Preprints, 4th IFAC Conference on
Advances in Proportional-Integral-Derivative Control
Almería, Spain | June 12-14, 2024

Genetic PI Controller Tuning to Emulate a Pole Assignment Design ⋆

Marco Paz Ramos * **Axel Busboom** * **Andriy Slobodyan** **

* *Department of Engineering and Management, Munich University of Applied Sciences, Munich, Germany*
** *Business Unit Catalysts, Clariant AG, Munich, Germany*

Abstract: Tuning of proportional-integral (PI) and proportional-integral-derivative (PID) controllers continues to be a current topic, as the control needs in industry are broad and diverse. Although PID controllers have been the predominant controller type for several decades, there are still opportunities to improve the performance of both the installed base and newly deployed controllers. One of the main challenges is a reliable and easy tuning that can be performed by an operator on site. From a computer science point of view, the problem of tuning PID controllers qualifies as a nondeterministic polynomial-time hard problem (NP-hard). Genetic algorithms are a heuristic approach to approximate this type of problem, and the continued growth of computational capacity makes them increasingly more viable. In this work, we present a PID tuning architecture using a genetic algorithm, which incorporates in its fitness function an emulation of pole assignment and implicit cancellation of the additive dynamics of zeros in a closed loop, in addition to reducing discretization losses. The tuning is performed with a paradigm different from the one usually applied in the literature. Rather than simply minimizing the error between the process variable and the setpoint, the error between the process variable and an ideal response curve associated with a desired pole assignment is minimized. This approach provides better control over closed-loop performance.

Keywords: PID, PI, Genetic Algorithms, Pole Assignment.

1. INTRODUCTION

The field of control engineering is very broad and diverse, and new contributions are continuously made to this field. Despite these advances, the de facto standard in the process control industry for single-loop controllers, e.g., for temperature, flow, or level control, is still proportional-integral-derivative (PID) controllers. It is estimated that approximately 90% of the control loops in industry are PID type (Åström and Hägglund, 2001; Knospe, 2006).

Since its inception, probably the most significant challenge in implementing PID controls has been that of parameter tuning. Early operators did not have the theoretical and practical training necessary for accurate tuning, which led to the introduction of simple heuristics such as those proposed by Ziegler and Nichols and many others (O'Dwyer, 2009). Although understanding of the dynamics of PID loops has increased, the practical implementation of PID control often continues to be suboptimal. It is estimated that up to one third of the installed loops are manually tuned and that at least one in four uses the manufacturer's default gain settings (Ender, 1993). Analytical tuning requires mathematical modeling of or data from the plant in either the time or frequency domain. At the higher end of the spectrum is exhaustive or mechanistic mathematical modeling, which requires detailed and significant consideration of the physical and/or chemical laws involved. This approach usually results in complex models that may require custom controllers. Halfway to the simplest heuristic tuning, there is the alternative of designing controllers using black box models, i.e., functional relationships between the inputs and outputs of a system that represent the main characteristics of the system without being based on physical relationships (Zhang, 2010).

Many plants to be controlled by PID controllers can reasonably be approximated as first- or second-order linear models. If such approximate models are available, the calculation of gains by pole assignment is feasible. Beyond the representativeness of the chosen model, there are several reasons why the performance of the practical implementation may differ from that of the theoretical control design. Aspects that may cause discrepancies between design and implementation performance include:

(a) the additive dynamics of zeros when closing the loop;
(b) effects of discretization when using digital controllers (unless this is considered in the design from the beginning);
(c) differences in the algorithm implementation between the design and the actual industrial controller;
(d) discrepancies between digital algorithms and configuration errors when selecting time bases.

Since the tuning task only involves choosing three parameters, it may seem simple. However, like all nonconvex optimization problems, tuning the PID parameters for an

⋆ This work was supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy (StMWi) under Grant DIK0397/03.

accurate and stable closed-loop control becomes an NP-hard problem (Somefun et al., 2021). The complexity of the tuning problem is further increased by the aforementioned discrepancies between theory and implementation.

