TECHNICAL REPORT: DYNAMIC PENDULUM SIMULATOR WITH REGRESSION ANALYSIS

BY SHAFIQ R

DISEDIAKAN OLEH,

DISEMAK OLEH,

DISAHKAN OLEH,

Contents

Executive Summary / Abstract

This report presents a web-based pendulum experiment simulator that demonstrates the fundamental relationship between a pendulum's length and its oscillation period through interactive data visualization and analysis. The application generates synthetic experimental data with adjustable parameters, simulating real-world conditions by incorporating controlled noise into the measurements. Using JavaScript and Chart.js, it automatically performs linear regression on the period-squared versus length data, calculating both the best-fit line and associated uncertainties while deriving the experimental value of gravitational acceleration with proper error propagation. The intuitive interface allows users to repeatedly generate and analyse datasets, observing how random variations affect results and gaining practical insights into data interpretation and experimental physics. Designed as an educational tool, the simulator effectively bridges theoretical concepts with hands-on data analysis, providing a virtual laboratory experience that reinforces key principles of simple harmonic motion, linearization techniques, and uncertainty quantification. The modular architecture supports future enhancements such as real-world data integration, advanced statistical features, and expanded physics models, making it adaptable for both classroom instruction and independent learning. By combining clear visualizations with immediate analytical feedback, this tool offers an engaging platform for students and educators to explore pendulum dynamics while developing essential data literacy skills.

Introduction

In physics education, practical experiments are essential for reinforcing theoretical concepts and promoting scientific inquiry. One fundamental experiment in mechanics involves studying the relationship between the period of a simple pendulum and its length to derive the acceleration due to gravity. However, conducting such experiments in physical labs may be constrained by limited time, equipment, or environmental consistency.

This web application was developed to simulate the pendulum experiment digitally, enabling students to generate realistic data, visualize results, and perform linear regression to estimate gravitational acceleration. By offering an interactive, repeatable, and accessible platform, the simulator supports inquiry-based learning and enhances students' understanding of data analysis, experimental uncertainties, and the application of mathematical models in physics.

System Overview

The Pendulum Experiment Simulator is a lightweight, browser-based web application designed to model the relationship between the square of the period (T^2) and the length (L) of a simple pendulum. It provides users (primarily students and educators) with an interactive tool to simulate experimental data, analyse relationships through graphing, and estimate the gravitational constant using linear regression.

The system operates entirely on the client side, requiring no server or backend infrastructure. It combines HTML, CSS, and JavaScript with the Chart.js library to dynamically generate data, plot graphs, and display computed results including uncertainties and derived values.

Key Features

(a) **Data Simulation**: Random generation of realistic experimental data with configurable noise to simulate natural variability.

(b) **Interactive Table & Plotting**: Real-time population of a data table and a scatter plot of T^2 vs L.

(c) **Linear Regression Analysis**: Automatic computation of the best-fit line, including slope, intercept, and their uncertainties.

(d) **Physics Application**: Estimation of the gravitational acceleration using the relationship between T^2 and L.

(e) **Responsive UI**: Clean, accessible layout for use on most desktop browsers.

This simulator bridges the gap between theory and practice in a virtual learning environment, making it especially useful in remote learning contexts or as a supplement to hands-on labs.

File Structure

The web application is organized into three primary files, each serving a distinct role in the functionality, presentation, and behaviour of the system. All files are static and run entirely on the client-side, making the application lightweight and easy to deploy.

File Name	Type	Purpose
index.html	HTML	Defines the structure of the web page, including UI elements such as the header, table, button, chart container, and script references.
style.css	CSS	Controls the visual appearance and layout of the web interface, ensuring readability and consistent styling.
script.js	JavaScript	Contains the core application logic, including data generation, chart rendering, table population, and linear regression calculations.

These three files interact cohesively to deliver a self-contained simulation experience:

- The **HTML** file lays out the elements and connects to both the stylesheet and the script.
- The **CSS** file styles the layout and ensures the app is visually accessible.
- The **JavaScript** file dynamically drives the experiment logic and visualization.

Together, these components make up a fully functional simulation tool that requires no additional dependencies aside from the Chart.js library loaded via CDN.

