Week 11 Assignments: PID Control and Algorithm Analysis

Assignment 1: PID Control for Spring-Damping System

Objective:

Develop a graphical application using MATLAB App Designer to simulate a spring-damping system controlled by a PID algorithm. The application should allow the user to set parameters for mass, damping coefficient, spring constant, and PID gains.

Requirements:

1. **Graphical Interface**:
   - Input fields for:
     - Mass m (kg)
     - Damping coefficient c ($\mathrm{N} \cdot \mathrm{s/m}$)
     - Spring constant k (N/m)
     - PID gains: kP, kI, kD
     - Desired setpoint (position)
   - Button to **run the simulation**.
   - Plot showing the position of the system over time.
   - Ability to **show the last two runs**:
     - The most recent run should be plotted with solid lines.
     - Example: The previous run can be plotted with dashed lines however, you may choose a different method if you feel it offers a better user experience.
2. **Functionality**:
   - Implement the PID control using the provided `simulatePIDControl` function.
   - Ensure the app clears previous plots and shows only the last two runs.
   - The app should handle different sets of inputs and visualize how changes affect the system response.
3. **Submission**:
   - Submit your MATLAB `.mlapp` file along with a short write-up (1-2 paragraphs) explaining
     - How your app works
     - What did you find to be the best numbers
     - Were you able to determine a pattern in the three constants regarding how they control the results

Tips:

- Experiment with various values for kP, kI, and kD to understand their effects.
- Consider edge cases, such as very high or very low damping coefficients, and explain what happens.

Example

As always, this is merely an example, you have free rein to design however you feel so long as it meets the requirements.

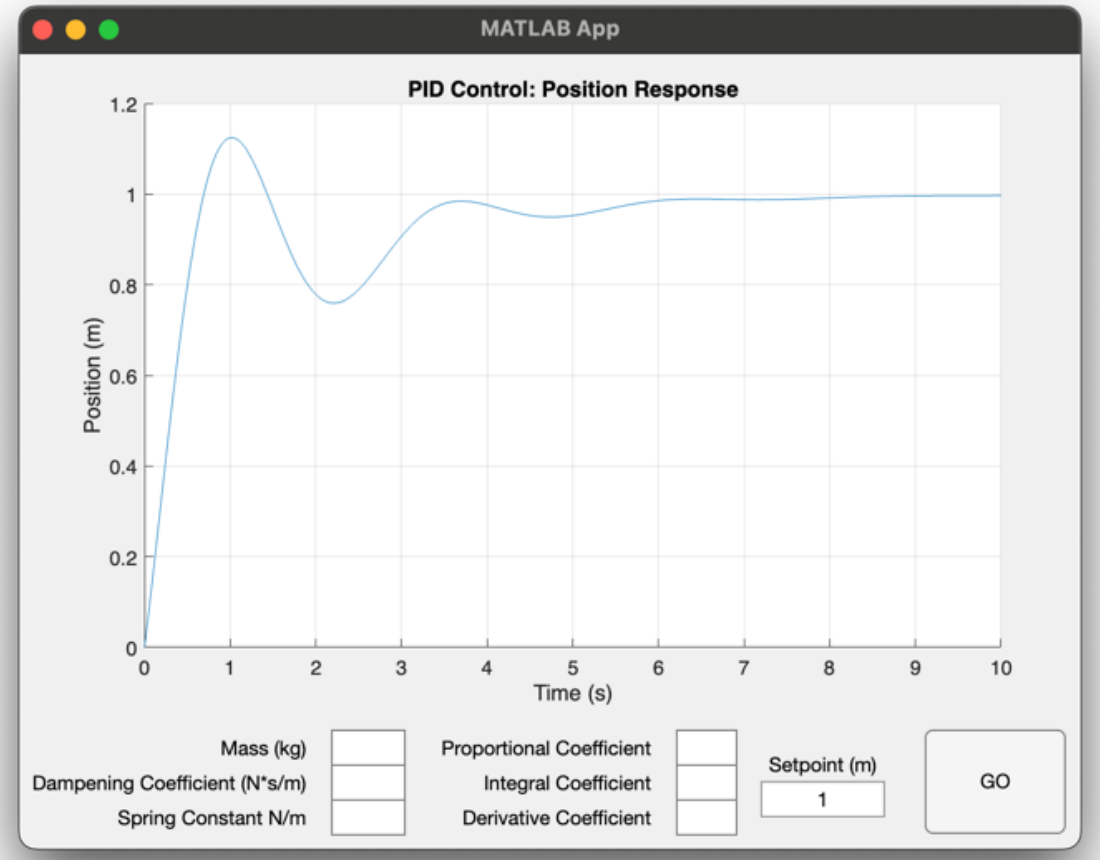

Figure 1: spring dampening comparison

Assignment 2: PID Line-Following Robot

Objective:

Develop a graphical application using MATLAB App Designer to simulate a line-following robot controlled by a PID algorithm. The application should allow the user to set PID gains and observe how the robot adjusts its path.

Requirements:

1. **Graphical Interface**:
   - Input fields for:
     - PID gains: kP, kI, kD
     - Setpoint for the Pololu sensor (e.g., 3500)
     - Random seed to allow repeatable simulation runs
       - Changing the seed will generate a new set of random numbers
       - Be sure to record the seed, along with the other numbers that you provide in the report.
   - Button to **run the simulation**.

- Plot showing sensor readings and motor speeds over time.
- Ability to **show the last two runs**:
  - The most recent run should be plotted with solid lines.
  - Example: The previous run can be plotted with dashed lines however, you may choose a different method if you feel it offers a better user experience.

