Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. X No. Y (20--)

Applying Particle Swarm Optimization to Refine PID Controllers for Second-Order Linear Systems

Nulaksala vamsi Krishna , Dr. K. Rajasekhar

Department of ECE,UCEK,JNTUK,Kakinada,India,

Department of ECE,UCEK,JNTUK,Kakinada,India,

Abstract—This research proposes a novel approach for designing a proportional-integral-derivative (PID) controller for a class of second-order linear systems. The method employs particle swarm optimization implemented in MATLAB to optimize the PID controller parameters based on time domain specifications. Particle Swarm Optimization (PSO) has emerged as a valuable tool for the fine-tuning of PID controllers, complemented by the formulation of appropriate fitness and constraint functions. This methodology facilitates the automatic and efficient adjustment of PID controller parameters, a crucial factor in achieving optimal control performance across diverse systems. In our study, we showcase the implementation of PSO and its profound impact on enhancing controller performance. This enhancement is characterized by improved system stability, minimized overshoot, and swifter response times. Emphasizing the pivotal role of the fitness function, we demonstrate its significance in steering the optimization process by quantifying system performance, ensuring the alignment of controller parameters with predefined control objectives.

Keywords*: PID controller,Particle swarm optimizatio, fitness function, optimization*

1. Introduction

The classical Proportional-Integral-Derivative (PID) controller is widely recognized for its essential role in controlling diverse engineering systems [1]. Particle Swarm Optimization (PSO) has emerged as a valuable technique for fine-tuning PID controllers, offering an automated and efficient means to optimize the critical parameters governing control performance across diverse systems represent the most widely used technology for industrial process control[8] .By defining a well-suited fitness function and constraint functions, PSO empowers us to automatically adjust PID controller parameters to attain optimal control performance, In this paper, we will delve into the implementation of PSO and its profound impact on improving controller performance, ultimately leading to heightened system stability, diminished overshoot, and faster response times. This expansion of traditional PID controllers adds robustness and flexibility to the system [2].we further believe that control engineering may very well break the hold of classical PID and enter a new era, an era that brings back the spirit of innovation[10]. Our investigation will underline the significance of the fitness function in guiding the optimization process by quantifying system performance and ensuring that controller parameters align with predefined control objectives These parameters are used here proportional gain constant (KP), the integral gain constant (KI), the derivative gain constant (KD) [2].

2. Objectives

Designing A Pid Controller Under Time Domain Specificcations

When it comes to creating a PID controller while adhering to time domain requirements, the process entails the careful selection of controller parameters to attain the desired control performance, focusing on factors like settling time, rise time, overshoot, and steady-state error. Below is a summarized method for crafting a PID controller in accordance with time domain specifications. The PID controller operates by effectively modulating the variable to be controlled using a balanced combination of three distinct control actions: the Proportional (P) control action, the Integral (I) control action, and the Derivative (D) control action. Each of these actions serves a specific purpose in ensuring precise control.

Firstly, the Proportional (P) action responds to the actuating error signal, which is essentially the difference between the desired input and the feedback signal from the system. The P action is directly proportional to this error, which means it provides an immediate response based on the magnitude of the error. This helps in reducing deviations from the desired setpoint [13].

Secondly, the Integral (I) action focuses on the integral or accumulation of the actuating error signal over time. It is proportionate to the integral of the error and is particularly useful in addressing steady-state errors that may persist even after the P action has been applied. The I action helps in eliminating long-term deviations from the setpoint.

Lastly, the Derivative (D) action is proportional to the rate of change or derivative of the actuating error signal. It responds to how quickly the error is changing, which can be essential in preventing overshoot and oscillations. The D action adds damping to the control system and helps stabilize it.

By combining these three control actions – P, I, and D – the PID controller can effectively regulate the controlled variable. The P action provides immediate response to current errors, the I action eliminates accumulated errors over time, and the D action contributes to system stability by counteracting rapid changes in the error. This amalgamation of control actions results in a continuous and effective PID control mechanism, widely used in various industrial and engineering applications to maintain desired system behavior and performance [13]. PID yields good control performance and is not complicated to configure [14].

