Particle Swarm Optimization approach for optimal design of PID controllers

Ashwin R Nair (122101004), Nakul C (122101024)

Abstract: Proportional-Integral-Derivative (PID) control is one of the most common control algorithms used in the industry. A PID controller reads a sensor and computes its proportional, integral and derivative responses and computes the output. It is used to improve the dynamic response and to reduce the steady state error. Random search methods like genetic algorithm and simulated annealing have been used in control systems in search of optimal PID control parameters. But due to the computational complexity and degradation in efficiency, we need to look for other suitable methods. One such method was Particle Swarm Optimization (PSO). In this project we compare the characteristics of PSO-PID controller and GA-PID controller using Matlab and control systems toolbox and compare the performance of both methods.

I. Objectives

Proportional-Integral-Derivative (PID) controller is a widely used control method in the industry. However it is quite difficult to tune its gains due to its high order, time delays, and non linearities. One of the classic methods used to tune the gains of PID controllers is the Ziegler-Nichols method. But using this method also it is hard to determine near optimal PID parameters. The genetic algorithm method has high potential for optimization but requires high computational efforts. Particle Swarm Optimization (PSO) is one of the modern heuristic algorithms that is very robust in solving nonlinear optimization problems. In this project, our objective is to compare the characteristics and performance of both GA-PID and PSO-PID controllers and assess their computational efficiency.

II. Motivation / Origin of the project

PID controllers are one of the widely used control system in the industry. So optimizing the gains of PID controllers is very crucial. Even though there are random search methods like GA, they are very computationally complex. This prompts us to search for other algorithms that can provide optimal solutions for effective tuning of PID controllers. PSO is one such method that is simple, can converge fastly and is very efficient in tuning PID controllers. This motivates us to compare the characteristics of GA-PID and PSO-PID to assess the advantages and disadvantages of both methods and arrive at a conclusion that which of these methods is better and can be applied in real world control systems.

These methods, inspired by nature, offer innovative solutions to complex optimization problems by emulating natural systems' behavior. PSO, for example, is inspired by the social behavior of bird flocks and fish schools, where individuals adjust their movements based on the best-performing peers.

Similarly, GA mimics the process of natural selection and evolution, where individuals with better traits have a higher chance of survival and reproduction. By harnessing these principles, nature-inspired optimization methods can efficiently search through vast solution spaces and find optimal solutions in complex and dynamic environments.

III. State-of-the-art

Over the years there has been different methods introduced to tune PID controllers. There are gradient-based optimization methods like gradient descent algorithms, stochastic gradient descent(SGD) and Adam optimizer. There are many metaheuristic algorithms like Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Ant Colony Optimization (ACO), Differential Evolution (DE), and Harmony Search (HS) that are also good

optimization techniques and each offers a unique convergence property.

In recent times, machine learning based techniques have seen a rise with neural networks, deep learning and reinforcement learning that can easily integrate with PID controllers and learn optimal control based on system feedback. There are works going on that combines one or more of these methods to come up with optimization strategies to improve convergence speed.

IV. Methodology/Approach

A. PID Controller

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{1}$$

The PID controller enhances the dynamic response and minimize steady-state errors of a system.

- The derivative controller improves transient response by introducing a finite zero to the open-loop plant transfer function.
- The integral controller adds a pole at the origin, boosting the system type by one and reducing steady-state errors caused by step functions to zero.

Hence, the PID controller fine-tunes the system's behavior, making it respond faster and more accurately to changes.

B. Modeling a Linearized AVR System

The Automatic Voltage Regulator (AVR) is used for maintaining a steady voltage from a generator. It's made up of four parts: amplifier, exciter, generator, and sensor. We've made the AVR system into a linearized model by simplifying each part's behaviour:

- **Amplifier**: is used for boosting signals quickly. We represented it with a gain (K_A = 10) and a small time constant (τ_A = 0.01).

$$\frac{V_R(s)}{V_e(s)} = \frac{10}{1 + 0.01s} \tag{2}$$

- **Exciter**: is used for regulating the generator's field. Hence, It'll have a longer time constant (τ_e = 0.4).

$$\frac{V_F(s)}{V_R(s)} = \frac{1}{1 + 0.4s} \tag{3}$$

- **Generator**: converts mechanical energy into electrical energy. Its response depends on the load and has a time constant (τ_g = 1).

