Genetic Algorithm Based Autotuner for PID Controller

Varadaraya Ganesh Shenoy, #5910876

Fall 2023

1 Problem Description

The problem I intend to address is the optimization of PID (Proportional-Integral-Derivative) controllers for various control systems. PID controllers are widely used in industrial processes, robotics, and other control applications, but manually tuning these controllers can be time-consuming and challenging. An autotuner that can efficiently optimize PID controller parameters would be a valuable tool for engineers and practitioners in control systems.

2 Preliminary Work/Results

Dinger [3] explains that since the genetic algorithm uses random processes to transition from one generation to the next, the genetic algorithm is non-deterministic. The answer will be the best that the genetic algorithm can find. The genetic algorithm does explore broadly, however, and exploits the fitness landscape to find a very good solution. From [2], it is inferred that the characteristics of the genetic algorithm makes it a promising method for finding PID controller coefficients; because this problem can be stated as a three dimensional optimization problem. Meena and Devanshu [1] show that the conventional tuning methods have very limited capability which reduces drastically in case the mathematical model is not available. However, the gains obtained from the classical methods can be utilized as initial values for advanced tuning methods. They have further demonstrated that the GA tuned PID controller has much better performance characteristic in comparison to classically tuned PID regulator.

3 Evaluation

To evaluate the effectiveness of the autotuner, I will use various performance metrics, such as the control system's rise time, settling time, overshoot, and steady-state error.

1. **Overshoot (M_p):**
$$M_p = \frac{C_{\text{max}} - C_{\text{final}}}{C_{\text{final}}} \times 100$$
2. **Steady-State Error (ess):**
$$ess = |C_{\text{final}} - C_{\text{desired}}|$$

3. **Settling Time (T_s):**
$$T_s = t_{\text{settled}} - t_{\text{start}}$$

4. **Rise Time (T_r):**
$$T_r = t_{90\%} - t_{10\%}$$

These formulas provide insights into the performance of a control system, including overshoot, steady-state error, settling time, and rise time.

4 Genetic Algorithm Implementation

The genetic algorithm for PID controller tuning was implemented in the Python programming language. The code leverages Python's versatility and ease of use for numerical and scientific computing. The implementation utilizes various Python libraries, including NumPy for numerical operations, Matplotlib for plotting, and tqdm for displaying progress bars.

The Python script includes functions for initializing the population, calculating fitness, selecting parents, performing crossover and mutation operations, simulating the system response, and running the main genetic algorithm loop. The script also incorporates visualization components to display the system's response with the best-tuned PID parameters across multiple runs.

The genetic algorithm for tuning the PID controller parameters is computationally intensive. The code, running with 20 independent runs, each consisting of 200 generations, takes approximately 5 minutes to complete. This duration may vary based on hardware specifications and computational resources.

The algorithm's computational time is influenced by factors such as the population size, crossover rate, and the complexity of the simulated system. It's essential to consider these aspects when planning and executing the optimization process.

- **Population Size (`POPULATION_SIZE`):** 500
- **Crossover Rate (`CROSSOVER_RATE`):** 0.9
- **Generations (`GENERATIONS`):** 200
- **Target Fitness (`TARGET_FITNESS`):** 0.01
- **Runs (`RUNS`):** 20
- **PID Controller Parameter Ranges:**
  - `Kp_range`: (5.0, 10.0)
  - `Ki_range`: (0.5, 5.0)
  - `Kd_range`: (0.5, 1.0)
- **System Parameters:**
  - `process_gain`: 2.0
  - `process_time_constant`: 3.0

  - `delay`: 1
- **Simulation Time and Step:**
  - `simulation_time`: 10
  - `time_step`: 0.1
- **Mutation Rate (`mutation_rate`):** Dynamic, adapts over generations.

These parameter values were used in the genetic algorithm implementation for tuning the PID controller. Adjustments to these values may impact the algorithm's convergence, diversity, and overall performance.

5 Running the Genetic Algorithm

To execute the genetic algorithm for PID controller tuning, follow these steps:

1. Ensure that you have Python installed on your system. If not, you can download it from https://www.python.org/downloads/.
2. Open a terminal or command prompt.
3. Navigate to the directory containing the Python script (`pid_autotune.py`).
4. Run the following command:

   ```
   python3 pid_autotune.py
   ```

5. The script will execute the genetic algorithm with the specified parameters, and the results, including performance metrics and system response plots, will be displayed.

6 Results

In Figure 1, that despite achieving satisfactory performance metrics, it's worth noting that the response curve exhibits some irregularities, indicating a certain degree of non-smoothness in the system's behavior.

