Digital Twin of an Inverted Pendulum System (Lab3)

Soheil Behnam Mina Ghaderi Amir Jafari

1 Introduction

In this lab, the goal is to **optimize the pendulum model** developed in **Lab 1** by adjusting its parameters to minimize the difference between the **observed data** and the **predicted behavior** from the simulation. This optimization process is crucial for making sure that the simulated pendulum behaves accurately and realistically, reflecting the observed data as closely as possible.

1.1 Data Preparation

Before proceeding with the optimization, we filtered the recorded data using a **Kalman filter**, which allowed us to get a clean version of the sensor data, including:

- **Time** (time steps)
- **Theta** (angle of the pendulum)
- **Theta dot** (angular velocity)
- **X** (position of the cart)

The pendulum's angle (θ) is **calibrated** based on the **offset** at the end of the recorded data, where the pendulum is assumed to be stationary. This ensures that the initial conditions are accurately set before optimization begins. Additionally, θ is calculated using the formula $\arctan2(x_{\text{accel}}, z_{\text{accel}})$, derived from accelerometer data. The **position** (x) is calculated based on the measurements obtained from the code and our data.

1.2 Passive Mode Assumption

The optimization in this lab and the following one focuses solely on the **passive mode** of the pendulum, where we **do not consider motor effects**. The assumption is that the motor does not influence the pendulum's motion, allowing us to simplify the model. In **Lab 1**, the formula used to compute $\ddot{\theta}$ (angular acceleration) included torque contributions from gravity, damping, and motor torque.

Here's the equation used for θ double dot in the motor-influenced model:

```
def get_theta_double_dot(self, theta, theta_dot):
    torque_gravity = -(self.g / (self.I_scale * self.l)) * np.sin(theta)
    torque_damping = -(self.c_c / self.I) * theta_dot
    xdoubledot = self.a_m * self.R_pulley * self.currentmotor_acceleration
    torque_motor = - (1 / (self.I_scale * self.l)) * xdoubledot * np.cos(theta)

    return torque_gravity + torque_damping + torque_motor
```

1.3 In Passive Mode

In passive mode, we assume that the **motor torque is zero**, so the $\ddot{x}$ term is eliminated, leaving only the **gravity torque** and **damping torque**:

```
def get_theta_double_dot(self, theta, theta_dot):
    # Torque due to gravity (restoring force)
    torque_gravity = -(self.g / (self.I_scale * self.l)) * np.sin(theta)

    # Damping term - mass naturally cancels out
    torque_damping = -(self.c_c / self.I) * theta_dot

    # Since we are in passive mode, the motor torque is assumed to be zero
    return torque_gravity + torque_damping
```

1.4 Simplification of Parameters

To reduce the number of parameters, we make some simplifications:

- We assume that the **moment of inertia (I)** is proportional to the mass and length of the pendulum:
$$I = \text{I\_scale} \times m \times l^2$$
- The **friction coefficients**, including **Coulomb friction** and **air friction**, are modeled as functions of **velocity**. These are combined into a single **damping coefficient**, effectively reducing the number of unknown parameters.

The final model that we focus on during optimization has three unknown parameters:

- **Mass (m)**
- **Moment of inertia scale (I_scale)**
- **Damping coefficient (c_damping)**

These parameters are estimated during the optimization process to minimize the **cost function**, which evaluates the model's performance by comparing the simulated behavior to the real-world data.

1.5 Implementation and Report

The implementation of the optimization is carried out in the code `optimize_pendulum_grid_search.py`, and the full report for this lab can be found in the **reports folder** of our repository.

2 Approach and Methodology

2.1 Grid Search

Grid search is a straightforward and systematic method for exploring the parameter space. The goal of the grid search is to find a **region of interest** in the parameter space where the model's performance is optimal or near-optimal. This is done by testing different combinations of the following parameters:

- **Moment of Inertia Scale (I_scale)**
- **Damping Coefficient (c_damping)**
- **Mass (m)**

2.1.1 Why Grid Search?

Grid search is chosen because it ensures that all combinations of parameters within the specified ranges are tested, providing a comprehensive search of the parameter space. This method is particularly useful when the parameter space is small, and the computational cost of testing all combinations is manageable.

