Journal of Scientific & Industrial Research
Vol. 84, March 2025, pp. 269-277
DOI: 10.56042/jsir.v84i03.8230

Improved Machine Learning based Approach for Autotuning PID Controller using Genetic Algorithms and Parallel Processing

Aghil Ahmadi & Reza Mahboobi Esfanjani*

Department of Electrical Engineering, Sahand University of Technology, Tabriz, Iran

Received 22 January 2024; revised 18 September 2024; accepted 22 December 2024

PID controllers are widely applied in approximately 95% of continuous control systems across process industries, making them a cornerstone of control engineering. Despite their widespread use, these controllers are often inadequately tuned. This study proposes an intelligent adaptive Proportional-Integral-Derivative (PID) controller for managing complex, uncertain processes. To enhance the capabilities of traditional PID controllers, an advanced machine learning approach using Deep Neural Networks (DNNs) is implemented. To optimize the configuration of the neural network and reduce computational load, a Genetic Algorithm (GA)-based structural learning technique is used, combined with parallel computing to accelerate training. Simulation results show that the proposed controller achieves an RMSE of 0.70 in the absence of disturbances, outperforming the Standard PID (0.95 RMSE) and Shallow Neural PID (0.74 RMSE). Under a 20 dB SNR disturbance, the proposed approach maintains robust performance with an RMSE of 0.79, compared to the Standard PID (1.10 RMSE) and Shallow Neural PID (0.86 RMSE). These findings highlight the superiority of the proposed innovatively tuned controller over both standard PID controllers and recently introduced intelligent network-based tuners, particularly in the presence of uncertainties.

Keywords: Adaptive tuning, Deep neural networks, Intelligent control, Parallel computing

Introduction

Many complex processes with multiple control loops facing different types of uncertainty are found in modern industries. Although many efficient control systems have been found thus far, their complexity prevents control engineers from designing and implementing them.[1] Furthermore, enterprises are encouraged to use basic linear controls due to budgetary concerns. However, non-linear real-world processes with complex behavior cannot benefit from these controllers. Combining the recently developed machine learning methods with the previously mentioned basic controllers could be one creative way to address control issues in contemporary process industries. The majority of modern industrial processes behave nonlinearly, which is generally difficult to control.[2] Additionally, linear plants that have a noticeable temporal delay are frequently challenging to manage because time delays cause poor performance and even instability.[3] The need for more sophisticated solutions an extension from embedded systems to cyber-physical systems, whose key attributes are "Intelligence," "Autonomy," and "Self-Learning," is brought about by the need for more intelligent solutions to such issues in the industry.[4] Recently, control systems have seen the use of machine learning techniques that may be useful for uncertain nonlinear systems. For example, an adaptive fuzzy controller was used to overcome position control issues.[5] A feed-forward neural network and a fuzzy PID controller were used to operate a two-tank system.[6] An advanced network was designed to work on a specialized robot.[7] In most industrial applications, the conventional PID controller is commonly used.[8,9] The proportional, integral, and derivative components which take into consideration the error's present value, rate of change, and cumulative error over time are combined to provide the control output of a controller. The aforementioned terms are multiplied by their respective constants—KD for the derivative, KI for the integrated, and KP for the present errors before being combined together. These three constants are adjusted to achieve tuning. The most popular controllers utilized in industrial control procedures across a variety of operating circumstances are PID controllers. Reaching the required tracking, good speed, minimal overshoots and undershoots, and the ability to protect actuators from failures and damages which result in fewer products that are off-spec are just a few of the many impressive

*Author for Correspondence
E-mail: mahboobi@sut.ac.ir

advantages of appropriate tuning. From a theoretical standpoint, when the transformation function of the system is available, it is possible to calculate the controller parameters with the various existing techniques and reach the ideal answer, but considering that most industrial systems are non-linear, The transformation function is rarely accessible, because firstly The transformation function obtained is not completely accurate and it is probably obtained by linearization and approximation methods, and secondly, by changing the loop components in the future, such as changing the actuator (which happens a lot in the industry), the system transformation function will change. And naturally, the controlling coefficients obtained based on the primary transformation function will no longer be valid. Nonetheless, trial-and-error-based tuning techniques are applied in the process industries. Obviously, these techniques are not precise enough and may result in significant hazards as well as a loss of productivity.