Genetic algorithms are a widely used optimization tool to approximately solve NP-hard problems (Arabi, 2016; Panchal and Panchal, 2015). The application of genetic algorithms for PI and PID control tuning is significant for several reasons: 1) it is cost-effective in that the same evolutionary optimization framework can be reused for plants of different complexity (Borase et al., 2021), 2) the genetic algorithm can easily assimilate different optimization criteria (Joseph et al., May 2022), 3) the tuning task can be approached as a multi-objective optimization complying with a Pareto front (Guenounou et al., 2012) and 4) it allows circumscribing validity zones for standard structures, as is the case for tuning tables (Visioli, 2001). In the literature on genetic tuning of PID controllers, error criteria such as Integral of the Squared Error(ISE), Integral of the Absolute magnitude of the Error(IAE), or Integral of Time multiplied by Absolute Error (ITAE) (Borase et al., 2021) are frequently used. These have the advantage of directly using the tracking error with respect to the setpoint given during the control. However, a disadvantage is that the desired closed-loop performance cannot be precisely established beyond the error minimization itself. For example, with ITAE, the genetic algorithm will aim to make the control response as fast as possible. In practice, however, other considerations such as actuation saturation must be taken into consideration. Tuning PI controllers by pole assignment can be useful for establishing specific closed-loop responses, but even the additive dynamics of zeros can slightly change the closed-loop response from an originally desired one.

This article presents a proposal to tune PI and PID controllers with genetic algorithms. It applies an optimization criterion to obtain a specific closed-loop response as in tuning by pole assignment, but implicitly mitigates the effects of the additive dynamics of zeros and of discretization.

2. POLE ASSIGNMENT DESIGN

PI tuning by pole assignment is well understood (Aström and Hägglund, 1995). This section deals with the approach to transfer functions discussed in (Paz et al., 2017). Assume we have a first-order plant of the form

$$G_1(s) = \frac{K}{\tau s + 1} \tag{1}$$

where K is the open loop gain and τ is the time constant. Now we close the loop with a PI controller (Aström and Hägglund, 1995) of the form

$$G_c(s) = \frac{K_c T_i s + K_c}{T_i s} \tag{2}$$

where K_c is the proportional gain of the controller and T_i is the integral time. With negative feedback, the following closed-loop transfer function is obtained

$$G_{CL1}(s) = \frac{\frac{KK_c}{T_i\tau}(T_i s + 1)}{s^2 + \frac{1}{\tau}(1 + KK_c s) + \frac{KK_c}{T_i\tau}}, \tag{3}$$

which is similar to the canonical form of a second-order transfer function, but differs by the presence of a zero in the numerator

$$G_{SO}(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}. \tag{4}$$

Here, ξ is the damping coefficient and ω_n is the undamped natural frequency. From the denominator, we can calculate the approximate maximum overshoot M_p when ξ is in the range between 0 and 1

$$M_p \approx e^{\frac{-\pi\xi}{\sqrt{1-\xi^2}}} \tag{5}$$

and the settling time t_s defined within an error band of 1%, (Franklin et al., 1998)

$$t_s \approx \frac{4.6}{\xi\omega_n}. \tag{6}$$

If K_c and T_i are known, we can calculate the settling time t_s and the maximum overshoot M_p for the closed loop from (5) and (6).

To tune the controller gains, we specify a desired closed-loop settling time t_{sd} and maximum overshoot M_{pd}, and solve (5) and (6) for the desired damping coefficient ξ_d and undamped natural frequency ω_{nd}:

$$\xi_d \approx \sqrt{\frac{\ln^2(M_{pd})}{\pi^2 + ln^2(M_{pd})}}, \quad \omega_{nd} \approx \frac{4.6}{\xi_d t_{sd}}. \tag{7}$$

Finally, by comparing the denominators of equations (3) and (4) we get

$$K_c = \frac{2\tau\xi_d\omega_{nd} - 1}{K}, \quad T_i = \frac{KK_c}{\tau(\omega_{nd})^2} \tag{8}$$

for the parameters of the PI controller.