2.1.2 Steps in Grid Search:

1. **Define Parameter Ranges:** The ranges for the parameters were chosen based on realistic values for a pendulum system:
   - **Moment of Inertia Scale (I_scale)**: The range is from 0.6000 to 0.9000.
   - **Damping Coefficient (c_damping)**: The range for the damping coefficient is set from 0.001 to 0.040.
   - **Mass (m)**: A mass range from 0.5000 to 1.5000 is chosen.
2. **Define Resolution:** We test **15 values** for each of the three parameters, resulting in $15^3 = 3375$ total combinations.
3. **Simulation and Evaluation:** For each combination of parameters, we run a simulation and compute the error between the predicted behavior and the observed data. The error is calculated using a **cost function** (which we will discuss next).

2.2 Cost Function

The **cost function** is a critical element of the optimization process. It measures how good a set of parameters is by comparing the model's output to the observed data. The cost function used in this method includes several error components, each quantifying different aspects of the model's performance. These components are weighted according to their importance in evaluating the simulation's fidelity to real data.

2.2.1 Time Domain Error

The time domain error compares the real and simulated pendulum angles over time. It emphasizes errors that occur at later times by applying an exponential weighting to the differences. The formula for the time domain error is:

$$\text{Time Domain Error} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\theta_{\text{sim}}(t_i) - \theta_{\text{real}}(t_i)}{\max |\theta_{\text{real}}|} \right)^2 \cdot e^{\alpha t_i}$$

Where:

- $\theta_{\text{sim}}(t_i)$ is the simulated angle at time t_i,

- $\theta_{\text{real}}(t_i)$ is the real angle at time t_i,
- N is the number of time steps considered,
- α is the exponential weighting factor, which allows us to give more importance to recent data.

This error term is weighted by a factor of 50 in the overall cost function.

2.2.2 Frequency Domain Error

The frequency domain error compares the frequency content of the real and simulated pendulum motions. The error is calculated using the Fast Fourier Transform (FFT) of both the real and simulated angle data. The formula for the frequency domain error is:

$$\text{Frequency Domain Error} = \frac{1}{N_{\text{freq}}} \sum_{i=1}^{N_{\text{freq}}} \left(\frac{|\mathcal{F}(\theta_{\text{sim}})(f_i)|}{\max |\mathcal{F}(\theta_{\text{sim}})(f_i)|} - \frac{|\mathcal{F}(\theta_{\text{real}})(f_i)|}{\max |\mathcal{F}(\theta_{\text{real}})(f_i)|} \right)^2$$

Where:

- $\mathcal{F}(\theta)$ denotes the Fourier transform of the pendulum angle,
- f_i represents the frequency bins,
- N_{freq} is the number of frequency bins used for comparison.

The frequency domain error is weighted by a factor of 600 in the total cost function.

2.2.3 Amplitude Error

The amplitude error compares the peak amplitudes of the real and simulated pendulum oscillations. The formula for the amplitude error is:

$$\text{Amplitude Error} = \frac{1}{M} \sum_{j=1}^{M} \left(\frac{|A_{\text{sim}}(t_j)|}{\max |A_{\text{real}}|} - \frac{|A_{\text{real}}(t_j)|}{\max |A_{\text{real}}|} \right)^2$$

Where:

- $A_{\text{sim}}(t_j)$ is the amplitude of the simulated pendulum at time t_j,
- $A_{\text{real}}(t_j)$ is the amplitude of the real pendulum at time t_j,
- M is the number of peak values considered.

This error is weighted by a factor of 200 in the total cost function.

2.2.4 Decay Error

The decay error compares the rate at which the oscillation amplitude decays over time for the real and simulated pendulum. The formula for the decay error is:

$$\text{Decay Error} = \frac{1}{P}\sum_{k=1}^{P}\left(\frac{A_{\text{sim}}(k+1)/A_{\text{sim}}(k)}{A_{\text{real}}(k+1)/A_{\text{real}}(k)} - 1\right)^2$$

Where:

- $A_{\text{sim}}(k)$ and $A_{\text{real}}(k)$ are the amplitudes at the k-th and $(k+1)$-th peak,
- P is the number of peak-to-peak comparisons made.

The decay error is weighted by a factor of 200 in the total cost function.