PID controllers are easier to install and maintain, but these advantages come at the expense of limited functionality. Only linear systems are well controlled by classic PIDs. PID controllers are often calibrated prior to deployment, so they are unable to account for unforeseen disruptions or adapt to changing circumstances in a way that would enable them to meet the demands of today's industries.[10] Finding the ideal or nearly ideal PID constants with the current traditional tuning techniques is difficult. The following will provide a quick explanation of many traditional methods that were primarily utilized for PID parameter tuning: There have been ways to modify the PID controller's settings in recent years. The conventional methods are Ziegler-Nichols, Chien and T-Sigma. The Ziegler Nichols technique was shown in 1942 and was recognized as one of the easiest and most famous approach for tuning. These methods require a measured step response from the control system and are in fact dependent on the step response. The first two methods have a table to determine the controller coefficients and the third method uses an integration approach for system responses. There are other methods that operate based on Root Lucas analysis, or other methods that look at adjusting the controller coefficients as an optimization problem. Genetic algorithms are typically employed in these situations. Genetic algorithms are one heuristic approach that has proven useful for PID tuning.[11] Common methods of tuning have been reviewed in numerous references.[12] For example, the use of genetic algorithms to design PID controllers in the distillation column process, or genetic algorithms to adjust the said controller to control glucose concentration, or in other works of the PSO algorithm to adjust and validate the control. It has been used on real physical plants such as bioreactors. The fact is that, despite the extensive previous applications, these methods have been used for static adjustment of coefficients that were constant or had slight changes during the process. Artificial Neural Networks (ANNs) have received a lot of attention lately and provide extremely high processing capacity due to their learning and generalization capabilities, as well as their parallel distributed topology. A distributed processor with greater capacity for parallelism is the neural network. In this way, it serves the same purpose as the human brain by storing knowledge throughout training in the form of weighted connections between neurons. To address a problem, an appropriate structure with appropriate weights needs to be identified. It may be conceptualized as the reduction of an error function that is determined by desired and actual outputs of the neural network. Several feed forward neural networks were employed to modify the controller parameters in several contexts.[13,14] But these particular neural network topologies provide highly unique behavior.[15] Therefore, they are not adaptable enough to be effectively adapted for many unpredictable contexts. In a broader study, appropriate neural network topologies are examined for use in process industries, in contrast to competing efforts; our method is more useful because it effectively handles noise and disturbance. A network's topology, number of hidden layers, and associated weights are all significant and critical components that affect how well a network performs. There are some drawbacks to both small and large networks. Networks of very small sizes are difficult to train. On the other hand, both theory[16] and experience[17,18] demonstrate that networks with fewer parameters perform better in terms of generalization, which may be explained by keeping in mind the comparison between curve fitting and neural network training. Neural networks are not frequently utilized for parameter adjustment in practical applications, despite their potential. One of the main reasons is the poor interpretability, i.e., the obtained PID parameters cannot be assessed for their appropriateness. Current neural network-based PID tuning research overlooks

closed-loop stability, an important area of control theory.[19] Given that neural networks reveal a system's black box,[20] Their use in industry for PID tuning is still uncommon.[21] A new machine learning approach is used to adjust PID values live. The most similar work to ours in the literature tunes the PID controller using a basic neural network without discussing the neural network's architecture.[10] This work explicitly discusses structural learning and suggests the potential use of genetic algorithms as a future research direction. Unlike the current study, which employs a basic shallow network, A deep network is utilized in the proposed structure. To enhance speed and efficiency, Genetic algorithms and parallel processing are innovatively incorporated in an optimized manner. This approach effectively manages nonlinearity and uncertainties. The key innovation of our approach lies in its unique structural optimization, enabling the simultaneous identification of optimal weights and network structure for the deep network in consideration. In the remainder: First, the system with a tuner and deep neural network is presented, along with its beneficial characteristics for our study and more details are provided next. Comparative results and discussions are presented in the following part, and finally, the concluding summarizes the paper.

Proposed ANN Structure for PID Tuning

In this work, the traditional PID controller is enhanced to be more intelligent and accurate while maintaining its basic structure to align with current industrial process control applications. The selection of input and output types and quantities represents the initial stage for configuring the neural network. As shown in Fig. 1, the controller parameters are taken as the three outputs of the network (KP, KI, and KD), with the inputs being the closed-loop control system output (y) and control error (e). This approach employs a network to estimate the PID controller parameters.

Important considerations are the neural network's ideal architecture and size. Along with the online computation and adjustment durations, the first offline training is another issue that must be considered. On larger networks, these times inevitably rise. And providing a suitable solution in this regard (obtaining the optimal network topology, which naturally involves removing some connections during structure learning, while using parallel processing with an efficient creative structure) is an important part of this work, which will be discussed in detail.