If the process can be approximated as an overdamped second-order system ($\xi > 1$)

$$G_2(s) = \frac{K}{(\tau_1 s + 1)(\tau_2 s + 1)}, \tag{9}$$

when the PI control of equation (2) is applied to the plant of (9), the closed-loop transfer function becomes

$$G_{CL2}(s) = \frac{\frac{K_c K}{\tau_1\tau_2}s + \frac{K_c K}{T_i\tau_1\tau_2}}{s^3 + (\frac{\tau_1+\tau_2}{\tau_1\tau_2})s^2 + (\frac{K_c K+1}{\tau_1\tau_2})s + \frac{K_c K}{T_i\tau_1\tau_2}}. \tag{10}$$

Following the same scheme of equalizing denominators of the controlled plant and the desired plant, it is necessary to add a third pole to the desired denominator of the form

$$D(s) = (s + p_1)(s^2 + 2\xi_d\omega_{nd}s + \omega_{nd}^2). \tag{11}$$

As in the design for the first-order plant, the desired second-order closed-loop component can be constructed by choosing the desired settling time t_{sd} and the desired maximum closed-loop overshoot M_{pd}. Term-by-term matching of (11) and the denominator of (10) yields

$$p_1 = \frac{\tau_1 + \tau_2}{\tau_1\tau_2} - 2\xi_d\omega_{nd}. \tag{12}$$

However, it is not sufficient to arbitrarily choose the value for the desired settling time t_{sd} to construct the second-order component of the desired polynomial as this may

lead to a polynomial $D(s)$ that is not asymptotically stable. To avoid this, the following necessary and sufficient condition for the settling time must be fulfilled (Paz et al., 2017):

$$t_{sd} > \frac{9.2(\tau_1\tau_2)}{\tau_1+\tau_2}. \tag{13}$$

Following the same equalization process as above, the controller gains for this case can be obtained as

$$K_c = \frac{\omega_{nd}^2\tau_1\tau_2 + 2p_1\xi_d\omega_{nd}\tau_1\tau_2 - 1}{K}, \tag{14}$$

$$T_i = \frac{K_cK}{\tau_1\tau_2\omega_{nd}^2 p_1}. \tag{15}$$

3. GENETIC ALGORITHMS

Evolutionary computation is a family of computational optimization techniques inspired by natural evolution (Eiben and Smith, 2015). Examples include genetic algorithms, evolutionary programming, genetic programming, memetic algorithms, or ant systems.

Genetic algorithms are probably the most representative technique of evolutionary computation and have been adopted in the field of automatic control for a long time (Wang et al., 2003). Genetic algorithms are inspired by the exchange of genetic information in evolution in nature. Although optimization techniques based on gradient descent are very popular in machine learning due to their efficiency, in some optimization problems where a balance between exploration and exploitation is required, evolutionary optimizers can be more effective and have multiple advantages: Genetic algorithms are an inherently parallel method (Goldberg, 1988), which can maintain a population of possible solutions at a given time and whose final population is part of the Pareto front (Wei and Söffker, 2015). The use of genetic algorithms is viable when performing exploration can contribute to the search for a better result. Furthermore, genetic algorithms can support cost functions with complex rules and restrictions since they are related to an execution approach rather than a specific calculation. Finally, genetic algorithms are good when the objective function has high modality, i.e., many local optima (Mirjalili, 2019).

The optimization of PID gains in a discrete space can be understood as an NP-hard problem. Depending on the requirements, there may be several acceptable solutions for tuning a controller; i.e., the tuning problem can then be interpreted as a combinatorial one. This makes it appropriate to use a genetic algorithm, which beyond the search for a valley or crest, searches for various combinations that may be viable for the joint satisfaction of a set of restrictions.

4. GENETIC PID WITH EMULATION OF POLE ASSIGNMENT

Genetic algorithms can be applied to the tuning of PID controllers in different ways. In this work, a parametric optimization architecture is proposed in which a genetic algorithm incorporates a black-box model that abstracts the dynamics of a given process. Controller tuning is approached as a combinatorial optimization problem with the objective of choosing the controller parameters in such a way that the desired response of the internal model is achieved. Fig. 1 shows the architecture of the proposed emulative pole assignment scheme for PID controllers.

