Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6, 7252-7272
2024
Publisher: Learning Gate
DOI: 10.55214/25768484.v8i6.3576

System identification-based tuning of PID algorithm parameters by using PSO and genetic algorithms

Bejad Al-Harbi[1], Abdenour Bounsiar[2*], Abdulelah Algosaibi[3]
[1,2,3]College of Computer Sciences and Information Technology, Al Ahsa, 31982, King Faisal University, Al Ahsa, Saudi Arabia; bejad111@gmail.com (B.A.H.) abounsiar@kfu.edu.sa (A.B.) aaalgosaibi@kfu.edu.sa (A.A.).

Abstract: The PID control algorithm is the most used industrial control method. This is due to its simplicity, ease of use, and because it yields stable results. PID parameters vary for each controller; therefore, finding the optimal parameters is crucial for obtaining good results. However, tuning PID parameters for complex systems is not a trivial task and is also somewhat difficult as conventional techniques rely on time and effort manual tuning or rely on simplified statistical estimation techniques of the system's model yielding sub- optimal results. In this project, we propose to use machine learning for PID parameter tuning on proprietary historical time series operating process control data. The data is processed with the help of computational and machine-learning techniques to better identify the process model and predict the optimal PID parameters. The research methodology consists of three main steps. First, process-model identification is done by using a Radial Basis Function (RBF) neural network. Secondly, Particle Swarm Optimization (PSO) hybrid with Genetic Algorithms (GA) is used for finding the optimal PID parameters. The PID values predicted by PSO, will be fed to GA optimization process as an initial starting point. The final step consists of integrating the identified process model with the PID optimization algorithm in a computer-based simulation environment (Simulink). The experimental simulations are done for various study cases. Results showed that the predicted PID parameters error rate and standard deviation for PID control and process are decreased, which enhances the process controlling and stability.

Keywords: *Control systems, Genetic algorithms, Particle swarm optimization, PID tuning, Radial basis function neural networks, System identification.*

1. Introduction

A Proportional–integral–derivative (PID) controller is a feedback-based control loop mechanism used in industrial control systems and other application domains requiring continuous control of changing system parameters.

The usage of control systems has increased during the last decades and has become a fundamental part of most production environments, as automation has been adopted in almost every production environment. Despite recent advances in the control industry, the PID remains the most used controller. Control complexity increases with complicated systems with non-linear characteristics that cannot be represented using linear models.

The accuracy of PID controllers is bound to the accurate choice of its parameters. Obtaining the optimal PID parameters is one of the big issues in the systems control field [14]. Using predefined parameters is impossible for nonlinear process control for different operating circumstances. Also, finding the mathematical model for each process is a difficult task due to the complexity of the process and the lack of knowledge about the Physics and the Maths of the internal and external system reactions and interactions [6].

Nowadays, tuning PID parameters requires extensive time and effort because it is done either manually on trial and error or by using estimation methods. These two approaches are time- consuming

* Correspondence: aaalgosaibi@kfu.edu.sa

and generally yield sub-optimal results [7,8]. A major reason for the reduced efficiency of the parameter estimation methods is that they rely on estimating the system model for tuning PID parameters. However, industrial systems nowadays are generally distributed making the interaction with different external environments highly complicated and difficult to model. Moreover, these systems involve many interdisciplinary processes such as electrical, mechanical, hydraulic, and chemical, with complicated interactions. As a result, estimation approaches rely on simplified and sub-optimal system models which will affect the PID parameters tuning reliability [6].

Encouraged by the success of artificial intelligence methods in solving many complex non-linear engineering problems such as Image Processing, Natural Language Processing, and Stock Index Prediction [34], in this project we proposed to use Machine Learning (ML) to develop the prediction model based on operational data to find the optimal PID controller parameters.

ML algorithms solve this problem by predicting the best PID parameters that should work well for specific measurements of current system status. Control system optimization is not possible in real production lines for many reasons such as production loss and safety of the human and machines [10,11]. For this, A historical operating dataset containing a set of points, controller outputs, and process variables is used to help learn the correct behavior of the control process for a specific system, instead of finding a mathematical model that describes the behavior of the system to be controlled. We will use the RBF neural network to do system identification. The learned controller behavior (prediction model) will then be used to infer new PID parameters for the new system status.

