ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

PYTHON GROUP PROJECT
Symbolic Lagrangian Mechanics with Python
DOCUMENTATION

Student name	Id number
1. Robel Amare	ETS 1160/15
2. Robel Worku	ETS 1159/15
3. Fasil Fiseha	ETS 0532/15
4. Dagim Bogale	ETS 0355/15
5. Tamirat Bekele	ETS 1294/15
6. Nigussie Assefa	ETS 1093/15
7	

Simple Pendulum Numerical Analysis Project Documentation

Table of Contents

1. Introduction

This document provides detailed documentation for a Python script designed to perform numerical analysis of a simple pendulum. Unlike symbolic approaches that derive the equation of motion, this script focuses on calculating key physical properties and simulating the pendulum's motion over time based on user-provided parameters and initial conditions. The script utilizes the math library for standard mathematical operations and matplotlib for generating a visual representation of the system's energy over time. The primary output is delivered directly to the command line, with an optional graphical plot.

The simple pendulum is a fundamental system in classical mechanics, consisting of a point mass (bob) suspended from a fixed point by a massless, inextensible string or rod of length l. The motion is governed by the gravitational force.

2. Theoretical Background (Lagrangian Mechanics)

The equation of motion used in this numerical simulation is derived from fundamental principles of classical mechanics. One powerful method for this derivation is **Lagrangian mechanics**.

Lagrangian mechanics describes the dynamics of a system using the Lagrangian (L), which is defined as the difference between the system's kinetic energy (T) and potential energy (V):

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

For a simple pendulum of mass m and length l, with the angle θ from the vertical as the generalized coordinate:

- Kinetic Energy (T): The kinetic energy of the bob is $T = 0.5mv^{2}$. The velocity of the bob is $v=l\dot{\theta}$, so $T = 1/2\, m(l\dot{\theta})^{2}$

$$T = \frac{1}{2} m l^2 \dot{\theta}^2$$

-

- Potential Energy (V): Taking the pivot as the zero potential energy level, the height of the bob is −lcos(θ). So the potential energy is V=mgh=−mglcos(θ).

  V=−mglcos(θ)

The Lagrangian is therefore:

$$L = \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos(\theta)$$

The dynamics of the system are then governed by the **Euler-Lagrange equation**:

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

Applying this equation to the Lagrangian of the simple pendulum:

1. Calculate ∂θ/∂L:

$$\frac{\partial L}{\partial \dot{\theta}} = \frac{\partial}{\partial \dot{\theta}}\left(\frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos(\theta)\right) = ml^2\dot{\theta}$$

2. Calculate d/dt (∂θ/∂L):

$$\frac{d}{dt}\left(ml^2\dot{\theta}\right) = ml^2\ddot{\theta}$$

3. Calculate ∂L/∂θ:

$$\frac{\partial L}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos(\theta)\right) = -mgl\sin(\theta)$$

Substitute these results into the Euler-Lagrange equation:

$$ml^2\ddot{\theta} - (-mgl\sin(\theta)) = 0$$

$$ml^2\ddot{\theta} + mgl\sin(\theta) = 0$$

Assuming m≠0 and L≠0, we can divide by ml^2 to get the equation of motion:

$$\ddot{\theta} + \frac{g}{l}\sin(\theta) = 0$$

$$\ddot{\theta} = -\frac{g}{l}\sin(\theta)$$

This is the fundamental equation that the numerical simulation in this script solves to determine the pendulum's motion over time.

3. Methodology

The script employs a numerical approach to analyze the pendulum's dynamics.

Numerical Approach

The core of the simulation relies on the known equation of motion for a simple pendulum, derived from Newtonian or Lagrangian mechanics:

θ¨=−Lgsin(θ)

Where:

- θ is the angular displacement from the vertical.
- θ¨ is the angular acceleration.
- g is the acceleration due to gravity.
- L is the length of the pendulum.

Since this is a second-order ordinary differential equation, we use a numerical integration method to approximate the values of θ and the angular velocity θ˙ over time. The **Euler-Cromer method** is used, which is a slight modification of the basic Euler method that offers better stability for oscillatory systems like the pendulum. The update rules for angle and angular velocity at each small time step Δt are:

$$\dot{\theta}current + \ddot{\theta}_{current}\Delta t$$

$$\theta_{new} = \theta_{current} + \dot{\theta}_{new}\Delta t$$

The angular acceleration θ¨current is calculated at the current angle θcurrent using the equation of motion.

