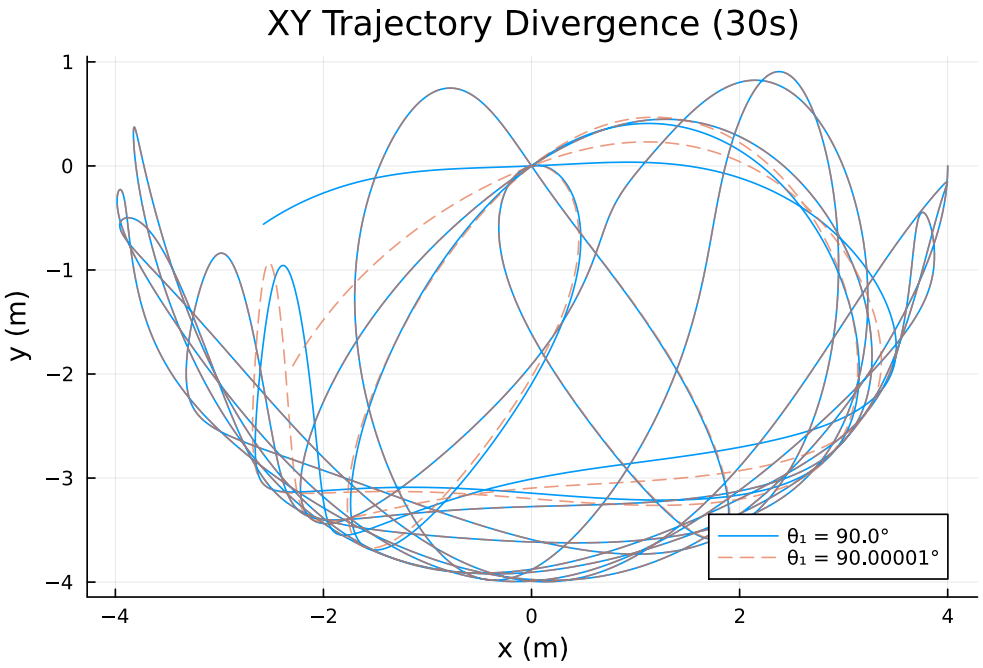

Figure 4.1: Divergence of trajectories with initial conditions differing by 10^{-5} degrees.

described by Equations 3.3 and 3.4), it is practically unpredictable over long time horizons due to this extreme sensitivity.

Plotting the angular difference $\Delta\theta$ between the two systems further illustrates this chaotic dissociation. As shown in Figure 4.2 on the next page, the error does not grow linearly but exhibits rapid, unpredictable fluctuations, the peaks of which grow exponentially, characteristic of the butterfly effect.

4.2. Conclusion

In this project, we modeled the dynamics of a double pendulum, moving from the theoretical physical setup to a full numerical simulation. By utilizing the Lagrangian formulation, we avoided the vector complexity of Newtonian mechanics and directly derived the coupled non-linear equations of motion.

We transformed these second-order differential equations into a state-space representation suitable for computational solving. The implementation of the fourth-order Runge-Kutta (RK4) method allowed us to accurately approximate the solution, maintaining numerical stability even as the system entered chaotic regimes.

The results validate that simple physical systems can exhibit complex, non-linear behavior. The visualization of the trajectory divergence serves as a concrete demonstration of deterministic chaos, satisfying the primary objective of this study. Future work could involve solving triple or n-pivot pendulums, This would require much higher dimension vector fields to incorporate into the Runge-Kutta methods.