2.2.5 Total Cost Function

The total cost function is a weighted sum of all the error components described above:

$$\text{Total Cost} = 50\cdot\text{Time Domain Error} + 600\cdot\text{Frequency Domain Error} + 200\cdot\text{Amplitude Error} + 200\cdot\text{Decay Error}$$

Each component contributes to the overall error, and the weights are chosen to reflect the relative importance of each term in the optimization process. By minimizing the total cost, we ensure that the simulated pendulum behavior matches the real data as closely as possible.

2.3 Model Enhancements

Although the current model provided reasonable results, there is room for improvement in capturing the **real-world dynamics** of the pendulum. One significant limitation is the assumption that the pendulum's motion is purely passive and that the only forces involved are gravity and damping. If the error in the simulation remains high, we could consider introducing additional parameters to better capture the complexities of the system.

Friction is one such factor that could be modeled more accurately. In real-world systems, **static friction** and **kinetic friction** affect the motion, especially during slow oscillations or when the pendulum comes to a near stop. Static friction, in particular, is more noticeable at low velocities. To model this, we introduced a friction term that enhances the damping force when the pendulum's velocity is near zero.

The effect of static friction is modeled by applying a smooth enhancement near zero velocity. Specifically, the *enhanced friction* is introduced when the velocity is below a certain threshold, which corresponds to the **static threshold**. The threshold for enhanced friction was set to a value of 0.5, and the damping force increases smoothly as the pendulum's velocity approaches zero. This approach provides a more realistic representation of the pendulum's behavior when it slows down significantly.

The equation used for the enhancement is:

$$\text{enhancement} = 5.0 \times \left(1.0 - \left(\tanh\left(\frac{|\theta_{\text{dot}}|}{\text{static threshold}}\right)\right)^2\right)$$

This frictional term ensures that the model better captures the real-world behavior of the pendulum, particularly when it experiences resistive forces at low speeds.

This frictional term ensures that the model better captures the real-world behavior of the pendulum, particularly when it experiences resistive forces at low speeds.

Additionally, **external forces**, such as environmental forces (wind resistance, air pressure, or vibrations from surrounding objects), could also be introduced. While these are not typically dominant in simple pendulum systems, their effects may become more pronounced in specific contexts, such as large-scale pendulums or pendulums operating in non-ideal environments (e.g., in a vacuum, where air resistance is negligible).

2.4 Local Optimization (Nelder-Mead)

Once we identify a promising region of the parameter space using grid search, we perform **local optimization** to fine-tune the parameters for a more precise solution. We use the **Nelder-Mead method**, which is a gradient-free optimization technique suitable for non-differentiable functions. This method is ideal in cases where the cost function may be complex, as it does not require the derivatives or gradients of the objective function.

2.4.1 Why Nelder-Mead?

The Nelder-Mead method is chosen because: - It does **not require gradients** or derivatives of the cost function, making it well-suited for this case, where the cost function is often non-differentiable or computationally expensive to compute gradients for. - The method is **efficient for high-dimensional optimization problems** and tends to **converge quickly** to a local minimum, even in complex parameter spaces with multiple minima.

2.4.2 How It Works:

The Nelder-Mead method follows an iterative approach that refines the parameters to minimize the cost function. The key steps in the algorithm are:

- Initialization : The algorithm starts with an initial set of parameter values. In our case, these values are the best parameters found from the grid search. - Reflection : The method first reflects a set of points around a centroid to explore regions that might have lower cost. If a reflection results in a better solution, this reflection point is used. - Expansion : If reflection provides a better result, the method expands the search in that direction to try to improve the solution further. - Contraction : If the new point does not improve upon the current best point, the method contracts the search space, moving the points toward better solutions. - Termination : The optimization stops when the difference between the parameters in successive iterations is below a set tolerance or after a predefined number of iterations.

The Nelder-Mead method works by iteratively adjusting the parameters in search of the optimal values that minimize the total cost function. The method can be applied by updating the following formula in the cost function:

$$\text{Cost Function} = \alpha \cdot E_{\text{time}} + \beta \cdot E_{\text{frequency}} + \gamma \cdot E_{\text{amplitude}} + \delta \cdot E_{\text{decay}} + \varepsilon \cdot E_{\text{energy}}$$

Where the parameters $\alpha, \beta, \gamma, \delta, \varepsilon$ are adjusted using the Nelder-Mead method to find the optimal balance.