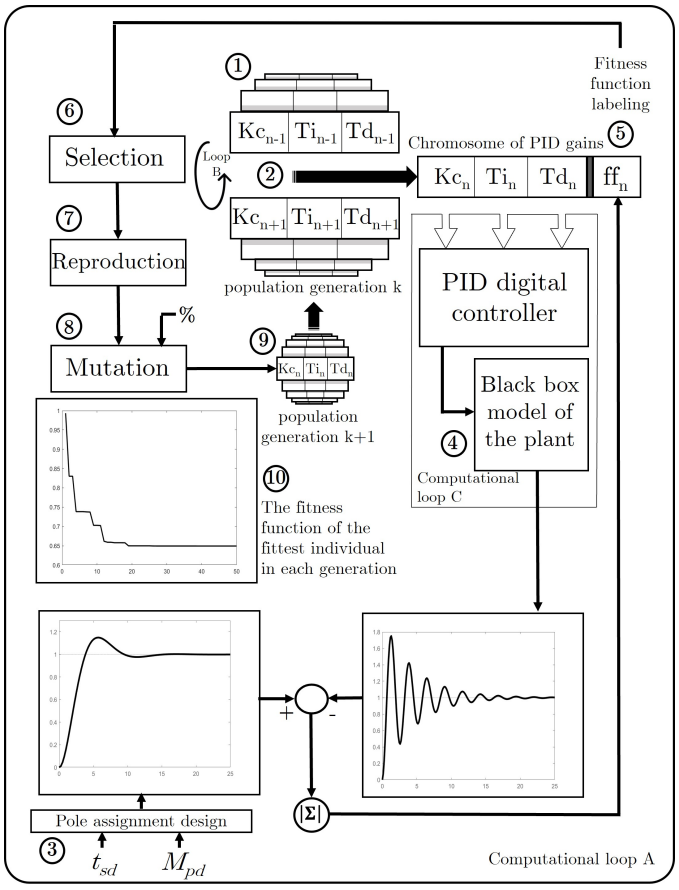

Fig. 1. Architecture of the proposed emulative pole assignment scheme.

The individual steps in the diagram are explained below.

1) A table is generated with random candidate values of the controller gains. In the terminology of evolutionary computation, this first set of potential solutions is called a population and each ordered triad of parameters

$$C_{k,n} = [K_{c_n}, T_{i_n}, T_{d_n}] \tag{16}$$

is called an individual or chromosome where k is the generation and n is the number of individuals in a given population.

2) The parameters of this intial population are applied to a dependent PID in velocity form (Aström and Hägglund, 1995), which is common in industrial digital controllers

$$\begin{aligned} u(t) = u(t-1) + K_c[e(t) - e(t-1)] + \frac{T}{T_i}e(t) \\ + \frac{T_d}{T}(e(t) - 2e(t-1) + e(t-2)) \end{aligned} \tag{17}$$

where u is the control signal, e is the error, T is the sampling period and T_d it is the derivative time. Each element of the population is evaluated sequentially.

3) Using equation (7) from a desired overshoot M_{pd} and desired settling time t_{sd}, a transfer function of the

desired dynamics is calculated. This transfer function is discretized using a sampling period T, then transformed from a differential equation to a difference equation that synthesizes the expected response. From this, the expected step response is vectorized within a time window that contains at least the desired settling period t_{sd}.

4) Each chromosome of the population of generation k is evaluated within a time window that contains the desired settling time t_{sd} and the response of the controlled plant is vectorized. The black box model, depending on the case, corresponds to a discretized model with a zero-order hold (ZOH) of equations (1) or (12), which is implemented as a recursive difference equation (Franklin et al., 1998).

5) In genetic algorithms, the objective function is referred to as the "fitness function". Here it is the absolute error of the standard response of the desired pole assignment and the response obtained with the n^{th} test chromosome

$$ff_n = \sum |y_d - y_g| \tag{18}$$

where ff_n is the fitness function of an individual n, y_d is the vector of the desired output of the controlled plant, and y_g is the output using the n^{th} test chromosome. We have chosen the absolute error as opposed to a squared error, as the squared error tends to focus too much on small time intervals with the largest error, while the absolute error tends to lead to an overall closer approximation of the desired closed-loop response.

6) Once all chromosomes in a population k have been labeled with their corresponding fitness functions, we proceed to a "tournament" (Goldberg and Sastry, 2007; Kramer, 2017), the size of which is determined as a percentage of the population size (10% of the population size is used for this design to promote genetic diversity). Through the tournament, pairs of potential parents are selected to carry out the reproductive process.