Proposed Method

In order for the presented method to be applicable on an industrial scale, all important and influential issues have been considered in this work. As explained in the part before, the considered neural network that acts as a regulator of the controller coefficients has two inputs and three outputs. The deep neural networks employed in this work are limited to two hidden layers, each containing four neurons, to ensure low-hardware online computation while simultaneously taking advantage of deep network advantages (Fig. 2).

Reusing features that arise repeatedly is made possible by deep networks' hierarchical nature of organizing and storing information. As a result, they serve a more precise purpose than a surface-level one.[22] This type of deep network memorization capability greatly aids in the management of uncertainties in our problem. To ensure acceptable performance before initiating control, the neural network is trained offline, with both its weights and structure optimized. During operation, the deep network must be trained efficiently at each step. An ANN's success greatly depends on the parameters that are chosen. Therefore, it is crucial to focus more on accurately determining them. The number of hidden layers and neurons in each layer are the primary components of neural architecture. Considering the

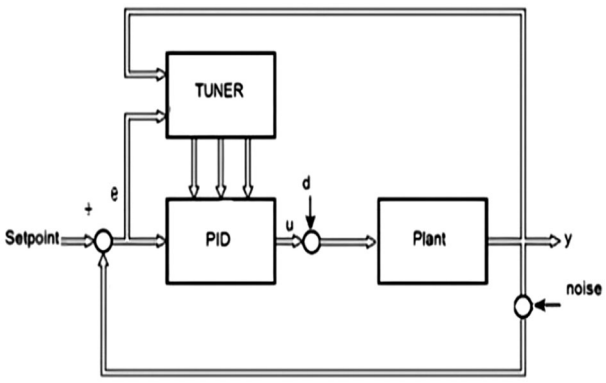

Fig. 1 — The control structure with tuner

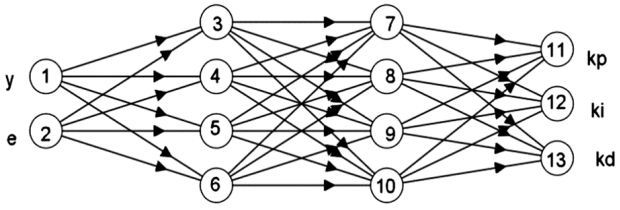

Fig. 2 — Employed deep neural network

given problem and the number of its parameters, The structure of input and output layers can be determined. Therefore, the main challenge is determining the hidden layer structure, and weights are also assigned to network connections.

Genetic Algorithms for Learning and Optimization

Similar to other optimization techniques like PSO, Genetic Algorithms (GAs) are adaptable for search and optimization problems. First introduced by J. Holland and based on Darwin's theory of evolution, GAs are evolutionary algorithms that rely on mutation and selection. The process begins with an initial population—here, the network's initial weights and structure—where connections with zero weights are eliminated. This initial population represents potential solutions within a defined search space, evaluated using a fitness function to identify the best candidates. In the proposed structure (Fig. 3), weights represent both the network's structure and connections, enabling simplified representation through a single array without separating matrices or converting values to binary. Using real-valued weights directly facilitates efficient coding and eliminates the need to differentiate between structure and weights, streamlining the optimization process.

There are 13 neurons in our network: the first six have four connections, the following four have three connections, and the final three provide the outputs. Thus, the network can be shown with a total of 36 weights, which are placed in the array that serves as a chromosome (according to GA format) in this instance (Fig. 3). The subscripts of W are the numbers of the source and destination neurons of the corresponding connection. Therefore, an array of length 36 units is used to present the weights and structure of the network. As mentioned before, the array's weights determine both the weight and the structure for us; a zero indicates that the corresponding connection is absent, and a number other than zero indicates that the connection is present in the network with the weight equals that number. The given number also represents the weight of the connection. As a result, every array or chromosome serves as a network structure. If various population sizes are taken into account, the starting population can be displayed using a matrix like the one below:

W_{13}	W_{14}	W_{15}	W_{16}	...	W_{913}	W_{1011}	W_{1012}	W_{1013}

Fig. 3 — Array for present network

$$U\ (30 \times 36) = (Wmn) \quad \ldots (1)$$

There are 36 columns and 30 rows in this matrix. The row indicates the population size, and the column shows the counting of values used to display each candidate for a network structure. A starting configuration, as shown in Table 1, is needed to conduct appropriate testing. In the fully connected model for Multi Layer Perceptron (MLP), an initial network with 36 connections is started. Thirty initial populations are then examined, with each population representing a network that might solve the issue. An algorithm is initiated based on this preliminary data to identify the optimal structure with the least amount of network error. Taking into account the table containing the starting settings, experiments are run, and the genetic method is adhered to.