1. Understand the System:

Begin by comprehensively grasping the dynamic characteristics of the system under control. This involves determining its order (e.g., 2nd order) and acquiring either its transfer function or state-space representation.

2. Define Desired Specifications:

Specify the time domain requirements that best align with your control objectives and the system's nature. Key time domain criteria include settling time (ts), rise time (tr), overshoot (OS), and steady-state error (e). Establish appropriate values or ranges for these criteria based on your control needs.

3. Configure Proportional Gain (Kp):

Initiate the tuning process by setting the derivative and integral gains (Kd and Ki) to zero, focusing initially on adjusting the proportional gain (Kp). Gradually increase kp until you achieve the desired rise time and overshoot, thus molding the transient response as needed.

4. Incorporate Derivative Gain (Kd):

Introduce derivative action by incrementally enhancing the derivative gain (Kd). This step contributes to better damping and reduced overshoot. Adjust Kd to meet your settling time and overshoot specifications, but exercise caution as excessively high Kd values can introduce instability or amplify noise.

5. Apply Integral Gain (Ki):

Now, introduce integral action by progressively raising the integral gain (Ki). This helps eliminate steady-state error and enhance system stability. Adjust Ki to meet your steady-state error specifications while ensuring system stability. Exercise prudence with high Ki values, as they can lead to oscillations or instability.

6. Refine and Iterate:

Continuously iterate and fine-tune the PID controller parameters (Kp, Ki and Kd) based on your desired time domain criteria. Adjust each parameter incrementally while closely observing the system response, striking a balance between response speed (rise time, settling time) and stability (overshoot) to meet control requirements.

7. Simulate and Validate:

Validate your designed PID controller by simulating it alongside the system model. Employ suitable simulation tools or software to assess the response, verifying if it aligns with the specified time domain criteria. If necessary, make further adjustments and repeat the simulation until satisfactory results are attained.

8. Implement and Experiment:

Proceed to implement the PID controller the actual system and conduct experimental testing. Monitor the system's response and compare it to the desired time domain specifications. Fine-tune the controller parameters as required, based on the observed performance, to ensure they align with your control objectives

3. Methods

METHODOLOGY

Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the social behavior of birds and fish. It's used to find solutions to various optimization and search problems. Here are the key components and principles of PSO:

1.Population of Particles: PSO maintains a group of particles, each representing a potential solution to the problem.

2. Position and Velocity: Each particle has a position and a velocity vector. The position represents a potential solution, and the velocity determines its movement direction and speed.

3. Fitness Function: A fitness function evaluates how good a solution is based on the optimization objectives.

4. Personal Best (p Best): Each particle remembers the best solution it has encountered so far (its personal best).

5. Global Best (g Best): The best solution found by any particle in the entire group is considered the global best.

6. Movement Update: Particles update their velocities and positions based on their own best solution and the global best solution found so far.

7. Balancing Exploration and Exploitation: PSO balances exploration (searching a wide area of the solution space) and exploitation (refining solutions in promising areas) through velocity updates.

In Particle Swarm Optimization (PSO), the initial stage involves populating the "swarm" with a set of random solutions. Within this swarm, each individual particle represents a distinct configuration of unknown parameters that are subject to optimization. These particles essentially serve as points within the solution space. As the optimization process unfolds, each particle endeavors to adapt its trajectory, moving towards areas in the solution space that hold the promise of better outcomes. This adjustment is guided by the particle's own past experiences and is complemented by the sharing of information among particles within the swarm, fostering a collaborative approach to optimization [13]. The PSO algorithm is often used to optimize various functions or parameters. In your context, it's applied to the design of a PID (Proportional-Integral-Derivative) controller. MATLAB is a popular programming tool for implementing and visualizing PSO-based algorithms due to its efficiency and capabilities for numerical computing and plotting. MATLAB can assist in coding, optimizing, and graphically representing the steps of the PSO-based PID controller design algorithm. The primary goal in Particle Swarm Optimization (PSO) is to conduct an effective exploration of the solution space. This is achieved by orchestrating the collective movement of particles towards the most promising solutions discovered in prior iterations. The ultimate aim is to continually encounter improved solutions as the optimization process unfolds, ultimately guiding the swarm towards the convergence upon a single solution with the minimum error [13].