7) With the selected parents, a bit-level crossover is performed (Goldberg and Sastry, 2007; Kramer, 2017). A random cutoff point is chosen along the chromosomes, where each of the parents' chromosomes is separated into two parts. The resulting sections are now recombined in such a way that the lower part of one parent joins the upper part of the other to form a place for a child, while the complementary parts form the other. In this case, each pair of parents begets a pair of children, which are partially different from their parents and will become part of the $k + 1^{st}$ population.

8) Once the same number of offspring as parents have been assembled in a new population (to maintain ecological stability), mutations are made in a small percentage of individuals (Goldberg and Sastry, 2007; Kramer, 2017), i.e., a randomly chosen bit of a given chromosome is inverted. Mutation is an important operator to maintain genetic diversity on a generational scale.

9) Finally, the population obtained in the $k+1^{st}$ generation will entirely replace the previous generation.

10) The Holland's Schema Theorem (Holland, 1992) states that segments of the chromosome more fit than average will generate copies on an exponential scale in subsequent generations. Therefore, the fittest individual of each generation will provide a closed-loop performance y_g increasingly similar to the desired y_d and the error will decrease over the generations as far as possible, respecting the closed-loop structural capacity.

5. PI CONTROLLER GENETIC TUNING TESTS

This section presents some examples to compare the performance of the pole assignment method described in Section 2 with the genetic tuning algorithm. We assess how similar the closed-loop temporal response for both methods is to an ideal closed-loop response that meets a desired settling time t_{sd} and a maximum overshoot M_{pd} according to (4) and (11). The difference with respect to the reference curve is quantified by the sum of the absolute errors, described as the fitness function ff_n of equation (18). For all test cases, the genetic algorithm used was configured to compute 50 generations with a population size of 500, a chromosome size equal to 2 (K_c, T_i), and a mutation index of 6%. The resolution of each gene is 53 bits as before the decoding process, integers are used in significand IEEE double precision (IEEE 754-2019). These hyperparameters were chosen heuristically as a compromise between convergence and applicability for the set of cases presented in this section.

Fig. 2 shows an example of the step responses for a first-order plant of the form (1) with an open-loop gain of 1.2 and a time constant of 6.9. For this case, $t_{sd} = 3\tau$ and a $M_{pd} = 0.1$ have been used, so from (8) we get $K_c = 1.722$ and $T_i = 2.1195$. The tuning of the genetic algorithm yields gains $K_c = 0.9766$ and $T_i = 2.8253$. The sampling period is $T = 1$ for both tuning methods. It can be seen that the tracking of the target pole assignment curve is closer with the genetic tuning method.

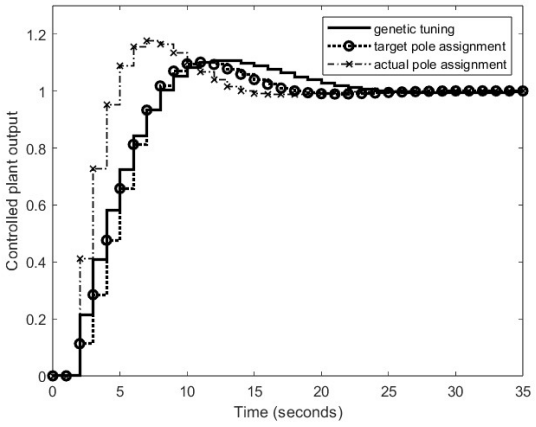

Fig. 2. Discrete PI performance tuned by a genetic algorithm to emulate a non-zero dynamic pole assignment for a first-order plant (ZOH, T=1).

We performed a set of simulations related to this first-order example, where we varied the maximum overshoot M_{pd}, which is a design requirement, and the time constant τ, which can be interpreted as a parametric perturbation or a change in the plant specification. The desired settling time t_{sd}, was kept three times the time constant τ. The sum of the absolute errors is shown in Table 1. It can be seen that the cumulative error is always lower with

genetic tuning than with pole assignment, which shows a better capability of the proposed method to meet the design requirement. On average, the cumulative absolute error of the genetic algorithm was 74.4% lower than with pole assignment for these first-order cases. The * symbol in the table indicates that for the given combination of M_p and t_{sd}, the closed loop of the pole assignment method was unstable. While this problem could be easily fixed by decreasing the sampling period T, the experiment is meant to demonstrate the limits of both tuning methods. It can be seen that the genetic algorithm was able to find a viable solution even in the situations where the conventional pole assignment yielded an unstable solution.