2. **Functionality**:
   - Implement the PID control using the provided `simulateLineFollowerPololu` function.
   - Ensure the app clears previous plots and shows only the last two runs.
   - Allow the user to adjust the PID parameters and run the simulation multiple times, observing the differences.
3. **Submission**:
   - Submit your MATLAB `.mlapp` file along with a short write-up (1-2 paragraphs) explaining
     - How your app works
     - What did you find to be the best numbers
     - Were you able to determine a pattern in the three constants regarding how they control the results

Tips:

- Adjust the input random seed to see consistent deviations, making it easier to observe how the PID settings respond.
- Describe how changes in PID parameters affected the robot's performance and any observations you found interesting.
- Be sure to include the two datasets for Deviation, it may appear as though this dataset doesn't change, but it will if you change the seed which will be important to see for testing values across different datasets (seeds)

Example

As always, this is merely an example, you have free rein to design however you feel so long as it meets the requirements.

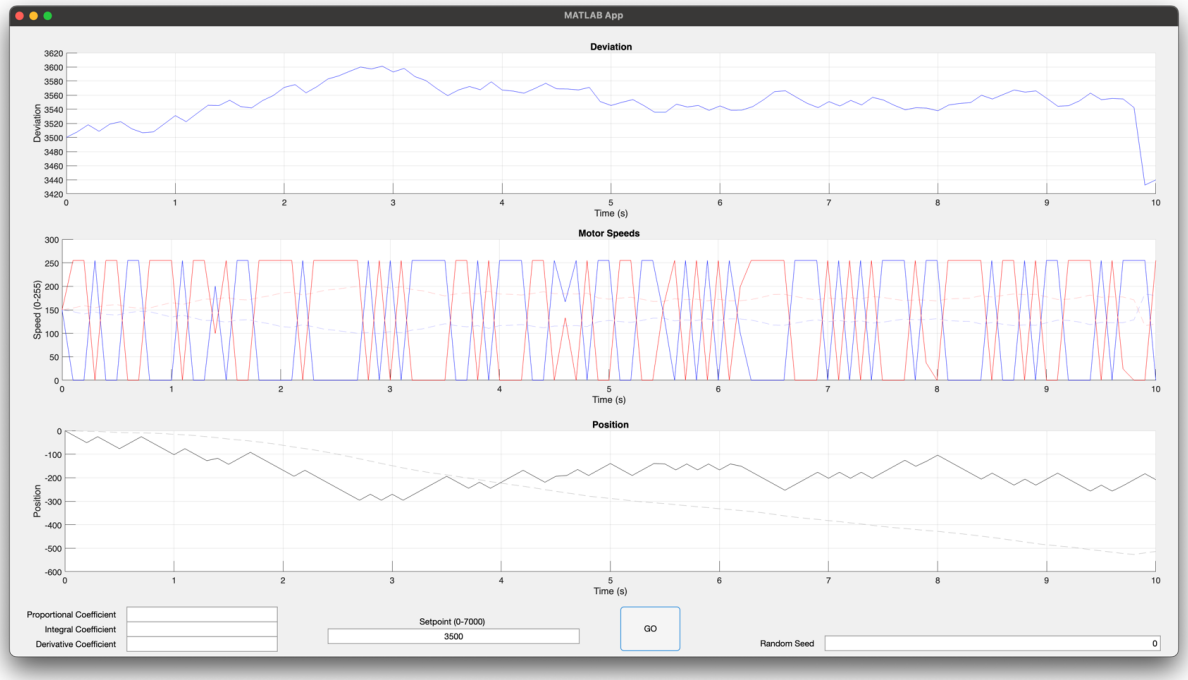

Figure 2: line follow comparison

Assignment 3: Custom Algorithm Comparison

Objective:

Select an algorithm or a set of algorithms, and develop a graphical application using MATLAB App Designer to perform comparative analysis. You may choose to:

1. Run a single algorithm on multiple datasets or with varying parameters, OR
2. Run multiple algorithms on the same dataset or with the same parameters.

The goal is to demonstrate an understanding of how algorithmic behavior changes with different inputs or configurations and to present results in a clear, comparative format.

Requirements:

1. **Graphical Interface**:
   - Allow input of all necessary parameters for the chosen algorithm.
   - Button to **run the simulation**.
   - Plot that shows the output of the algorithm over time or as appropriate.
   - Ability to **show the last two runs**:
     - The most recent run should be plotted with solid lines.
     - Example: The previous run can be plotted with dashed lines however, you may choose a different method if you feel it offers a better user experience.
2. **Functionality**:
   - Choose any algorithm you are interested in that involves tuning parameters and observing performance.
   - Implement the algorithm and provide functionality to easily compare different runs.
   - The application should handle different inputs and visualize how changes affect the system response.

3. **Submission**:
    - Submit your MATLAB `.mlapp` file along with a short write-up (1-2 paragraphs) that explains:
        - The chosen algorithm and why you selected it.
        - How your app is designed, and how to use it.
        - Observations and insights from comparing different runs.
        - Screenshots of the application and plots (These can just be image files in the folder).

Tips:

- For comparing algorithms, you might choose different sorting, searching, or optimization techniques and evaluate their efficiency or accuracy.
    - If you choose this, you may need very large sets of data to achieve a noticeable difference so plan accordingly.
- For comparing datasets or parameters, you could adjust input data distributions, noise levels, or control settings.
- Consider what insights you'd like to gain from the comparison and design your app accordingly.

General Submission Guidelines:

1. Each `.mlapp` file should be well-organized and user-friendly.
2. Include your write-up in any format (word, text, md, pdf, etc), clearly explaining your approach and observations.