2.5 Why Use Both Grid Search and Local Optimization?

Using a combination of **grid search** and **local optimization** provides several advantages in the parameter tuning process: - Grid search ensures a **broad exploration of the parameter space**,

helping us find a **good initial region** to start the optimization. The grid search covers the full parameter space in a systematic way, ensuring that we do not overlook potentially good solutions within the parameter range. - Local optimization then **fine-tunes the parameters** within this region, ensuring that we obtain the best possible results with minimal computational cost. Local optimization focuses on refining the solution found by the grid search, allowing us to achieve a more accurate and efficient parameter set while avoiding unnecessary computations in less promising areas of the parameter space.

2.5.1 Summary:

In this methodology, we leverage both grid search and Nelder-Mead local optimization techniques to ensure thorough exploration and precise parameter refinement. Grid search provides a comprehensive scan of the parameter space, while Nelder-Mead optimizes the parameters for a more precise and computationally efficient solution. Together, these techniques offer a robust framework for optimizing the pendulum model's parameters.

3 Results and Analysis

3.1 Optimization Parameters

After completing both grid search and local optimization, the optimal parameters for the pendulum model were found to be:

- **Mass:** 0.5786 kg
- **Damping Coefficient:** 0.00878749
- **Moment of Inertia (I_scale):** 0.7140

These parameters were obtained through an iterative optimization process. The grid search helped identify a promising region in the parameter space, and the local optimization (using the Nelder-Mead method) refined these values for the most accurate solution.

3.2 Performance Metrics

The performance metrics provide a quantitative measure of how well the optimized model matches the observed pendulum motion. The key metrics are summarized in Table 1:

Table 1: Performance Metrics of the Optimized Model

Metric	**Value**
RMS Error (rad)	0.0437
Max Absolute Error (rad)	0.1671
Frequency Error (%)	22.26
Damping Ratio	0.0037

3.3 Sensitivity Analysis

Sensitivity analysis was performed to understand how changing each parameter affects the total cost. The analysis showed that:

- I_scale has the most significant impact on the total cost. A 20% change in I_scale resulted in a substantial increase in cost, indicating that moment of inertia is crucial for pendulum behavior.
- Mass impacts the cost function significantly, though to a lesser degree than I_scale. Variations in mass outside the optimal range caused noticeable increases in cost.
- Damping also played a significant role. A 10% change in damping led to a noticeable increase in cost, underlining the importance of damping in modeling the pendulum's motion accurately.

3.4 Plots Analysis

Several plots were generated to visualize the optimization results. These plots allow us to compare the real data with the simulated pendulum behavior.

3.4.1 Optimization Sensitivity Plots

The sensitivity analysis is illustrated in the following plots:

- I_scale Sensitivity : The plot shows how variations in the moment of inertia (I_scale) affect the cost function. A 10% variation leads to a significant change in cost, indicating the importance of this parameter.
- Mass Sensitivity : The plot highlights how changes in mass affect the cost. While less sensitive than I_scale, variations in mass still result in higher costs when moved outside the optimal range.
- Damping Sensitivity : This plot illustrates the sensitivity of the cost to changes in damping. A 10% change in damping results in noticeable changes in cost.

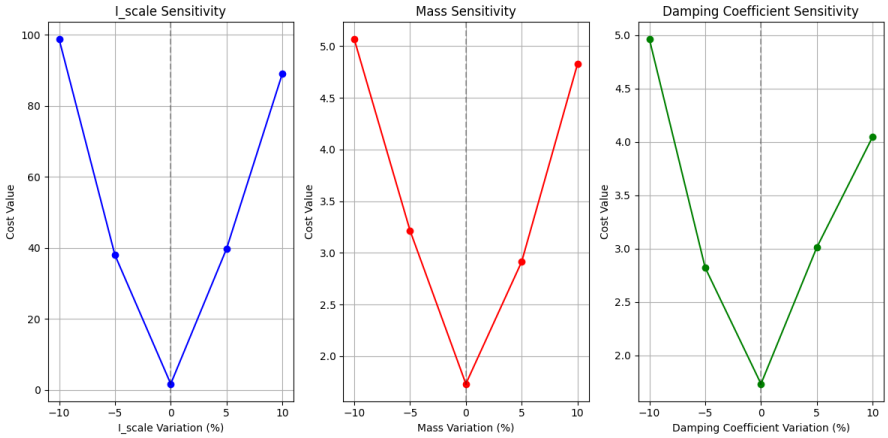

Figure 1: Sensitivity Analysis Plot showing the effect of parameter variations (I_scale, mass, and damping) on the cost.