Technologies Used

The Pendulum Experiment Simulator is developed entirely using front-end web technologies, ensuring that it is lightweight, portable, and fully operable within any modern web browser. The structural foundation of the application is built with **HTML5**, which defines the layout of the page, including user interface elements such as the heading, data table, action button, and containers for chart output and results.

To enhance the visual presentation and ensure a clean, user-friendly interface, **CSS3** is used to style the elements consistently. This includes centring content, formatting tables for clarity, and applying spacing and typography that make the interface intuitive and accessible, especially for educational use.

The core logic of the application is powered by **vanilla JavaScript**, which is responsible for generating simulated experimental data, adding randomized noise to mimic real-world variation, and performing mathematical calculations such as linear regression. It also dynamically updates the HTML content by populating the data table, computing the regression results with

uncertainties, and estimating the experimental gravitational acceleration based on the slope of the line.

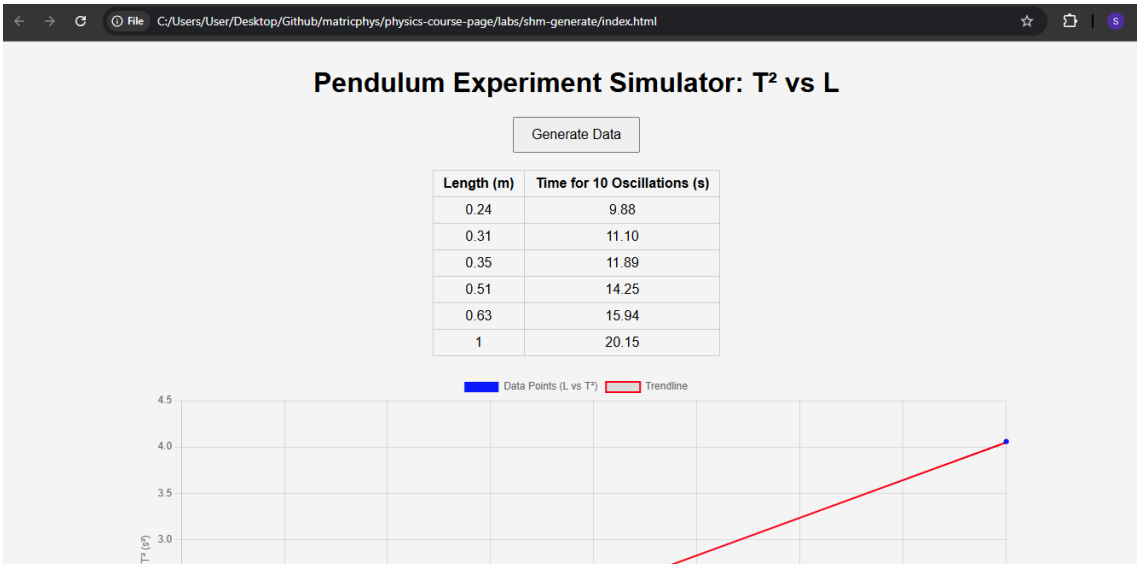
For data visualization, the app utilizes **Chart.js**, a lightweight open-source charting library that enables the rendering of a scatter plot representing T^2 (period squared) against pendulum length. This chart includes a trendline derived from the linear regression analysis, providing a clear visual correlation that supports both interpretation and learning.

Importantly, the application runs entirely in the **browser environment** with no server-side dependencies or external APIs (aside from the Chart.js CDN). This design makes the simulator easily deployable, highly responsive, and ideal for integration into both in-person and remote learning settings.