Table 1. Sum of the absolute difference between curves, f, first-order plant (1), $T = 1$, time window length 35.

	M_p=0.05		M_p=0.1		M_p=0.15		M_p=0.2		M_p=0.25	
τ	p.a.	g.t.	p.a.	g.t.	p.a.	g.t.	p.a.	g.t.	p.a.	g.t.
τ=1.15	*	0.096	*	0.071	*	0.098	*	0.332	*	0.173
τ=2.3	1.276	0.239	1.475	0.243	2.640	0.261	14.463	0.292	*	0.303
τ=3.45	1.783	0.404	1.614	0.408	1.547	0.433	1.650	0.438	1.690	0.500
τ=4.6	2.293	0.593	2.012	0.741	1.885	0.688	1.798	0.613	1.814	0.626
τ=5.75	2.805	0.787	2.415	0.932	2.180	0.871	2.053	0.796	2.006	0.767
τ=6.9	3.321	0.921	2.821	1.114	2.539	1.026	2.348	0.981	2.234	0.936
τ=8.05	3.840	1.129	3.240	1.114	2.872	1.246	2.639	1.143	2.474	1.086
f average	2.553	0.596	2.263	0.660	2.277	0.660	4.158	0.656	2.044	0.627

p.a.: pole assignment; g.t.: genetic tuning; *: pole assignment does not work with the chosen T.

Fig. 3 shows the time responses for a second-order plant of the form (9). In this simulation, we chose $K = 1.65$, $\tau_1 = 5$ and $\tau_2 = 14$. We have used a value of $t_{sd} = 75\%$ for the approximate settling time [1] and $M_{pd} = 0.25$, so from (14) and (15) we find $K_c = 2.2887$ and $T_i = 11.885$ for the pole assignment design. The tuning of the genetic algorithm yields $K_c = 1.6310$ and $T_i = 9.999$. In this example, the sampling period is $T = 5$ for both tuning methods. As in the previous case, Fig. 3 shows that the tracking to the target pole assignment curve is closer with the genetic algorithm tuning method.

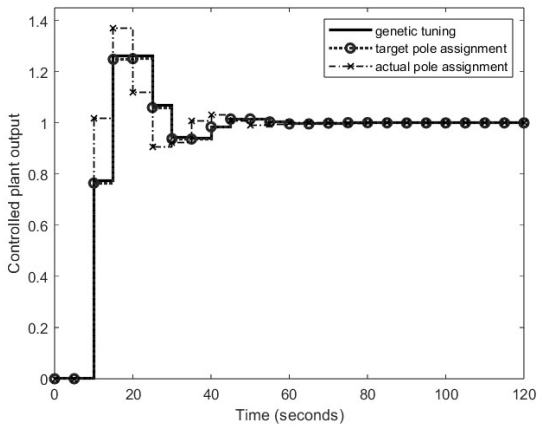

Fig. 3. Discrete PI performance tuned by a Genetic Algorithm to emulate a non-zero dynamic pole assignment for a second-order plant (ZOH, T=5).

As with the first-order example, two parameters were varied. In this case, the variation on the desired maximum overshoot M_{pd} was maintained and τ_2 was chosen as the parameter to be varied. The values of τ_1, K and t_{sd} were maintained from the example above. As in the previous case, it can be seen from Table 2 that the cumulative error is always lower for genetic tuning than with pole assignment (58.4% in average across all cases).

Table 2. Sum of the absolute difference between curves, f, second-order plant (9), $T = 5$, time window length 120.