Calculations Performed

The script performs the following calculations:

1. Small Angle Period: Calculates the period of oscillation using the small angle approximation formula:

$$T \approx 2\pi\sqrt{\frac{l}{g}}$$

This provides a quick estimate of the oscillation period, valid for small initial angles.

2. **Numerical Simulation:** Simulates the pendulum's motion over a specified duration using the Euler-Cromer method. At each time step, it calculates:
   - Time (t)
   - Angular Position (θ)
   - Angular Velocity (θ˙)
   - Kinetic Energy (T)
   - Potential Energy (V, taking the pivot as the zero potential energy level)
   - Total Mechanical Energy (E=T+V)

User Inputs

The script requires the user to provide the following numerical inputs via the command line when prompted:

- **Mass of the pendulum bob (**m**):** In kilograms (kg).
- **Length of the pendulum (**l**):** In meters (m).
- **Acceleration due to gravity (**g**):** In meters per second squared (m/s^2).
- **Initial angular position (**θ0**):** In degrees. This value is converted to radians internally for calculations.
- **Initial angular velocity (**θ˙0**):** In radians per second (rad/s).
- **Simulation duration:** The total time for which the simulation should run, in seconds (s).
- **Time step (**Δt**):** The size of the time interval used in the numerical integration, in seconds (s). A smaller time step generally leads to more accurate results but increases computation time.

Input validation is included to ensure that mass, length, gravity, duration, and time step are positive values.

4. Results

The script presents the results of the analysis and simulation in two ways: command-line output and an optional energy plot.

Command-Line Output

After the user provides the inputs, the script prints the following to the console:

1. **Analysis Results:** Displays the calculated period of the pendulum using the small angle approximation.

2. **Simulation Results:** Presents the data from the numerical simulation in a formatted table. The table includes columns for Time, Angle, Angular Velocity, Kinetic Energy, Potential Energy, and Total Energy for each simulated time step. For very long simulations, the output is truncated to a maximum number of rows to avoid overwhelming the console.

Energy Conservation Description

A brief explanation of energy conservation in the context of the simple pendulum is printed to the console after the simulation results table. This helps the user interpret the energy values in the table and the subsequent plot.

Energy Plot

Optionally, after the command-line output, the user is prompted if they want to see the energy plot. If the user enters 'yes', a matplotlib window will appear displaying a plot of:

- Kinetic Energy vs. Time
- Potential Energy vs. Time
- Total Energy vs. Time

This plot visually demonstrates the exchange between kinetic and potential energy and how the total energy ideally remains constant over the simulation duration (any drift in total energy indicates numerical error from the simulation method).

5. How to Run the Script

Follow these steps to run the simple pendulum numerical analysis script:

Prerequisites

1. **Python:** Ensure you have Python installed on your system.
2. **Matplotlib:** You need the matplotlib library to generate the energy plot. Install it using pip:

   ```
   pip install matplotlib
   ```

Steps to Run

1. **Save the Code:** Copy the entire Python code from the immersive document titled "Simple Pendulum Numerical Analysis Script (with Energy Plotting)" and paste it into a plain text file. Save this file with a **.py** extension
2. **Open Terminal/Command Prompt:** Open your system's terminal or command prompt application.
3. **Navigate to Directory:** Use the cd command to navigate to the directory where you saved the pendulum_analysis.py file. For example, if you saved it in a folder named PhysicsScripts on your Desktop, you might type:
   cd Desktop/PhysicsScripts

4. **Run the Script**

Providing Input

Once you run the script, it will prompt you to enter numerical values for the pendulum parameters and simulation settings one by one in the terminal. Type each value and press Enter.

Note: If you encounter an EOFError or the script finishes without prompting for input, it means your execution environment does not support interactive input. You will need to run the script in a standard local terminal.

Viewing Output

After you provide all inputs, the script will print the calculated period and the simulation results table directly in your terminal window. Scroll up to see all the command-line output.

Interacting with the Plot

After the command-line output, the script will ask if you want to see the energy plot:

```
Show energy plot? (yes/no):
```

Type yes and press Enter to open the matplotlib plot window. Type no and press Enter to exit the script without showing the plot.

If you choose 'yes', a new window titled "Simple Pendulum Energy over Time" will appear, displaying the energy curves. You can typically use tools within the plot window to zoom, pan, and save the plot. Close the plot window to end the script.

6. References

- Taylor, J. R. (2005). *Classical Mechanics*. University Science Books.
- Matplotlib Documentation. (n.d.). Retrieved from https://matplotlib.org/stable/contents.html (Official documentation for the matplotlib plotting library).