Functional Components

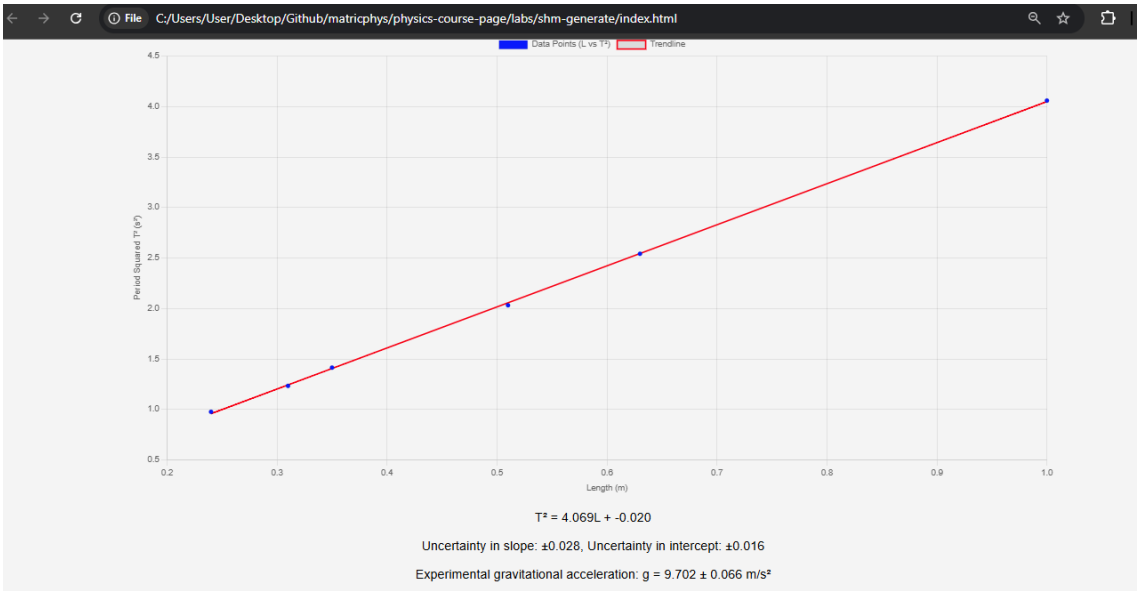
User Interface

The user interface of the Pendulum Experiment Simulator is designed to be minimalistic, intuitive, and focused on educational clarity. Upon loading the application, users are greeted with a clear and descriptive title at the top of the page, indicating the nature of the experiment: the relationship between the square of the pendulum's period (T^2) and its length (L).



Below the title, a centrally placed **"Generate Data"** button serves as the primary interaction point. This button initiates the simulated experiment and drives the application's core functionality.

Once data is generated, a structured **data table** appears, displaying each length and its corresponding time for ten oscillations. This tabular format allows users to inspect and reflect on the numerical data before engaging with visual or analytical outputs. Beneath the table, a **chart container** holds a canvas element where the relationship between T^2 and length is plotted in real-time.



In the lower section of the interface, three **result boxes** display key output values: the regression equation, the uncertainties in the slope and intercept, and the calculated gravitational acceleration with its associated uncertainty. These results are presented in a visually distinct and readable format to emphasize their importance in the analysis.

The layout ensures that all interactive and output elements are centralized and easily accessible, making the simulator well-suited for student use in both guided classroom settings and independent learning contexts.

Styling

The visual styling of the Pendulum Experiment Simulator is handled using CSS3 and is crafted to provide a clean, uncluttered interface that supports ease of use and readability. A consistent font (Arial, sans-serif) is used throughout the application to maintain clarity and a modern look. The overall layout is centred and framed with generous margins to give the content room to breathe, contributing to a focused and distraction-free experience.

background is set to a soft, neutral grey (#f4f4f4), which contrasts well with the text and graphical elements, reducing visual strain during extended use. Headings are centred and clearly distinguishable, serving as visual anchors on the page. The **"Generate Data"** button is styled with ample padding and a moderate font size to make it easily clickable and prominent without overwhelming the page.

The **data table** is styled with a collapsed border layout and alternating row spacing, ensuring that numerical values are easy to read and compare. Table cells are padded and bordered for neatness, and all data is centred within the cells to maintain a structured appearance.

The **chart container** is designed to span 80% of the page width and is automatically centred, allowing the scatter plot and trendline to be prominently displayed regardless of screen size. Additionally, result sections are spaced out and centre-aligned, with a slightly larger font size to highlight the significance of the regression outputs and calculated gravitational value.

Altogether, the styling reinforces the application's pedagogical goals by ensuring that users can easily engage with both the interface and the data it presents, without distractions or usability issues.

Core Logic

The core logic of the Pendulum Experiment Simulator is implemented entirely in vanilla JavaScript and is responsible for coordinating data generation, analysis, and user interface updates. At the heart of this logic is the generateData() function, which orchestrates the entire simulation workflow upon user interaction. This function calls a series of supporting functions to generate randomized but physically meaningful data, update the HTML table with results, compute derived quantities, and render the data visually using Chart.js.

