Iterative Learning Control with MuJoCo Simulation

1 Introduction

This document provides an in-depth explanation of a Python script that implements iterative learning control (ILC) using the MuJoCo physics engine. The script simulates a two-link robotic arm, iteratively updates control torques, and visualizes the results. The purpose of this documentation is to guide you through each part of the script, detailing its functionality and implementation.

2 Step-by-Step Guide

2.1 Step 1: Import Required Libraries

First, we need to import the necessary libraries:

- **csv**: For handling CSV file operations.
- **matplotlib.pyplot**: For plotting the results.
- **mujoco.viewer**: For visualizing the MuJoCo simulation.
- **numpy**: For numerical operations and handling arrays.

These libraries provide the tools needed for data manipulation, visualization, and simulation.

2.2 Step 2: Define Constants

Next, define constants that control the learning behavior:

- **LAMBDA_FACTOR**: The forgetting factor which influences the retention of previous control torques.
- **LEARNING_RATE**: The learning rate which determines the magnitude of the update applied to the control torques based on the error.

2.3 Step 3: Iterative Learning Update Function

Create a function called `iterative_learning_update` which updates the control torques. This function performs the following steps:

1. Calculate the error between the desired and current angles.
2. Compute the control update using the learning rate and the error.
3. Update the new torques by combining the previous torques with the control update, scaled by the forgetting factor.

This function is crucial for the iterative learning process as it adjusts the torques to minimize the error over time.

2.4 Step 4: Generate Desired Trajectories

Define a function called `generate_waveform` that generates the desired joint angle trajectories over time. Typically, sinusoidal waveforms are used for this purpose. The function normalizes the time step and computes the target angles using sine and cosine functions.

2.5 Step 5: Simulate with Phases and Viewer

The main simulation function, `simulate_with_phases_and_viewer`, performs the following tasks:

1. Initialize data structures to store results.
2. Generate the desired angles for the entire simulation duration.
3. For each trial:
    - Reset the joint positions.
    - Determine the current phase and corresponding perturbation.
    - For each time step:
        - Perform a simulation step in MuJoCo.
        - Record the current joint angles.
        - Update the control torques using the `iterative_learning_update` function.
        - Apply the new torques to the actuators.
4. Calculate the mean squared error (MSE) between the desired and final angles.

This function iterates over multiple trials and time steps, updating and recording the system's behavior.

2.6 Step 6: Define Phase and Perturbation Functions

Two helper functions, `get_phase` and `get_perturbation`, are defined to manage the experimental phases:

- **get_phase**: Determines the phase of the trial (Baseline, Adaptation, Washout, Readaptation) based on the trial number.
- **get_perturbation**: Returns the perturbation value for the current phase, used to simulate different experimental conditions.

These functions help in organizing the simulation into different phases and applying appropriate perturbations.

2.7 Step 7: Plot Functions

Several plotting functions are defined to visualize the results:

- **plot_specific_trials_theta_time_series**: Plots the joint angles over time for specific trials.
- **plot_error_vs_timesteps**: Plots the error between the desired and actual angles over time for specific trials.
- **plot_previous_torques_series**: Plots the control torques over time for specific trials.
- **plot_mse_across_phases**: Plots the mean squared error across different phases of the simulation.

These plots provide insights into the performance and behavior of the ILC system.

2.8 Step 8: Write Results to CSV

The `write_to_csv` function writes the simulation results, specifically the error results, to a CSV file. This allows for easy storage and further analysis of the data.

2.9 Step 9: Main Function

The main function orchestrates the entire process:

1. Load the MuJoCo model and initialize the simulation data.
2. Launch the MuJoCo viewer for visualization.
3. Run the simulation using `simulate_with_phases_and_viewer`.
4. Write the results to a CSV file.

5. Generate and display the plots for analysis.

This function serves as the entry point to the script and coordinates all the other functions.

3 Conclusion

This detailed documentation outlines the implementation of iterative learning control using the MuJoCo physics engine. By following these steps, one can simulate a two-link robotic arm, apply iterative learning control, and visualize the results to study the performance and behavior of the system. This approach provides a structured way to implement and analyze ILC in robotic systems.