	M_p = 0.05		M_p0.1		M_p = 0.15		M_p = 0.2		M_p = 0.25	
τ_2	p.a.	g.t.	p.a.	g.t.	p.a.	g.t.	p.a.	g.t.	p.a.	g.t.
τ_2=14.0	0.524	0.498	0.538	0.141	0.602	0.176	0.670	0.084	0.852	0.057
τ_2=16.0	0.730	0.329	0.681	0.203	0.692	0.326	0.786	0.163	0.907	0.119
τ_2=18.0	0.943	0.395	0.854	0.363	0.842	0.395	0.876	0.223	0.946	0.184
τ_2=20.0	1.152	0.556	1.009	0.446	0.946	0.403	0.956	0.325	1.052	0.254
τ_2=22.0	1.354	0.699	1.166	0.576	1.166	0.576	1.080	0.422	1.118	0.339
τ_2=24.0	1.555	0.815	1.322	0.699	1.198	0.601	1.159	0.571	1.173	0.426
τ_2=26.0	1.748	0.945	1.471	0.821	1.332	0.757	1.265	0.593	1.283	0.472
f average	1.144	0.605	1.006	0.464	0.968	0.462	0.970	0.340	1.047	0.264

p.a.: pole assignment; g.t.: genetic tuning.

The similarity between the pole assignment tuning and the desired target curve could be increased by adding a feedforward block for closed-loop zero cancelation, but this is difficult to implement on commercial off-the-shelf controllers. The proposed genetic tuning approach emulating pole assignment with zero cancellation does not require modifications to the standard PID structure.

Note that a direct comparison between our approach and other genetic tuning algorithms (Borase et al., 2021) that minimize the error between the process variable and the setpoint, is difficult. When replacing our error metric (18) by the IAE against the setpoint, both methods yield comparable error values. However, when using the error against the desired step response, the cumulative tracking error of standard genetic tuning methods would be much larger (approximately 200% in the first-order example and 1,000% in the second-order example). However, these differences strongly depend on the desired response curve chosen for each case.

Additional factors may not be considered in the practical deployment of the controller. As mentioned above, deviations due to discretization must be considered during the design stage, particularly when the dynamics of the process are in the same order of magnitude as the sampling period. Other aspects that may influence the practical performance of the tuning include (a) sensor delays, (b) noise, (c) losses due to resolution, (d) actuator saturation, and (e) actuator dynamics, e.g., valve motion. Each of these aspects can change the performance of the control loop from the desired behavior. Three of them can be incorporated directly into the genetic controller tuning scheme: 1) noise as an observable percentage of the steady-state response of the plant, 2) actuator saturation (e.g., valve limits), and 3) actuator dynamics [2].

Fig. 4 shows a simulation of a practical case, taking into account these three aspects. The figure compares the pole assignment method with the genetic algorithm, which considers noise, saturation, and actuator dynamics. The simulation is for a first-order plant with $K = 1.5$ and $\tau = 6.5$; the desired response has a $M_{pd} = 0.1$ and a

[1] A heuristic way to calculate the approximate settling time in overdamped systems is to multiply the sum of both time constants by 3.5.

[2] As a rule of thumb, a value of one second in the time constant of the response of the valve per inch of the valve size is often used in practice.

$t_{sd} = 3.5\tau$. For both controllers, additive noise is simulated with zero mean and standard deviation $1x10^{-3}$, actuator limits of 0 and 1 are considered, and an actuator time constant of 3 is simulated. These effects are taken into account in the training of the genetic algorithm. While the use of anti-windup, which temporarily turns off the accumulation on the integral gain while the actuator is at a limit, is an industry standard, the inclusion of saturation in the genetic optimization aims at taking constraints into account already in the controller design and designing an intrinsically less aggressive controller if necessary. Fig. 4 shows a similar initial slope of the step response due to actuator saturation for both approaches. However, genetic tuning provides weaker oscillations and an overall error 20% lower compared to that of pole assignment, underscoring the point that the algorithm can be readily used to account for non-ideal effects in the plant behavior.

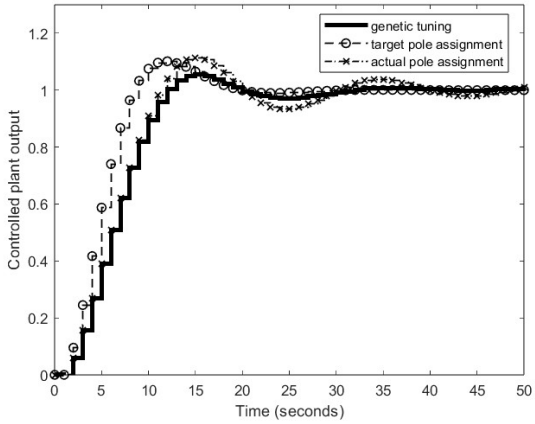

Fig. 4. Comparative case with additive noise, actuator saturation, and actuator dynamics.