Experiments were initially conducted for a population size of 20, and the process was then repeated for a population size of 30, with different findings obtained in each case. Both stages underwent multiple iterations (10, 100, 500, 1000, and 2000). The computation time and error (RMSE) for each test were depicted in Table 2.

This table can be distinguished into three important parts. First, periodic variation is observed in the initial steps of the procedure. In the second step, a decrease in error is evident, and finally, a saturation state is reached. At the end, the optimum weights for the network are determined based on the minimum error. Connections with weights equal to or near zero are ignored. Using innovative GA-based procedures, the network is initially trained for both its weights and structure.

Another important challenge around neural networks is speeding up calculations and reducing computation time. The first solution involves the use of supercomputers, but these systems present challenges, including high costs and maintenance issues, as well as limited accessibility. Instead, an optimized parallel processing technique is employed, which will be explored in detail with supporting figures in the discussion section. The network is intended to be used as a tuner for PID tuning during

Table 1 — Initial setting of Neural Network (NN)

NN configuration parameters	Initial configuration values
ANN Type	Deep (MLP)
Number of initial neurons	13
Number of connections	36
Chromosome size	36
Population size	30
Neurons in hidden layers	8
Training patterns	32

Table 2 — Results for different Iterations

No. of Iterations	RMSE Popsize = 20	CPU time(s) Popsize = 20	RMSE Popsize = 30	CPU time(s) Popsize = 30
10	1.27	0.08	1.50	0.15
100	1.03	0.61	1.44	0.68
500	0.93	2.31	1.31	3.10
1000	0.85	4.41	1.36	6.23
2000	0.90	8.89	1.08	12.04

an online process. The initial offline setting serves as a starting point for control but is insufficient for addressing changing conditions and uncertainties. Therefore, adaptive online tuning is required.

Online Training

The popular method of training is back-propagation.[23] In this case, an unsupervised scenario is encountered. Unlike supervised techniques, where desired outputs are available, the ideal values—namely, the PID parameters to be tuned—are unknown. In supervised learning, network training relies on minimizing the difference between ideal and actual outputs. However, in this situation, neural network training becomes a nonlinear optimization issue.

To modify weights, deep learning typically employs end-to-end optimization; however, the Levenberg-Marquardt numerical technique (proper for small networks) is applied. The tuner's role is to identify appropriate controller parameters, guiding the output toward the set-point, essentially solving a tracking control problem. Consequently, network training focuses on minimizing tracking errors, with the objective function prioritizing this error reduction. Using the Jacobian, the impact of individual weights on the tracking error is assessed, and adjustments are made to reduce error levels. In the context of structural training, connections that result in reduced error when removed are pruned, and the error caused by deleting a connection is evaluated. As a result, a tuner is specifically designed with weights and topology optimized for error reduction. Leveraging the network's dynamic properties enables effective adaptation to unknown uncertainties, enhancing response speed and minimizing network size. Initially, the network structure is selected based on application requirements, with the PID controller trained offline using traditional methods. Structural and weight changes are made for fine-tuning during operation. The system's error (RMSE) is minimized as the objective function, and weights are adjusted at each training step to reduce the control error:

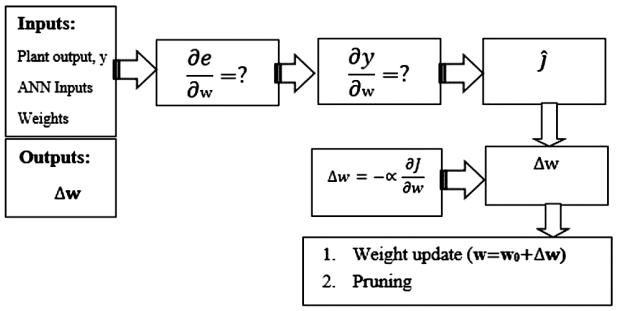

Fig. 4 — Weight and structure updating procedure

$$\Delta w = w_1 - w_0 = -\propto \frac{\partial J}{\partial w} \quad \text{... (2)}$$

where, α is the learning rate. Its optimal quantity depends on the specific problem and it is determined through a combination of empirical experimentation and established methodologies to ensure optimal convergence. Our goal is to reduce the error. Partial derivatives can be used as plant output instead of error.[24] Thus, updating is based on Eq. (2):[10]

$$\hat{j} = \frac{(e_1 - e_0)}{\Delta w} = \frac{((y_* - y1) - (y_* - y0))}{\Delta w} = \frac{(y0 - y1)}{\Delta w} \quad \text{... (3)}$$