$$\frac{V_t(s)}{V_F(s)} = \frac{1}{1 + s} \tag{4}$$

- **Sensor**: detects changes in voltage. Hence, has a small time constant (τ_s = 0.01).

$$\frac{V_s(s)}{V_t(s)} = \frac{1}{1 + 0.01s} \tag{5}$$

C. Performance Estimation of PID Controller

A new performance criterion as proposed by [1] was used. Once we get a set of good parameters for gains k_p, k_d and k_i, we can minimize the performance criterion. This performance criterion denoted by W(K) depends on maximum overshoot (M_p), rise time (t_r), settling time (t_s) and steady state error (E_{ss}).

$$W(K) = (1 - e^{-\beta})(M_p + E_{SS}) + e^{-\beta}(t_s - t_r) \tag{6}$$

where k is [kp, ki, kd] and β is the weighing factor. β is set to less than 0.7 if we want to reduce rise time and settling time and it is set to be more than 0.7 if we want to reduce maximum overshoot and steady state error. We've used a β of 0.8 to 1.5.

D. Genetic algorithm

In GA, initial population of candidate solutions is called chromosomes. These chromosomes are created randomly with parameters like population size, chromosome length, and encoding scheme defined.

Chromosomes with higher fitness are favored for reproduction, simulating the principle of "survival of the fittest." Common selection methods include roulette wheel, tournament, rank-based, and stochastic universal sampling.

Mutation introduces diversity by randomly altering genes or bits in offspring chromosomes, helping to prevent premature convergence. The mutation rate controls the probability of mutation occurrence. The next step involves replacing individuals in the current population with the new offspring and mutated chromosomes.

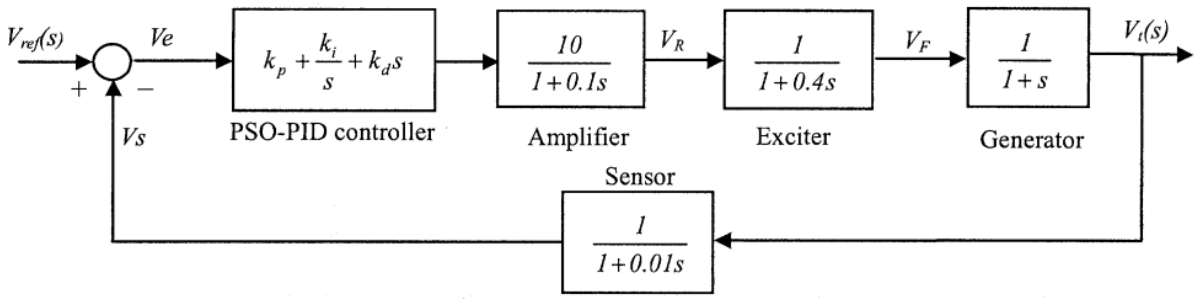

Fig. 1: AVR system with a PSO-PID controller

$$P(C_i) = \frac{f(C_i)}{\sum_{j=1}^{n} f(C_j)}, p_m = mutationrate \quad (7)$$

The algorithm iteratively repeats these steps until a termination criterion is met. This iterative process allows for continual improvement and exploration of the solution space until an optimal or near-optimal solution is obtained.

E. Particle Swarm Optimization

Particle Swarm Optimization was first described by James Kennedy and Russell C. Eberhart in 1995 was formulated on the idea of swarm intelligence based on the observation of swarming habits by certain kinds of animals like bees and birds.

PSO begins by initializing a population of potential solutions known as particles. Each particle represents a solution within the problem's search space. These particles collectively explore the solution space to find an optimal or near-optimal solution.

In the next step, each particle's objective function value is calculated. This assesses how well a particle performs in solving the optimization problem. The fitness function provides a quantifiable measure of the solution quality represented by each particle, guiding the search process towards better solutions.

In each iteration of PSO, particles update their velocities based on their current velocities, the best-known positions among other particles (local best), and the best-known positions among all particles in the population (global best). This velocity adjustment mechanism balances between exploration and exploitation.