The Particle Swarm Optimization (PSO) algorithm finds valuable application in the optimization process, particularly in tasks like tuning a PID (Proportional-Integral-Derivative) controller for optimal system performance. To achieve this, a fitness function is employed, critically evaluating how effectively the PID controller regulates the system, taking into account essential criteria such as settling time, overshoot, or error reduction. The process initiates with the setup of initial positions and velocities for the particles, representing candidate PID controller parameters. Subsequently, the algorithm determines both the global best (g Best) position, denoting the best overall solution across all particles, and the personal best (p Best)

positions for each particle. This information guides the PSO optimization loop, where particles iteratively adjust their positions and velocities to converge towards the optimal PID controller parameters. Upon loop termination, the optimized PID controller parameters are extracted. Finally, a closed-loop control system simulation is conducted, utilizing the optimized PID controller, to validate and assess its performance in regulating the system according to the specified criteria.

THE FITNESS FUNCTION

The fitness function is $\theta 1\ (t) - \theta ss$

where $t = ts,d + 0.05 \times ts,d = 1.05$. A 5% of ts,d is chosen for the tolerance ρ here. Fitness function evaluates the deviation of system's output theta1 at a specific time 't' from the steady state value theta SS.

Particle Swarm Optimization (PSO):

Objective of the Fitness Function:

In PSO-based PID tuning, the fitness function evaluates how well a given set of PID parameters satisfies the control system's performance specifications.

Performance Metrics:

performance metrics such as settling time (ts), overshoot (Mp), and steady-state error (ess) are often used.

Fitness Calculation:

The fitness function calculates a fitness score based on the control system's response to the PID controller with the given parameters.

The fitness function quantifies how closely the system's response matches the desired response, similar to GA.

Multi-Objective Optimization (Optional):

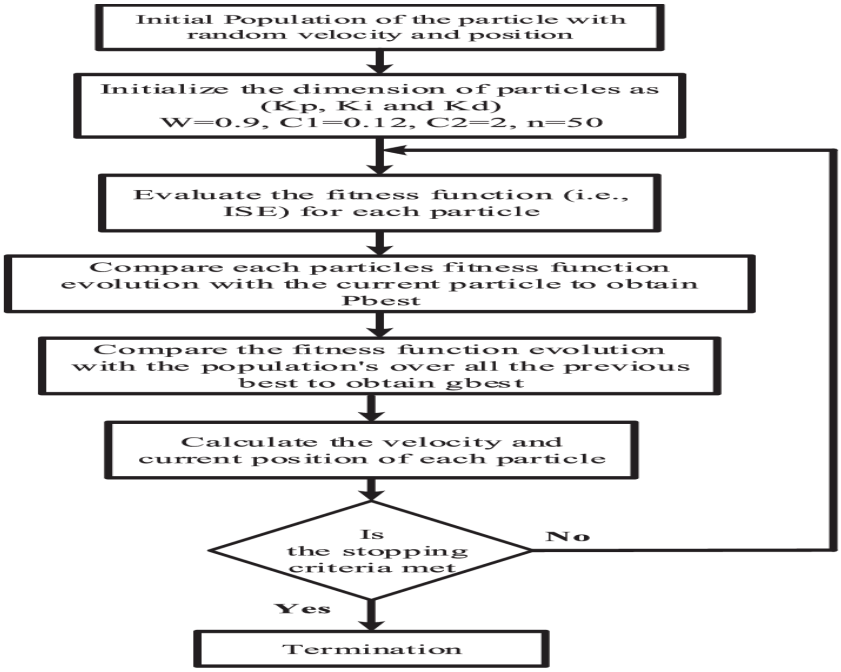
In complex control problems, a multi-objective fitness function may be used to balance multiple control objectives.