The remaining of the report is organized as follows. The research methodology is then explained in Section 2. Data Collection is explained in Section 3. Section 4 provides a detailed description of the proposed solution, and the experimental study is offered in Section 5. Finally, Discussions and conclusions are given in Section 6.

2. Methodology

This Section will provide an overview about the solution methodology, then it will explain the system identification idea and ANN benefits comparing to other approaches. and identify the optimization technique proposed for tuning PID parameters. Finally, the validation methods and evaluation metrics used as performance indicators are discussed.

2.1. Control Tuning

Since in our case, the mathematical model of the system is not available, in this project, we are focusing on model-free data-driven tuning method [37]. We will use machine learning (ML) with operating Dataset to predict the PID parameters. Using ML in control systems has become a trend in the research control engineering field. A lot of research has been done trying to improve PID tuning by using ML algorithms such as ANN, Genetic algorithm, linear regression, and reinforcement learning algorithms.

The proposed prediction model is for adaptive PID tuning. It consists of four main components as shown in Figure 1. Building this model will be done through two main steps.

The first step consists of modeling the system that is to be controlled by PID controller. Identifying the model based on the input (u) and measured output (y) data. Neural networks will be used for model identification. The model will be trained by a gradient decent algorithm and validated to be ready for tuning optimization.

The second step consists of applying the Particle Swarm Optimization (PSO) algorithm for adjusting the controller tuning parameters based on feedback error from NN model until reaching minimum error. PID parameter values corresponding to the minimum error will be considered as optimum PID parameters [42].

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

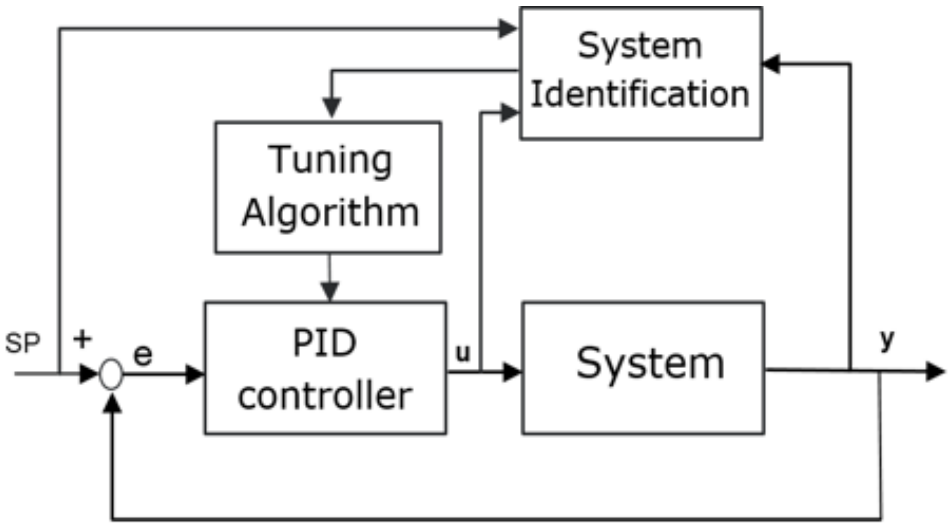


Figure 1.
Identification and tuning block diagram.

2.2. System Identification

Control system optimization for the nonlinear process is not possible in the real production lines for many reasons such as production loss and safety of the human and machines [10,11]. In practice, control design and optimization are only done in the simulated environment. System identification is an experimental modeling technique to simulate the actual system. It is based on real input and output measured operational data that were extracted from the actual system. Fitness is to estimate relationships between input and output data [49]. There are different methods to identify system dynamics, Identification can be done by many methods such as using frequency measurement, correlation analysis, or neural networks.

In practice, identification is done by statistical simulation tools such as Maltlab and Labview. These tools determine the system model by analyzing the signal frequency, amplitude, and time delay. It can be used to validate the NN model or as an alternative identification approach.