where, y is output and y_* is the set point. Therefore:

$$\hat{j} = \frac{\Delta y}{\varepsilon(w)} \; \varepsilon = max(1, w)\sqrt{\varepsilon_{min}} \quad \text{... (4)}$$

where, ĵ is the partial derivate of error to the weights and ε is the machine precision. Due to the lack of ideal training data, an end-to-end gradient descent-based online training procedure, as illustrated in Fig. 4, is implemented. This approach enables the network to adaptively fine-tune the controller parameters in real-time to minimize control error effectively. The training process iteratively evaluates each connection within the neural network by measuring its contribution to the overall system performance. Specifically, each connection is assessed based on the increase in control error when that connection is removed, ensuring that only the most significant connections are retained. By prioritizing critical connections, the network achieves optimal parameter

adjustments while maintaining computational efficiency. This strategy not only enhances the adaptability of the control system but also ensures robustness against disturbances and noise in real-world applications.

Data about the variable neural network is compiled into a table to apply the training method. The input weights, intermediate connection weights, and bias weights constitute the three main parts of network information. Together, these weights are also present the structural configuration. When a connection is removed and its corresponding weight is set to zero, the network output and the resulting control error are recalculated. Subsequently, a specific matrix, referred to as the evaluation table, is used to evaluate network connections based on their error contribution ratings. Every network connection is assessed at each stage according to the control mistakes that arise. Specifically, the error impact of omitting each connection is calculated, and those with minimal error contribution are identified by sorting them accordingly.

Comparative Simulation and Discussion

Proposed Parallel Processing

As mentioned earlier, the discussion of training time is a crucial issue regarding neural networks. In this context, parallel processing using MATLAB was employed (Fig. 5).

With a suitable setting, A study was performed to illustrate the practicality of this technique. The experiment was performed using different numbers of processor cores in an advanced research center, and the results were compared. A typical training problem was tested using a varying number of processors with shared memory. The processing load was divided among the processors, with each processor handling a portion of the load and subsequently providing the results to the manager system. The speedup ratio for different numbers of processors is shown in Fig. 6. According to Amdahl's law, achieving a speedup equal to n with n processors is not possible, but efforts were made to minimize the calculation time as much as possible.

Simulation and Test

To assess the neural PID controller's performance, two common control problems — a two-tank system and a system with significant time delay are used. Each system presents a distinct control challenge and

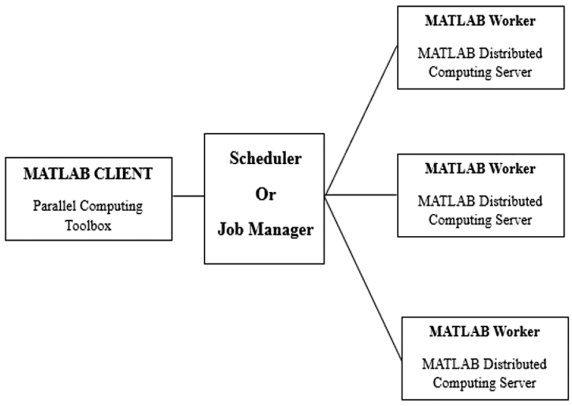

Fig. 5 — MATLAB distributed computing

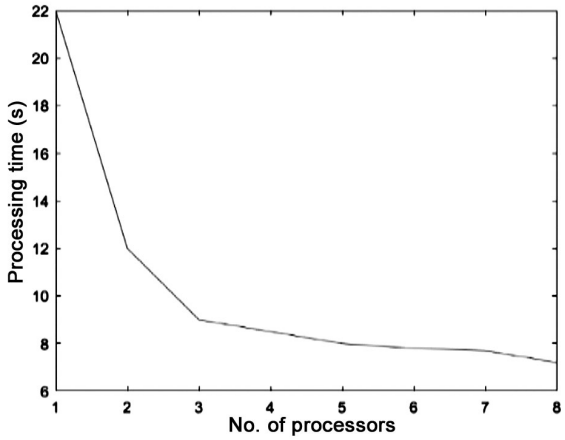

Fig. 6 — Speedup ratio for several processors

is tested both with and without added noise. Gaussian white noise, with a Signal-to-Noise Ratio (SNR) of 20 dB, is applied to the system output to simulate the effect of noisy sensor measurements. To further evaluate the controller's resilience, An appropriate disturbance is introduced into each system. A disturbance is an unexpected and undesired input to the system that increases system error. These disturbances vary by system type and may result from external factors or internal system malfunctions. The disturbance magnitude for each system is gradually increased from zero until only one control strategy can stabilize the system. Disturbances of this critical magnitude are then used in the experiments. Importantly, the neural PID controller's training data does not include these disruptions. To guarantee statistical significance, every experiment is conducted over a hundred separate runs. The controller output is updated every 0.1 seconds to simulate the discrete sampling times of real-world conditions. This intervention time represents the limitations of physical actuators, which cannot adjust instantly. The

solver runs iteratively for each 0.1-second interval, using the results of the previous step as initial conditions, while the controller output u remains constant throughout each 0.1-second window.