Constraints (Optional):

If constraints on PID parameters exist, the fitness function should ensure that parameter sets violating these constraints receive a high fitness score.

Fitness Function Output:

The fitness function returns a fitness score, where lower values indicate better solutions.PSO aims to minimize this fitness score during the optimization process



4. Results

Genetic algorithm outputs

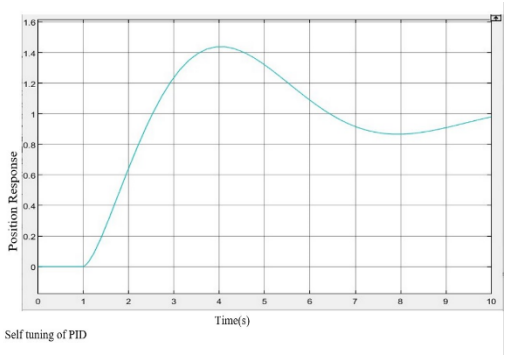


Figure1 self tunning of PID

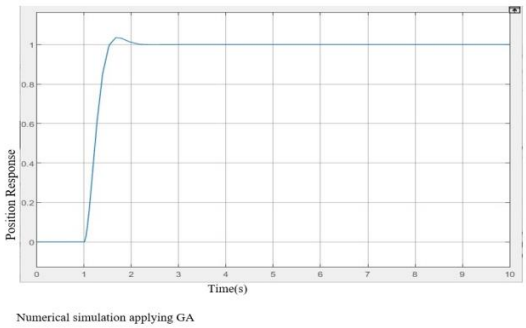


Figure2 Numerical simulation applying GA

Genetic Algorithm (GA):

PID Controller Parameters:

Kp = 9.999

Ki = 0.0096

Kd = 1.122

Performance Specifications:

Settling Time (ts) = 1.0 seconds

Overshoot (Mp) = 0.12 (12%)

PSO outputs

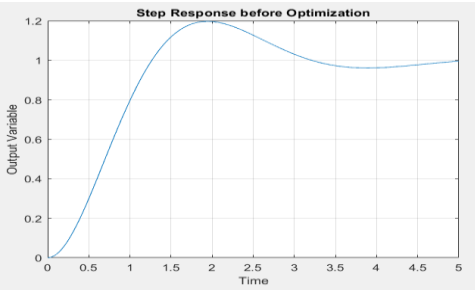


FIGURE3: Numerical simulation applying optimization

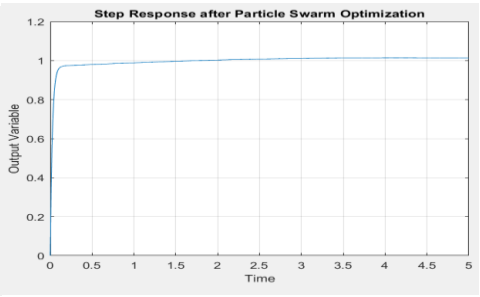


FIGURE4: Numerical simulation applying MATLAB before after optimization

Particle Swarm Optimization (PSO):

PID Controller Parameters:

Kp = 12.2164

Ki = 5.8188

Kd = 5.3541

Performance Specifications:

Settling Time (ts) = 0.5 seconds

Overshoot (Mp) = 0.12 (12%)

For both PSO and GA, the step response before optimization shows the system's response to a step input. It can be seen in the FIGURE1 and FIGURE3 overshoot is high for both the responses.

After optimization process, the PID controller parameters were tuned to optimal values. The step response graph which is in FIGURE2 after optimization using GA demonstrated the system's improved response to a step input. The step response graph which is in FIGURE4 after optimization using PSO also exhibited an enhanced system response to a step input.