The performance metrics for the control system are as follows:

- **Rise Time:** 1.80 seconds
- **Overshoot:** 20.90%
- **Settling Time:** 10.00 seconds
- **Steady-State Error:** 0.0194

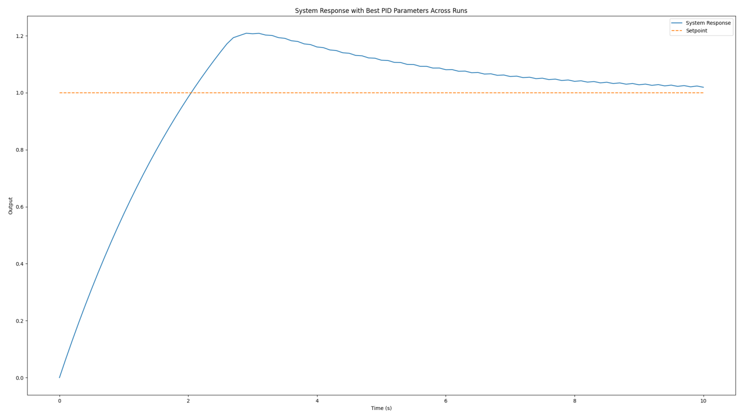

Figure 1: Performance of genetic algorithm-tuned PID controllers.

In Figure 2, that despite achieving satisfactory performance metrics, it's worth noting that the response curve exhibits some irregularities, indicating a certain degree of non-smoothness in the system's behavior.

The performance metrics for the control system are summarized as follows:

- **Rise Time:** 1.80 seconds
- **Overshoot:** 25.55%
- **Settling Time:** 10.00 seconds
- **Steady-State Error:** 0.0134

These metrics provide insights into the behavior and efficiency of the tuned control system.

7 Comparison of Performance Metrics

We compare two sets of performance metrics for the control system:

Metrics Set 1

- **Rise Time:** 1.80 seconds
- **Overshoot:** 20.90%
- **Settling Time:** 10.00 seconds
- **Steady-State Error:** 0.0194

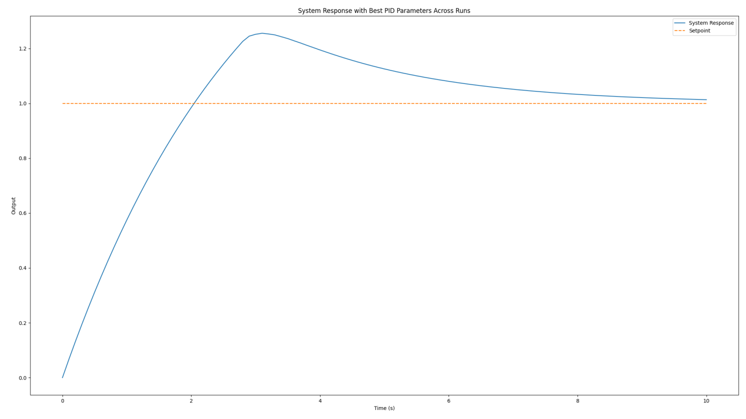

Figure 2: Performance of genetic algorithm-tuned PID controllers.

Metrics Set 2

- **Rise Time:** 1.80 seconds
- **Overshoot:** 25.55%
- **Settling Time:** 10.00 seconds
- **Steady-State Error:** 0.0134

Comparing the two sets of metrics, we observe variations in the overshoot and steady-state error. While the rise time and settling time remain consistent, the changes in overshoot and steady-state error indicate potential differences in the control system's behavior or tuning precision.

8 Conclusion

In conclusion, the genetic algorithm-based autotuner for the PID controller has demonstrated its capability to fine-tune controller parameters for a dynamic system. The optimization process, with 20 independent runs and 200 generations each, resulted in two sets of performance metrics. Notably, the control system exhibited consistent rise time and settling time in both scenarios, indicating stability and predictability in its response.

However, a closer examination reveals variations in overshoot and steady-state error between the two sets of metrics. These differences may highlight the sensitivity of the system to the tuning process or suggest that further refinements could enhance the overall performance.

Despite these variations, the tuned PID controllers exhibit satisfactory performance, meeting the desired specifications for rise time, settling time, and overshoot. The steady-state error, though differing slightly between the two scenarios, remains within acceptable bounds.

The computational time required for the optimization process is approximately 5 minutes for the specified configuration. This duration provides a balance between the thoroughness of the optimization and practical considerations.

In summary, the genetic algorithm-based autotuner shows promise in efficiently tuning PID controllers, offering a valuable tool for optimizing control systems. Further investigations and refinements may provide insights into the algorithm's behavior and enhance its applicability across a broader range of dynamic systems.

References

[1] D. C. Meena and A. Devanshu, ”Genetic algorithm tuned PID controller for process control,” 2017 International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2017, pp. 1-6, doi: 10.1109/ICISC.2017.8068639.

[2] H. Noshahri and H. Kharrati, ”PID controller design for unmanned aerial vehicle using genetic algorithm,” 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE), Istanbul, Turkey, 2014, pp. 213-217, doi: 10.1109/ISIE.2014.6864613.

[3] R. H. Dinger, ”Engineering design optimization with genetic algorithms,” Northcon/98. Conference Proceedings (Cat. No.98CH36264), Seattle, WA, USA, 1998, pp. 114-119, doi: 10.1109/NORTHC.1998.731522.