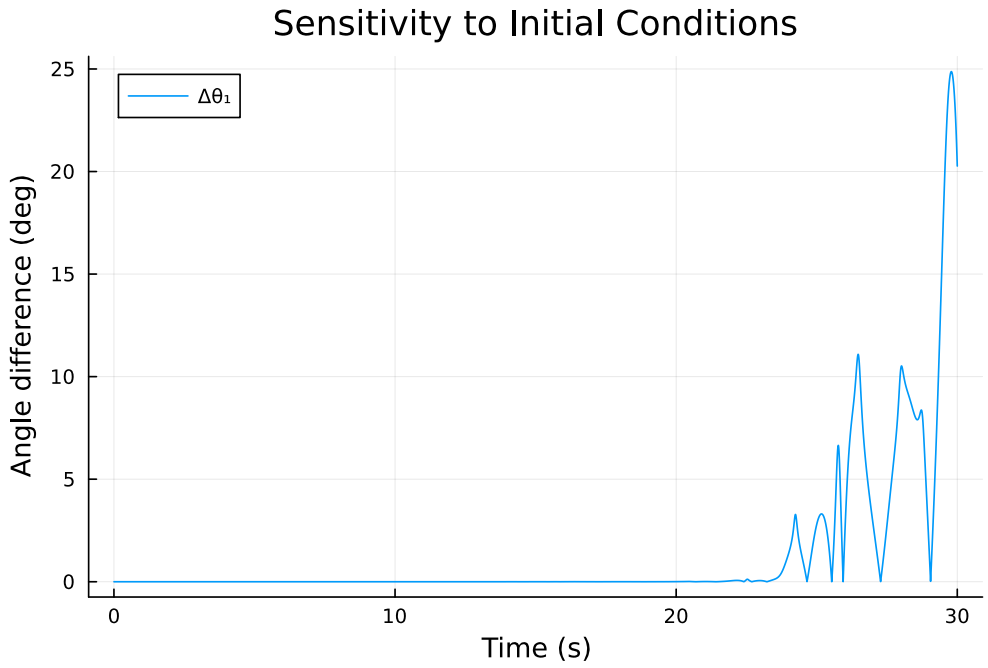

Figure 4.2: Evolution of the angular difference $\Delta\theta$ between the two systems over time.

A. Julia Code

```julia
using Plots
gr() # Set plotting backend

# parameters
G = 9.8
lengths = [2.0, 2.0]
masses = [2.0, 2.0]
time_span = (0.0, 30.0) # 30 seconds
n_steps = 15000 # RK4 steps

# ================== RK4 Solver ================== #
function rk4(IVP, n)
    a, b = IVP.tspan
    h = (b - a)/n
    t = [a + i*h for i in 0:n]

    u0 = IVP.u0
    m = length(u0)
    u = zeros(m, n+1)
    u[:,1] .= u0

    for i in 1:n
        k1 = h * IVP.f(u[:,i], IVP.p, t[i])
        k2 = h * IVP.f(u[:,i] + k1/2, IVP.p, t[i] + h/2)
```

```
        k3 = h * IVP.f(u[:,i] + k2/2, IVP.p, t[i] + h/2)
        k4 = h * IVP.f(u[:,i] + k3, IVP.p, t[i] + h)
        u[:,i+1] = u[:,i] + (k1 + 2*k2 + 2*k3 + k4)/6
    end

    return t, u
end

# Double Pendulum ODE (returns vector)
function double_pendulum(u, p, t)
    m1, m2, l1, l2, g = p
    θ1, ω1, θ2, ω2 = u

    Δ= θ2 - θ1
    denom1 = (m1 + m2)*l1 - m2*l1*cos(Δ)^2
    denom2 = (l2/l1)*denom1

    du = zeros(4)
    du[1] = ω1
    du[2] = ( m2*l1*ω1^2*sin(Δ)*cos(Δ) +
              m2*g*sin(θ2)*cos(Δ) +
              m2*l2*ω2^2*sin(Δ) -
              (m1+m2)*g*sin(θ1) ) / denom1

    du[3] = ω2
    du[4] = ( -m2*l2*ω2^2*sin(Δ)*cos(Δ) +
              (m1+m2)*g*sin(θ1)*cos(Δ) -
              (m1+m2)*l1*ω1^2*sin(Δ) -
              (m1+m2)*g*sin(θ2) ) / denom2

    return du
end

# IVP Struct
struct ODEProblem
    f::Function
    u0::Vector{Float64}
    tspan::Tuple{Float64, Float64}
    p::Tuple
end

# Simulation Function
function simulate_pendulum(θ1_deg, θ2_deg; ω1=0.0, ω2=0.0)
    u0 = deg2rad.([θ1_deg, ω1, θ2_deg, ω2])
    p = (masses[1], masses[2], lengths[1], lengths[2], G)
    IVP = ODEProblem(double_pendulum, u0, time_span, p)
    t, u = rk4(IVP, n_steps)
```

```
    return t, u
end

# Plotting Multiple Initial Conditions
initial_conditions = [
    (90.0, 90.0),
    (91.0, 90.0),
    (90.0, 91.0)
]

plot(title="Double Pendulum θvs Time", xlabel="Time (s)", ylabel="Angle (deg)")

for (θ1, θ2) in initial_conditions
    t, u = simulate_pendulum(θ1, θ2)
    plot!(t, rad2deg.(u[1,:]), label="θ start=(1,θ2)")
    plot!(t, rad2deg.(u[3,:]), label="θ start=(1,θ2)")
end

t, u1 = simulate_pendulum(90.0, 90.0)
t, u2 = simulate_pendulum(90.00001, 90.0)
x_vals = @. lengths[1] * sin(u[1,:]) + lengths[2] * sin(u[3,:])
y_vals = @. -lengths[1] * cos(u[1,:]) - lengths[2] * cos(u[3,:])

#Ploting Δθgraph
plot(t, rad2deg.(abs.(u1[1,:] .- u2[1,:])), label="Δθ", xlabel="Time (s)", ylabel
    ="Angle difference (deg)", title="Sensitivity to Initial Conditions")
#Ploting (x,y) graph
plot(x_vals, y_vals, xlabel="x (m)", ylabel="y (m)", title="Pendulum Trajectory",
     aspect_ratio=:equal, label="")
```