3.4.2 Pendulum Simulation Plots

The following five subplots show the comparison between the simulated and real data for various aspects of the pendulum motion:

1. Pendulum Angle vs. Time: This plot compares the angular displacement over time for the simulated pendulum and the real data. A good fit between the two curves indicates that the model successfully captures the pendulum's motion.

2. Angular Velocity Comparison: This plot compares the angular velocity of the real and simulated pendulums. The simulation captures the general trend of the observed angular velocity, although some deviations are present.

3. Phase Space Plot: The phase space plot shows the relationship between the pendulum's angle and angular velocity. Both the real and simulated data trace similar trajectories, indicating that the model's dynamics are in line with the observed data.

4. Zero Crossings and Peaks: This plot identifies the zero crossings and peaks of both the real and simulated pendulum motions. While the zero crossings and peaks align well, minor differences in peak timing suggest room for improvement in capturing the oscillation frequency.

5. Frequency Domain Analysis: The frequency domain analysis compares the frequency spectra of the real and simulated data. The mismatch at higher frequencies highlights the need for further refinement, especially with respect to the oscillation frequency.

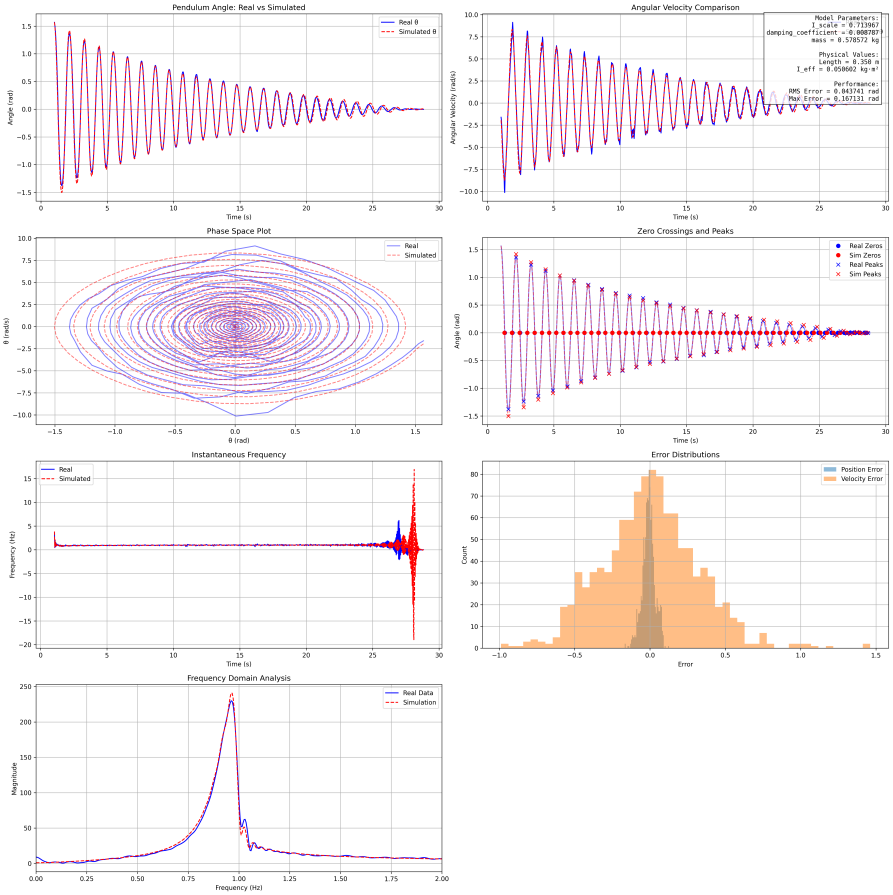

Figure 2: Pendulum Simulation Analysis showing the comparison between real and simulated pendulum behavior, including angular displacement, angular velocity, and frequency analysis.

These plots provide a detailed visualization of how well the optimized model replicates the observed pendulum motion. Despite a reasonable fit, there is room for improvement in the frequency response, as indicated by the frequency error of 22.26%.