The simulator uses the well-known physics formula for the period of a simple pendulum, $T = 2\pi\sqrt{L/g}$, scaled for 10 oscillations and augmented with random noise to mimic measurement uncertainty. The generated dataset is processed immediately to calculate T^2, which becomes the basis for both charting and regression analysis.

A key part of the core logic is the **linear regression function**, which computes the slope and intercept of the best-fit line using the least squares method. This function also evaluates the statistical uncertainties in these parameters based on the residuals of the data, providing a realistic touch of error analysis found in experimental science.

Once the regression is complete, the app computes the **experimental value of gravitational acceleration (g)** from the slope using the relationship $g = \frac{4\pi^2}{m}$, and also estimates the uncertainty in g. These results are then displayed in dedicated output sections of the page.

The overall logic is modular and readable, with clearly defined functions handling specific tasks such as table updates, chart plotting, and statistical computation. This design not only simplifies maintenance and extension but also makes the code base approachable for students or educators who wish to explore or adapt the simulation.

Algorithms

The web application employs a series of algorithms to simulate and analyse pendulum motion data. The core functionality begins with the “generateData” function, which generates synthetic data for a simple pendulum experiment. First, it creates six unique pendulum lengths ranging from 0.2 to 1.0 meters using a random number generator, ensuring no duplicates. For each length, the theoretical time for 10 oscillations is calculated using the formula $T_{10} = 20\pi\sqrt{L/g}$, where g*g* is

the gravitational acceleration (9.81 m/s²). To mimic real-world variability, random noise is added to the calculated times.

The generated data is then displayed in a table and plotted on a scatter chart. The algorithm transforms the oscillation times into squared periods (T^2) to facilitate linear regression analysis. The “linearRegression” function computes the best-fit line for the T^2 vs. length data, determining the slope (m) and intercept (c) along with their uncertainties (Δm and Δc). These results are derived using statistical methods, including the calculation of residuals and standard errors.

Finally, the experimental gravitational acceleration (g_{exp}) is derived from the slope of the regression line using the relationship $g_{exp} = \frac{4\pi^2}{m}$, and its uncertainty is propagated from the slope's uncertainty.

The results, including the regression equation, uncertainties, and experimental g value, are dynamically displayed on the webpage. This systematic approach ensures accurate simulation and analysis of pendulum motion while accounting for realistic experimental noise and uncertainties.

Output & Visualization

The web application presents the pendulum experiment data and analysis results through a clear and interactive interface. Upon clicking the "Generate Data" button, the following outputs are displayed:

1. **Data Table**: A table shows the generated pendulum lengths (in meters) and their corresponding times for 10 oscillations (in seconds). This provides a structured view of the raw data used for analysis.

2. **Scatter Plot with Trendline**: A dynamic chart visualizes the relationship between pendulum length (L) and the squared period (T^2). The blue scatter points represent the experimental data, while the red trendline depicts the linear regression fit. The axes are labeled appropriately, making it easy to interpret the proportionality between L and T^2.

3. **Regression Equation**: The equation of the best-fit line ($T^2 = mL + c$) is displayed, along with the slope (m) and intercept (c), formatted to three decimal places for precision.

4. **Uncertainty Metrics**: The uncertainties in the slope ($\pm \Delta m$) and intercept ($\pm \Delta c$) are shown, providing insight into the reliability of the regression model.

5. **Experimental Gravitational Acceleration**: The derived value of g (calculated from the slope) is presented with its uncertainty, allowing users to compare it with the theoretical value (9.81 m/s²).

The visualization is powered by **Chart.js**, ensuring smooth rendering and responsiveness. The clean layout, achieved with CSS, enhances readability, making the tool suitable for both educational and analytical purposes. Users can regenerate data at will, observing how noise and randomness affect the results, reinforcing key concepts in experimental physics.

Potential Improvements

While the current web application effectively simulates and visualizes pendulum experiment data, several enhancements could further improve its functionality, accuracy, and educational value:

1. **User-Defined Parameters**:

- Allow users to customize inputs such as the number of pendulum lengths, noise amplitude, and range of lengths. This would enable more flexible experimentation and better mimic real-world lab conditions.