The step response is settled smoothly after optimization process with less overshoots

Both GA and PSO-based tuning methods have been able to achieve the desired overshoot of 12%. However, there is a difference in the settling time.

GA: Settling time achieved is 1.0 seconds.

PSO: Settling time achieved is 0.5 seconds.

PSO has achieved a lower settling time (0.5 seconds) compared to GA (1.0 seconds). This means that the PSO-tuned controller responds faster to reach its desired state. Both methods have achieved the same overshoot value of 12%, indicating that they satisfy this control objective equally well, both GA and PSO have successfully tuned the PID controller to meet the desired overshoot specification. PSO has achieved a faster settling time, which can be beneficial in some applications.

5. Discussion

In summary, Particle Swarm Optimization (PSO) has proven to be a valuable tool for fine-tuning PID controllers. By defining a suitable fitness function and constraint functions, we enable an automated and efficient adjustment of PID controller parameters, a critical aspect of achieving optimal control performance across different systems. Our implementation of PSO has shown significant improvements in controller

performance, resulting in enhanced system stability, reduced overshoot, and quicker response times. The fitness function's role in quantifying system performance has been pivotal, ensuring that controller parameters are precisely adjusted to meet specific control objectives, making PSO an effective and versatile approach for PID controller tuning.

Refrences

[1] J. Zhang and L. Guo, ‘‘Theory and design of PID controller for nonlinear uncertain systems,’’ IEEE Control Syst. Lett., vol. 3, no. 3, pp. 643–648, Jul. 2019.

[2] B. Hekimoglu, ‘‘Optimal Tuning of fractional order PID controller for DC motor speed control Via chaotic atom search optimization algorithm,’’ IEEE Access, vol. 7, pp. 38100–38114, 2019.

[3] Z. Gao, "Scaling, Multistep Design Method for PID Controllers," ISA Transactions, vol. 33, no. 4, pp. 325-336, 1994.

[4] X. Li and D. Atherton, "An Improved Relay Auto tuner Based on the Proportional-Integral-Derivative Controller," IEEE Transactions on Industrial Electronics, vol. 39, no. 5, pp. 414-418, 1992.

[5] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley Professional, 1989.

[6] J. H. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, 1975.

[7] S. K. Dash and S. R. Mishra, "A Comparative Study of Various Types of Genetic Algorithms," International Journal of Computer Applications, vol. 32, no. 3, pp. 1-5, 2011.

[8] M. Huba, D. Vrancic, and P. Bistak, ‘‘PID control with higher order derivative degrees for IPDT plant models,’’ IEEE Access, vol. 9, pp. 2478–2495, 2021.

[9] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," Journal of Global Optimization, vol. 11, no. 4, pp. 341-359, 1997.

[10] J. Han, ‘‘From PID to active disturbance rejection control,’’ IEEE Trans. Ind. Electron., vol. 56, no. 3, pp. 900–906, Mar. 2009.

[11] K. Astrom and T. Hagglund, "PID Controllers: Theory, Design, and Tuning," Instrument Society of America, 1995. [12] R. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications, and Resources," Proceedings of the Congress on Evolutionary Computation, vol. 1, pp. 81-86, 2001.

[12] A. Rastogi and P. Tiwari, ‘‘Optimal tuning of fractional order PID controller for DC motor speed control using particle swarm optimization,’’ Int. J. Soft Compute. Eng., vol. 3, no. 2, pp. 150–157, 2013.

[13] D. C. Meena and A. Devanshu, ‘‘Genetic algorithm tuned PID controller for process control,’’ in Proc. Int. Conf. Inventive Syst. Control (ICISC), Jan. 2017.

[14] A. A. M. Zahir, S. S. N. Alhady, W. A. F. W. Othman, and M. F. Ahmad, ‘‘Genetic algorithm optimization of PID controller for brushed DC motor,’’ in Intelligent Manufacturing & Mechatronics. Singapore: Springer, 2018.