After updating velocities, particles adjust their positions accordingly in the search space. The position update equation computes the new position of each particle based on its current position and updated velocity. This iterative process allows particles to traverse the solution space dynamically, converging towards optimal solutions over successive iterations.

$$\begin{aligned} \mathrm{x}_{ij}^{t+1} &= \mathrm{x}_{ij}^{t} + \mathrm{v}_{ij}^{t+1} \\ \mathrm{i} &= 1, 2, ..., n \\ \mathrm{j} &= 1, 2, ..., m \end{aligned} \quad (8)$$

If a particle's current position yields a better fitness value than its local best, it updates its local best to the current position. Similarly, the global best position among all particles is updated if a particle discovers a better solution than the current global best. This mechanism encourages information sharing among particles, facilitating the discovery of better solutions collectively.

$$\begin{aligned} \mathrm{velocity}_{ij}^{t+1} = w \cdot \mathrm{velocity}_{ij}^{t} \\ + c_1 \cdot r_1 \cdot (\mathrm{local\_best}_{ij} - \mathrm{position}_{ij}^{t}) \\ + c_2 \cdot r_2 \cdot (\mathrm{global\_best}_{ij} - \mathrm{position}_{ij}^{t}) \end{aligned} \quad (9)$$

The constants c1 and c2 are called acceleration constants. Low values of c1 and c2 allow the particle to roam far away from target regions and high values result in movements towards the target region. In this project we use c1 = c2 = 2 as mentioned in [].

The algorithm continues iterating through these steps until a termination criterion is met. Common termination criteria include reaching a maximum number of iterations or achieving a satisfactory solution within a specified threshold. Once the termination criterion is satisfied, the best solution found, either the global best or a satisfactory solution, represents the optimized solution to the given optimization problem.

F. PSO-PID controller

PSO-PID was modelled in Matlab. The following PSO parameters were used

- Member of each individual is K_p, K_i and K_d
- Population size = 50
- w_{max} = 0.9 and w_{min} = 0.4

$$w = w_{\max} + \frac{(w_{\max} - w_{\min})}{\text{iter}_{\max}} \times \text{iter} \qquad (10)$$

  w is the inertial weight
- c1 = c2 = 2

Simulation was done for 50 iterations and PSO showed good convergence rates.

First we plotted the AVR transfer function without PID controller.

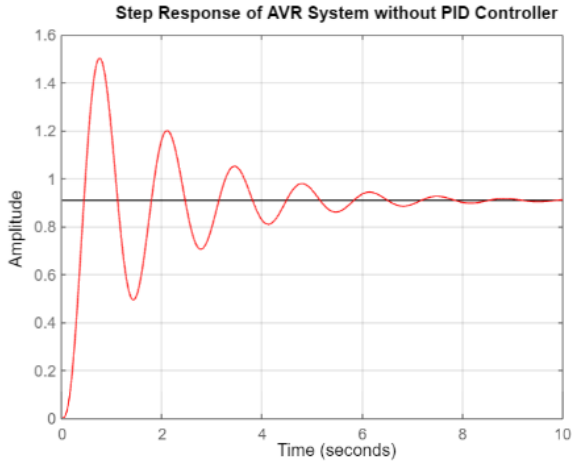

Fig. 2: Step response of AVR system without PID controller

From the above graph the we calculated the transient response

- Steady state error (E_{ss}) = 0.0894
- Maximum overshoot (M_p) = 0.6502
- Settling time (t_s) = 6.9878
- Rising time (t_r) = 0.2613

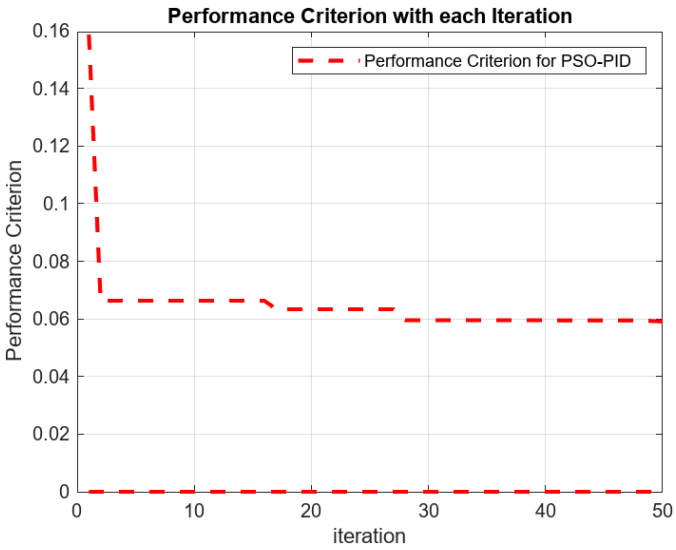

Fig. 3: Performance Criterion of PSO-PID with each iteration

Next, we modelled a PSO-PID controller with the PSO constants as mentioned above. It iterated to minimise the performance criterion.