2. **Interactive Chart Features**:
   - Add tooltips to data points to display exact values on hover.
   - Enable zooming and panning on the chart for closer inspection of data trends.
   - Include a toggle to switch between T^2 and raw period (T) plots for comparative analysis.

3. **Advanced Statistical Analysis**:
   - Implement additional regression metrics (e.g., R-squared, chi-squared) to quantify the goodness of fit.
   - Introduce outlier detection and optional data point removal to refine the regression model.

4. **Comparative Analysis**:
   - Overlay theoretical predictions (e.g., $T^2 = \frac{4\pi^2}{g} L$) on the chart for direct comparison with experimental results.
   - Add a "confidence band" around the trendline to visually represent uncertainty in the regression.

5. **Export and Data Persistence**:

- Provide options to export data as CSV/JSON for further analysis in external tools.
- Save generated datasets locally (e.g., using localStorage) to allow users to revisit previous results.

6. **Educational Enhancements**:
   - Include a brief tutorial or tooltips explaining key concepts (e.g., uncertainty propagation, linearization).
   - Animate the pendulum motion for select lengths to reinforce the connection between theory and experiment.

7. **Error Handling and Validation**:
   - Validate user inputs (e.g., prevent negative lengths or noise values) to avoid nonsensical results.
   - Display warnings if uncertainties exceed thresholds, prompting users to adjust parameters.

8. **Mobile Optimization**:
   - Improve responsiveness for smaller screens, ensuring tables and charts remain legible on smartphones and tablets.

9. **Extended Physics Models**:
   - Incorporate corrections for non-ideal conditions (e.g., air resistance, large-angle oscillations) to explore deviations from simple harmonic motion.

These improvements would make the tool more versatile for classroom use, research, and self-guided learning, bridging the gap between idealized theory and practical experimentation.

Conclusion

This pendulum experiment simulator effectively bridges theoretical physics concepts with practical data analysis through an interactive web application. By generating synthetic pendulum motion data, performing linear regression, and visualizing the relationship between period squared (T^2) and length (L), the tool provides a hands-on way to explore the principles of simple harmonic motion and uncertainty quantification. Key features such as dynamic charting, noise-injected data, and real-time calculation of gravitational acceleration (g) enhance its educational value, making it suitable for classrooms, labs, or self-guided learning.

While the current implementation successfully demonstrates core functionalities, the proposed improvements (such as user-defined parameters, advanced statistical metrics, and export capabilities) could further elevate its utility. These enhancements would transform the simulator into a more robust virtual lab, accommodating diverse experimental scenarios and fostering deeper engagement with data analysis techniques.

Ultimately, this project highlights the power of web-based tools in modernizing physics education. By combining intuitive design with rigorous computational methods, it not only reinforces fundamental concepts but also encourages critical thinking about experimental design, error analysis, and model validation. Future iterations could expand its scope to include real-world data integration or more complex pendulum dynamics, further solidifying its role as a versatile resource for students and educators alike.

Appendix: Code Implementation

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Simple Pendulum: T² vs Length</title>
  <link rel="stylesheet" href="style.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <h1>Pendulum Experiment Simulator: T² vs L</h1>
  <button onclick="generateData()">Generate Data</button>

  <table id="dataTable">
    <thead>
      <tr><th>Length (m)</th><th>Time for 10 Oscillations (s)</th></tr>
    </thead>
    <tbody></tbody>
  </table>

  <div id="chart-container">
    <canvas id="dataChart"></canvas>
  </div>

  <div class="result" id="equation"></div>
  <div class="result" id="uncertainties"></div>
  <div class="result" id="gravity"></div>

  <script src="script.js"></script>
</body>
</html>
```

script.js

```
const ctx = document.getElementById('dataChart').getContext('2d');
let chart;

function generateData() {
  const g = 9.81;
  const n = 6;
  const noise_amplitude = 0.1;
  const lengths = new Set();

  while (lengths.size < n) {
    let L = +(Math.random() * 0.8 + 0.2).toFixed(2);
    lengths.add(L);
  }

  const sortedLengths = Array.from(lengths).sort((a, b) => a - b);
  const times10 = sortedLengths.map(L => {
    const T10 = 20 * Math.PI * Math.sqrt(L / g);
    const noise = (Math.random() * 2 - 1) * noise_amplitude;
```