6. CONCLUSION

This study presented a proposal to tune PI controllers with genetic algorithms. Instead of a cost function based on the tracking error at the setpoint, we use one based on the minimization of the error between a desired and feasible step response (designed to achieve both a certain settling time and a maximum overshoot) and the actual closed-loop step response. This emulated pole assignment approach yields a closed-loop response that is closer to the target curve than conventional pole assignment. The method is also capable of accounting for practical aspects, such as noise or actuator saturation and dynamics. On the other hand, the method proposed in this work could encounter limitations if the quality of the approximate black box model is not adequate. Finally, the proposed structure can be extended to plants of different types and even incorporate an operator interface that facilitates the intuitive choice of performance criteria.

REFERENCES

Arabi, B.H. (2016). Solving NP-complete problems using genetic algorithms. In *18th Int. Conf. Comput. Model. Simul. (UKSim)*, 43–48. IEEE.

Åström, K.J. and Hägglund, T. (2001). The future of PID control. *Control Eng. Pract.*, 9(11), 1163–1175.

Aström, K. and Hägglund, T. (1995). *PID controllers: Theory, design and tuning*. ISA, Research Triangle Park, NC, 2nd edition.

Borase, R., Maghade, D., Sondkar, S., et al. (2021). A review of PID control, tuning methods and applications. *Int. J. Dynam. Control*, 9, 818—-827.

Eiben, A. and Smith, J. (2015). *Introduction to Evolutionary Computing*. Springer, Berlin, Heidelberg.

Ender, D.B. (1993). Process control performance: Not as good as you think. *Control Eng.*, 40(10), 180–190.

Franklin, G., Powell, J., and Workman, M. (1998). *Digital Control of Dynamic Systems*. Addison Wesley, Menlo Park, CA.

Goldberg, D. (1988). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Boston, MA.

Goldberg, D. and Sastry, K. (2007). *Genetic Algorithms: The Design of Innovation*. Springer, Berlin, Heidelberg, 2nd edition.

Guenounou, O., Dahhou, B., and Athmani, B. (2012). Optimal design of PID controller by multi-objective genetic algorithms. In *Int. Conf. Computer Related Knowledge*, 6p.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA.

IEEE 754-2019 (2019). *754-2019 - IEEE Standard for Floating-Point Arithmetic*.

Joseph, S., Dada, E., Abidemi, A., Oyewola, D., and Khammas, B. (May 2022). Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. *Heliyon*, 8, E09399.

Knospe, C. (2006). PID control. *IEEE Control Syst. Mag.*, 26(1), 30–31.

Kramer, O. (2017). *Genetic Algorithm Essentials*. Springer International Publishing, Cham.

Mirjalili, S. (2019). *Evolutionary algorithms and neural networks: theory and applications*. Springer International Publishing, Cham.

O'Dwyer, A. (2009). *Handbook of PI and PID Tuning Rules*. Imperial College Press, London, 3rd edition.

Panchal, G. and Panchal, D. (2015). Solving NP hard problems using genetic algorithm. *Int. J. Comput. Sci. Inf. Technol.*, 6(2), 1824–1827.

Paz, M., Ramirez, T., Garibo, S., et al. (2017). Adaptive proportional–integral controller using OLE for process control for industrial applications. *Int. J. Adv. Robot. Syst.*, 14(5), 1–11.

Somefun, O.A., Akingbade, K., and Dahunsi, F. (2021). The dilemma of PID tuning. *Annu. Rev. Control*, 52, 65–74.

Visioli, A. (2001). Optimal tuning of pid controllers for integral and unstable processes. In *IEE Proc., Control Theory Appl.*, 180–184. IEE.

Wang, Q., Spronck, P., and Trachth, R. (2003). An overview of genetic algorithms applied to control engineering problems. *Int. Conf. Mach. Learn. Cybern.*, 3, 1651–1656.

Wei, C. and Söffker, D. (2015). Optimization strategy for PID-controller design of amb rotor systems. *IEEE Trans. Control Syst. Technol.*, 24(3), 788–803.

Zhang, P. (2010). *Advanced Industrial Control Technology*. William Andrew Publishing, Norwich, NY.