Two-tank System

Aksu and Coban[25] explain the differential equations of a nonlinear two-tank system, as shown in Fig. 7. In this two-tank system, the control of water levels is crucial to maintain stability and achieve desired outcomes in applications ranging from chemical processing to water treatment plants. The water level in the second tank, acting as the controlled variable, must be adjusted based on the input provided by the pump, which is the manipulated variable. In such a setup, the control strategy focuses on ensuring that the water level remains within a target range, despite external disturbances or internal dynamics. A PID controller is commonly used due to its ability to handle system delays, maintain steady-state error correction, and adjust to varying load conditions. To do effective control, advanced tuning approaches are often applied to the PID parameters. These include optimization techniques, like genetic algorithms or particle swarm optimization, and machine learning approaches, which enable the controller to adaptively adjust to the system's dynamics, resulting in higher efficiency and stability. Such strategies become especially valuable when dealing with complex dynamics or unpredictable disturbances, which are typical in real-world scenarios. By fine-tuning the PID settings, operators can ensure reliable and optimal performance across various industrial applications, making two-tank systems an invaluable benchmark for testing and advancing control strategies.

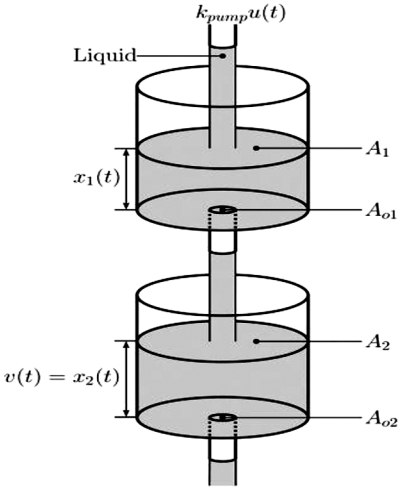

Fig. 7 — Two-Tank system

Between t = 0s and t = 10s, the two-tank system is continually perturbed to assess the robustness of the contrasted control techniques. The controller output is reduced to zero as a form of disturbance, simulating a scenario where the water supply is depleted.

System with time Delay

The second system exhibits a significant temporal delay and is classified as a first-order Linear Time-Invariant (LTI) system. Time delay, a critical aspect in control theory, is frequently overlooked during the design of controllers.[26] Delays can lead to system instability and reduced performance. These delays may be caused by sensor lags, actuator response times, or other physical limitations, preventing the PID controller from responding promptly to changes. This can result in oscillations, overshoot, or persistent errors, ultimately reducing the efficiency of the system. The results of all experiments are presented in Table 3, demonstrating the impact of various delay conditions on system stability and performance.

The RMSE results for three conventional PID tuning methods — Standard PID, Shallow Neural PID[10], and Reinforcement Learning[27] are presented in Table 3 along with the Proposed Deep Approach under varying disturbance and SNR (Signal-to-Noise Ratio) conditions. The results clearly demonstrate the superiority of the Proposed Deep Approach.

Results Analysis

1. No Disturbance, Infinite SNR:
   - Under baseline conditions, the Proposed Deep Approach achieves the best performance with an RMSE of 0.70, outperforming both Standard PID (0.95 RMSE) and Shallow Neural PID (0.74 RMSE).

Table 3 — RMSE on test data over 100 runs

	Two tank system				LTI system with input delay			
Disturbance	—	—	✓	✓	—	—	✓	✓
Signal-to-Noise Ratio	—	**20dB**	—	**20dB**	—	20dB	—	20dB
Standard PID	0.95	0.99	1.00	1.10	0.22	0.23	0.36	0.38
Shallow Neural PID	0.74	0.86	0.84	1.00	0.13	0.15	0.26	0.28
Proposed Deep Approach	0.70	0.79	0.75	0.90	0.09	0.11	0.17	0.17
Reinforcement Learning	0.72	0.80	0.78	0.92	0.11	0.13	0.20	0.21

- This highlights the superior accuracy and efficiency of the Proposed Deep Approach, even in ideal conditions.