```
    return T10 + noise;
  });

  updateTable(sortedLengths, times10);
  plotData(sortedLengths, times10);

  const T_squared = times10.map(T10 => (T10 / 10) ** 2);
  const { m, c, dm, dc } = linearRegression(sortedLengths, T_squared);

  document.getElementById('equation').textContent = `T² = ${m.toFixed(3)}L + ${c.toFixed(3)}`;
  document.getElementById('uncertainties').textContent = `Uncertainty in slope: ±${dm.toFixed(3)}, Uncertainty
in intercept: ±${dc.toFixed(3)}`;

  const g_exp = (4 * Math.PI ** 2) / m;
  const dg = (4 * Math.PI ** 2 / (m ** 2)) * dm;
  document.getElementById('gravity').textContent = `Experimental gravitational acceleration: g =
${g_exp.toFixed(3)} ± ${dg.toFixed(3)} m/s²`;
}

function updateTable(lengths, times10) {
  const tbody = document.querySelector('#dataTable tbody');
  tbody.innerHTML = "";
  for (let i = 0; i < lengths.length; i++) {
    const row = `<tr><td>${lengths[i]}</td><td>${times10[i].toFixed(2)}</td></tr>`;
    tbody.innerHTML += row;
  }
}

function plotData(lengths, times10) {
  const T_squared = times10.map(T10 => (T10 / 10) ** 2);
  const dataPoints = lengths.map((L, i) => ({ x: L, y: T_squared[i] }));
  const { m, c } = linearRegression(lengths, T_squared);
  const trendline = [
    { x: Math.min(...lengths), y: m * Math.min(...lengths) + c },
    { x: Math.max(...lengths), y: m * Math.max(...lengths) + c }
  ];

  if (chart) chart.destroy();

  chart = new Chart(ctx, {
    type: 'scatter',
    data: {
      datasets: [
        {
          label: 'Data Points (L vs T²)',
          data: dataPoints,
          backgroundColor: 'blue'
        },
        {
          label: 'Trendline',
          data: trendline,
          type: 'line',
          fill: false,
          borderColor: 'red',
          borderWidth: 2,
          pointRadius: 0
```

```
        }
      ]
    },
    options: {
      scales: {
        x: {
          title: {
            display: true,
            text: 'Length (m)'
          }
        },
        y: {
          title: {
            display: true,
            text: 'Period Squared T² (s²)'
          }
        }
      }
    }
  });
}

function linearRegression(x, y) {
  const n = x.length;
  const sum_x = x.reduce((a, b) => a + b, 0);
  const sum_y = y.reduce((a, b) => a + b, 0);
  const x_bar = sum_x / n;
  const y_bar = sum_y / n;

  let Sxx = 0, Sxy = 0;
  for (let i = 0; i < n; i++) {
    Sxx += (x[i] - x_bar) ** 2;
    Sxy += (x[i] - x_bar) * (y[i] - y_bar);
  }

  const m = Sxy / Sxx;
  const c = y_bar - m * x_bar;

  let residuals = 0;
  for (let i = 0; i < n; i++) {
    const y_fit = m * x[i] + c;
    residuals += (y[i] - y_fit) ** 2;
  }

  const sigma2 = residuals / (n - 2);
  const dm = Math.sqrt(sigma2 / Sxx);
  const dc = Math.sqrt(sigma2 * (1 / n + x_bar ** 2 / Sxx));

  return { m, c, dm, dc };
}
```

style.css

```
body {
  font-family: Arial, sans-serif;
  margin: 30px;
```

```
  background: #f4f4f4;
}

h1 {
  text-align: centre;
}

#chart-container {
  width: 80%;
  margin: 0 auto;
}

table {
  margin: 20px auto;
  border-collapse: collapse;
}

th, td {
  padding: 6px 12px;
  border: 1px solid #ccc;
  text-align: centre;
}

.result {
  text-align: centre;
  margin-top: 20px;
  font-size: 18px;
}

button {
  display: block;
  margin: 20px auto;
  padding: 10px 20px;
  font-size: 16px;
}
```

This page has been intentional left blank.