After convergence of the PSO algorithm, we got the final tuned parameters as:

- $K_p = 0.65315$
- $K_i = 0.55575$
- $K_d = 0.24551$

which gave the performance criterion with $\beta = 1$ of the PSO-PID controller to be $W = 0.0591$

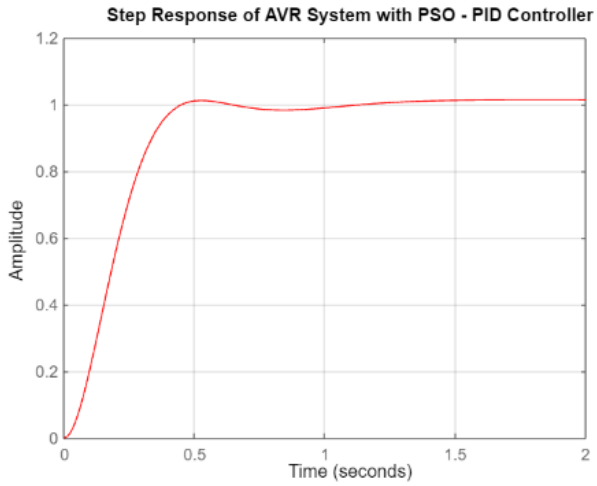

Fig. 4: Step response of AVR system with PSO-PID controller

G. GA-PID controller

We also implemented the GA-PID controller to compare it with PSO-PID. We gave the objective

function to minimize the performance criterion like in the same case as PSO-PID. Hence, It iterated to minimise the performance criterion

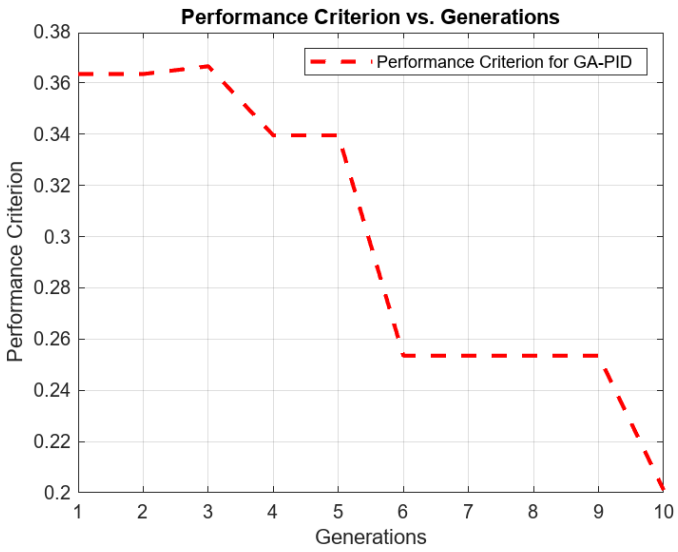

Fig. 5: Performance Criterion of GA-PID with each generation

The values of the tuned parameters obtained were:

- $K_p = 0.84721$
- $K_i = 0.72751$
- $K_d = 0.26716$

Which gave the performance criterion with $\beta = 1$ of the GA-PID controller to be $W = 0.20105$

Shown below is the step response of AVR system with GA-PID controller.

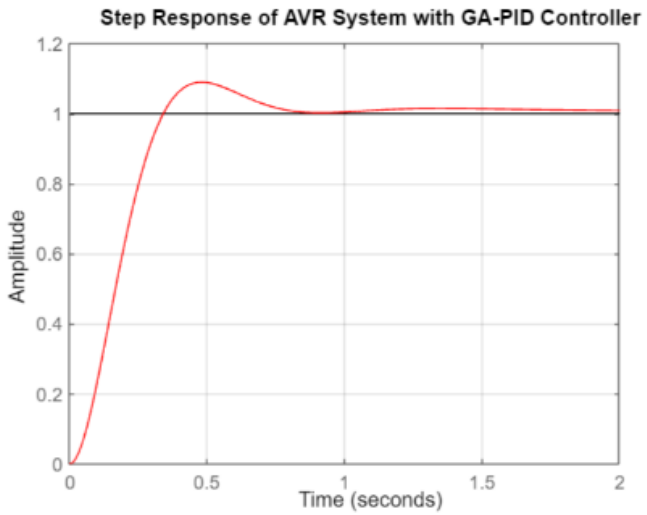

Fig. 6: Step response of AVR system with GA-PID controller

V. Results and Discussion

We observed that PSO-PID controllers demonstrated faster convergence and stability in reaching near-optimal solutions compared to GA-PID controllers. We can see that PSO-PID controller