In the next section, we will discuss potential improvements to the model and outline directions for future work.

4 Discussion and Future Improvements

4.1 Optimization Efficiency

The optimization process, including the grid search and local optimization steps, provided useful insights into the parameter space. However, grid search can be computationally expensive, especially when dealing with large parameter ranges or more complex systems. Grid search evaluates each possible combination of parameters, which can become impractical when the number of parameters or their possible values increases.

Alternative approaches, such as **Bayesian optimization**, could significantly improve efficiency. Bayesian optimization models the cost function using probabilistic models and selects new parameter values to evaluate based on past evaluations. This method is much more efficient because it narrows down the search space in fewer iterations while maintaining a high probability of finding optimal parameters.

Another potential improvement is the use of **Genetic Algorithms (GA)**, which simulate natural selection processes to evolve a population of solutions over several generations. GAs are particularly useful in complex optimization tasks with a large parameter space, and they can converge faster to a near-optimal solution by mimicking genetic evolution.

4.2 Model Enhancements

Although the current model provided reasonable results, there is room for improvement in capturing the **real-world dynamics** of the pendulum. One significant limitation is the assumption that the pendulum's motion is purely passive and that the only forces involved are gravity and damping. If the error in the simulation remains high, we could consider introducing additional parameters to better capture the complexities of the system.

Friction is one such factor that could be modeled more accurately. In real-world systems, **static friction** and **kinetic friction** affect the motion, especially during slow oscillations or when the pendulum comes to a near stop. Introducing frictional forces could improve the simulation's ability to replicate real-world behavior, especially in cases where the pendulum experiences a strong resistive force at low speeds.

Additionally, **external forces**, such as environmental forces (wind resistance, air pressure, or vibrations from surrounding objects), could also be introduced. While these are not typically dominant in simple pendulum systems, their effects may become more pronounced in specific contexts, such as large-scale pendulums or pendulums operating in non-ideal environments (e.g., in a vacuum, where air resistance is negligible).

4.3 Data Resolution

The resolution of the data used in the optimization process plays a crucial role in the accuracy of the model. In the current setup, the data was sampled at a relatively **low frequency**, which may limit the precision of the optimization results. By **increasing the data resolution**, i.e., using more frequent samples of the pendulum's motion, we can obtain more detailed information about its behavior and improve the model's accuracy. This finer-grained data would help capture small oscillations, subtle shifts in angular velocity, and other dynamics that may be missed at lower resolutions.

On the other hand, **downsampling** the data could be a potential way to reduce computational complexity while maintaining accuracy. By carefully selecting key data points that represent the essential characteristics of the pendulum's motion, we can decrease the number of data points and

speed up the optimization process without sacrificing the quality of the results. This trade-off between computational efficiency and model fidelity must be carefully balanced depending on the specific goals and constraints of the optimization task.

4.4 Answering Lab Questions

Based on the lab objectives outlined in **Lab3.pdf**, we can conclude the following:

- **Have the optimal parameters been successfully identified?** Yes, the grid search and local optimization led to the identification of optimal parameters, including mass, damping coefficient, and moment of inertia scale.
- **What is the effect of different parameters on the optimization?** The sensitivity analysis showed that **I_scale** had the most significant effect on the cost function, while variations in **mass** and **damping** also impacted the optimization, but to a lesser degree.
- **What other parameters could be considered for improving the model?** Additional parameters such as **friction** and **external forces** could be considered to improve the model's accuracy, especially in non-ideal or complex real-world environments.
- **What improvements can be made to the optimization process?** The current optimization process could be made more efficient by using advanced methods like **Bayesian optimization** or **Genetic Algorithms**, which are more suitable for large parameter spaces.

A Code and Report References

A.1 Code

The code used for the pendulum optimization process can be found at the following GitHub repository:

`https://github.com/Amirjf/single-rod-pendulum/blob/main/optimize_pendulum_grid_search.py`

This code implements the grid search and local optimization to find the optimal parameters for the pendulum system.

A.2 Reports

The reports related to this project, including optimization results and analysis, can be found in the reports directory of the GitHub repository:

`https://github.com/Amirjf/single-rod-pendulum/tree/main/reports`

These reports contain detailed explanations and results from the optimization process and sensitivity analysis.