2. Presence of Noise (20dB) and Disturbance (Two-Tank System):

- The Proposed Deep Approach maintains its superior performance with an RMSE of 0.79, which is notably lower than that of Standard PID (0.99 RMSE) and Shallow Neural PID (0.86 RMSE).
- These results underscore the robustness and adaptability of the Proposed Deep Approach to disturbances and low SNR environments.

3. LTI System with Input Delay:

- Under challenging conditions involving input delay and noise (20dB), the Proposed Deep Approach again achieves the lowest RMSE values (0.11–0.17), significantly outperforming the Standard PID (0.22–0.38 RMSE) and Shallow Neural PID (0.15–0.28 RMSE).
- The method's ability to handle delays and noisy environments highlights its practical applicability to real-world systems.

Key Insights

The Proposed Deep Approach consistently demonstrates:

- Lower RMSE across all tested scenarios, indicating superior control accuracy.
- Robustness to disturbances, noise, and input delays.
- Adaptability to dynamic and complex systems, which traditional methods fail to achieve.

In conclusion, the results in Table 3 confirm that the Proposed Deep Approach significantly outperforms conventional methods, making it an ideal solution for control systems under various challenging conditions.

The PID coefficients presented in Table 4 are dynamically tuned for a two-tank system using our structurally optimized neural network-based tuner. This method leverages neural networks enhanced by genetic algorithm optimization to ensure real-time adjustments that improve adaptability and control precision.

Table 4 — PID parameters for Two-tank system

Time (s)	K_p	K_i	K_d
0	4.30	0.55	0.20
2	3.75	0.60	0.18
4	4.00	0.58	0.19
6	3.10	0.65	0.18
8	5.45	0.72	0.17
10	4.60	0.70	0.19

The results indicate that the parameters Kp, Ki, and Kd are adjusted over time to address changing system dynamics, disturbances, and noise. Specifically, between 6 to 8 seconds, where noise and disturbances become more pronounced, the tuner increases Ki and Kp values to reduce steady-state error and improve system responsiveness. This highlights the tuner's ability to adapt quickly to challenging conditions while maintaining overall stability and performance. Such adaptability makes the proposed approach highly effective for real-world applications where system dynamics are constantly changing.

Conclusions

This study presents an advanced approach for PID parameter tuning using deep neural networks (DNNs), optimized with genetic algorithms (GAs) and enhanced through parallel processing. The proposed controller was evaluated on two benchmark systems: a two-tank system and an LTI system with input delay. Experimental results demonstrate that the proposed method outperforms conventional PID tuning techniques and shallow network-based approaches, particularly under challenging conditions involving disturbances, uncertainties, and sensor noise. The findings highlight the proposed controller's effectiveness in improving adaptability, robustness, and performance, making it highly suitable for industrial applications such as refineries and petrochemical plants, where precise control is essential. This research underscores the potential of machine learning to enhance classical control systems. Future work could explore the integration of advanced optimization algorithms or hybrid machine learning techniques to further improve control accuracy. Additionally, ensuring real-time stability during online adaptation remains a critical direction for further investigation.

References

1 Pistikopoulos E N, Povoa A B, Lee J H, Misener R, Mitsos A, Reklaitis G V, Venkatasubramanian V & Gani R, A review of process systems engineering advances, *Comput Chem Eng*, **147** (2021) 107252, https://doi.org/10.1016/j.compchemeng.2021.107252.

2 Iqbal J, Ullah M, Ghani Khan S, Khelifa B & Cukovic S, Dynamic modeling and control of robotic systems: A review, *Nonlinear Eng*, **8** (2017), https://doi.org/10.1515/nleng-2016-0077.

3 Espitia N & Perruquetti W, Predictor-feedback prescribed-time stabilization of LTI systems with input delay, *IEEE Trans Autom Control*, **67(6)** (2022) 2784–2799, https://doi.org/ 10.1109/TAC.2021.3093527.

4 Siepmann D & Graef N, Industrie 4.0 – grundlagen und gesamtzusammenhang, in *Einführung und Umsetzung von Industrie 4.0* (Springer) 2016, 17–82, https://doi.org/10.1007/978-3-662-48505-7_2.

5 Vijay M & Jena D, PSO-based neuro fuzzy sliding mode control for a robot manipulator, *J Electr Syst Inf Technol*, **4(1)** (2017) 243–256, https://doi.org/10.1016/j.jesit. 2016. 08.006.