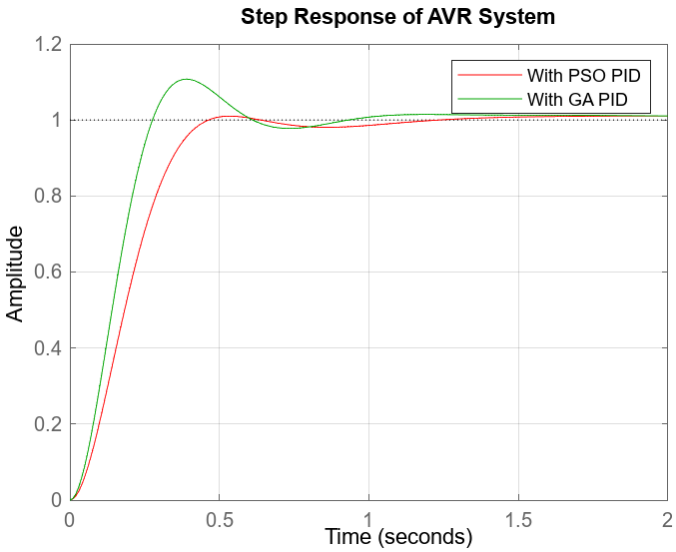

Fig. 7: Step response of AVR system with GA-PID and PSO-PID Controller

could create very perfect step response of the AVR system, indicating that the PSO-PID controller is better than the GA-PID controller. Also as noticed from the performance criteria of PSO-PID controller ($W = 0.0591$) is far less than the Performance criteria of the GA-PID controller ($W = 0.20105$).

VI. Conclusion

We have looked into the strengths and weaknesses of GA-PID and PSO-PID for improving PID controller performance. PSO works well in specific control scenarios because it converges quickly and handles tough situations effectively. On the other hand, GA takes a different approach, exploring various possibilities to find the best solution and adapting to changes in the system. However, GA can be unpredictable and give different results even with the same starting conditions.

Deciding between PSO and GA depends on your control goals, system characteristics, and optimization preferences. To make control systems even better, we can explore combining PSO and GA, integrating machine learning, and using optimized controllers in real-time situations.

VII. Future Perspective

Looking ahead, there are several possible advancements that can be made in PID controller tuning. We can look into the possibility of combining both the PSO and GA algorithms and make it a hybrid algorithm that can leverage the advantages of both algorithms. This may lead to faster convergence and better optimization.

There are different variants and extensions of PSO and GA algorithms with minor improvements. We can also investigate these methods and compare how they enhance the optimization process.

In recent times, there has been a boom for machine learning based approaches. We can also investigate ML based approaches like using reinforcement learning, neural networks and fuzzy design. The advantage of using ML based approach is that it is an adaptive and self tuning PID contoller so these parameters continuously learn and update their parameters based on system feedback.

References

[1] Zwe-Lee Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," in IEEE Transactions on Energy Conversion, vol. 19, no. 2, pp. 384-391, June 2004, doi: 10.1109/TEC.2003.821821. keywords: Particle swarm optimization;Three-term control;Control systems;Optimal control;Design methodology;Pi control;Proportional control;Computational efficiency;Time domain analysis;Genetic algorithms,

[2] Solihin, Mahmud Tack, Lee Moey, Lip Kean. (2011). Tuning of PID Controller Using Particle Swarm Optimization (PSO). Proceeding of the International Conference on Advanced Science, Engineering and Information Technology. 1. 10.18517/ijaseit.1.4.93.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968. keywords: Particle swarm optimization;Birds;Educational institutions;Marine animals;Testing;Humans;Genetic algorithms;Optimization methods;Artificial neural networks;Performance evaluation,