6 Xiao-kan W, Zhong-liang S, Wanglei & Dong-qing F, Design and research based on fuzzy PID-parameters self-tuning controller with MATLAB, *2008 Int Conf Adv Comput Theory Eng* (IEEE) 2008, 996–999, https://doi.org/10.1109/ICACTE.2008.9.

7 Chen Z, Liu Y, He W, Qiao H & Ji H, Adaptive neural network based trajectory tracking control for a nonholonomic wheeled mobile robot with velocity constraints, *IEEE Trans Ind Electron*, **68(6)** (2020), 5057–5067, https://doi.org/10.1109/TIE.2020.2989711.

8 Åström K J & Hägglund T, The future of PID control, *Control Eng Pract*, **9(11)** (2001) 1163–1175, https://doi.org/10.1016/S0967-0661(01)00062-4.

9 Chun-Tang C, Nana S, Juing-Shian C & Chi-Jo W, An optimal fuzzy PID controller design based on conventional PID control and nonlinear factors, *Appl Sci*, **9(6)** (2019), https://doi.org/10.3390/app9061224.

10 Guenther J, *Machine Intelligence for Adaptable Closed-Loop and Open-Loop Production Engineering Systems*, Doctoral Dissertation, Technische Universität München (2018).

11 Slavov T & Roeva O, Application of genetic algorithm to tuning a PID controller for glucose concentration control, *WSEAS Trans Syst* 11(7) (2012) 223–233, Retrieved from academia.edu.

12 Maghade D K, Sondkar S Y & Pawar S N, A review of PID control tuning methods and applications, *Int J Dyn Control*, **9** (2021) 818–827.

13 Rubio J J, Zhang L, Lughofer E, Panuncio Cruz P, Alsaedi A & Hayat T, Modeling and control with neural networks for a magnetic levitation system, *Neurocomput*, **227** (2017) 113–121, https://doi.org/10.1016/j.neucom.2016.09.101.

14 Rubio J J, Discrete time control based on neural networks for pendulums, *Appl Soft Comput*, **68** (2018) 821–832, https://doi.org/10.1016/j.asoc.2017.04.056.

15 Bergstra J, Yamins D & Cox D, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, *30th Int Conf Mach Learn*, **28(1)** (2013) 115–123.

16 Baum E B & Haussler D, What size net gives valid generalization?, *Neural Comput*, **1(1)** (1989) 151–160, https://doi.org/10.1162/neco.1989.1.1.151.

17 Denker J, Schwartz D, Wittner B, Solla S, Howard R, Jackel L & Hopfield J, Large automatic learning rule extraction and generalization, *Complex Syst*, **1** (1987) 877–922.

18 Chauvin Y, Generalization performance of overtrained backpropagation networks, in *Neural Networks* (Elsevier) 1990, 45–55.

19 Glasmachers T, Limits of end-to-end learning, *Proc Asian Conf Mach Learn*, **17** (2017) 17–32.

20 Yang Z, Zhang A & Sudjianto A, Enhancing explainability of neural networks through architecture constraints, *arXiv preprint* (2019) arXiv:1901.03838v2.

21 Zhao Z Y, Tomizuka M & Isaka S, Fuzzy gain scheduling of PID controllers, *IEEE Trans Syst Man Cybern*, **23(5)** (1993) 1392–1398, https://doi.org/10.1109/21.260670.

22 Lee J H, Shin J & Realff M J, Machine learning overview of the recent progresses and implications for the process systems engineering field, *Comput Chem Eng*, **114** (2018) 111–121, https://doi.org/10.1016/j.compchemeng.2017.10.008.

23 Chauvin Y & Rumelhart D E (Eds.), *Backpropagation: theory, architectures and applications* (Lawrence Erlbaum Assoc), 1995.

24 De Jesus O & Hagan M T, Back propagation algorithms for a broad class of dynamic networks, *IEEE Trans Neural Netw*, **18(1)** (2007) 14–27.

25 Aksu U & Coban R, Sliding mode PI and backstepping control design for a nonlinear coupled-tank system, *Int J Robust Nonlinear Control*, **29(5)** (2019) 45–61.

26 Zhang Y, Li J & Wei X, Fractional-order PID controller for first-order plus time-delay systems with model predictive control, *IEEE Trans Syst Man Cybern*, **52(8)** (2022) 1334–1342.

27 Liu Z, Zhang X & Wang Z, Reinforcement learning tuned PI controller for two tank interacting hybrid system, *IEEE Trans Ind Electron*, **68(12)** (2021) 11087–11096, https://doi.org/10.1109/TIE.2021.3061189.