2.2.1. Artificial Neural Networks for Identification

Neural networks are used widely in industrial applications for system identification, pattern recognition, and fault detection. In system identification, it plays a significant role in systems modeling. It is considered a good alternative to solving mathematical models due to complexity issues [42]. There are many reasons for using ANNs instead of another algorithm such as:

- Their capability to disclose the nonlinear behavior of complex systems without any prior knowledge of their physics and math compared to other methods, which rely on theoretically derived models.
- They offer good generalization results and work with incomplete data [43].
- They are adaptable to the problem at hand, and many online and offline learning algorithms are available for learning.
- ANN implementation is easy and shows good predictive performance.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

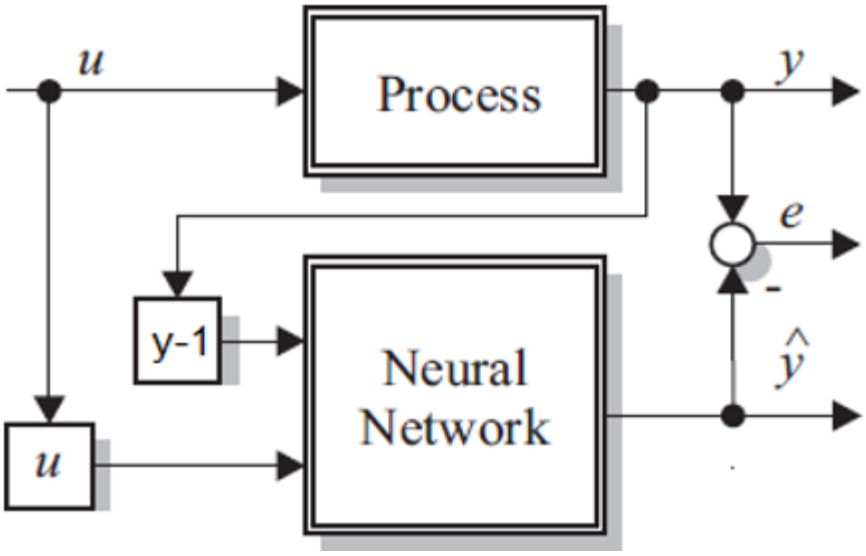


Figure 2.
ANN-based system identification block diagram [49].

Neural network identification processes as illustrated in Figure 2, use the actual system parameters to identify the system model, where u is the desired PID response, y future process output, and $y-1$ is the previous output. $\hat{y}$ is the prediction output and e is the error value between the actual system output and identified model approximation as depicted in the following equation:

$$\hat{y}(k) = f\big(u(k), y(k-1)\big) \quad (1)$$

where f is the NN's transfer function.

2.3. PID Parameters Tuning

The proposed optimization method is a hybrid method combining particle swarm optimization (PSO) algorithm and Genetic algorithm. Optimization processes are done in sequence, first step, PSO starts to examine the suggested PID values until the fitness value reaches the targeted function or reaches to maximum time without any improvement. Second step, GA continues the optimization after the PSO terminates. The GA optimization process starts with the PID values that are optimized by PSO as an initial point and tries to find the minimized fitness value.

2.3.1. Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is a type of evolutionary algorithm, introduced by Kennedy and Eberhart [45]. PSO was proposed for tuning PID controller and finding its optimal parameters (Kp, Ki, Kd). It has many advantages such as; requiring few numbers to be assigned, and sorting of fitness values output is not required, which is a significant advantage compared to other algorithms. especially for large population sizes. Also, it does not require an encoding and decoding process for updating velocity and position, just a simple arithmetic operation for real numbers [39–41].

It is a population-based optimization method. It contains particles, each particle has a position and velocity. As shown in Figure 3, PSO starts by generating an initial swarm of particles with a random position and velocity. Each particle is then evaluated for fitness value [39]. Each time a fitness value is calculated, it is compared against the previous best fitness value of the particle and the previous best fitness value of the whole swarm, and best and global best positions are updated until the stopping criterion is met [39–41].

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

2.3.2. Genetic Algorithm

GA is proposed as a hybrid function with PSO. It starts an optimization process with PID return by PSO as the initial population of individual Chromosomes. GA continues the optimization and tries to minimize the fitness function. The steps involved in implementing GA for PID tuning are shown in Figure 4. Start with the PID predicted by PSO as an initial population of individual Chromosomes (K_p, K_i, K_d) with a fixed size, then evaluate the value of the fitness function. Based on fitness values, start selection, crossover, and mutation operations and make up the new generation. repeat these processes until getting the best value, chromosome with the highest fitness will be considered as optimal PID controller parameters.

2.4. Model Validation

Model validation is done for the system identification step by comparing the predicted data with actual data from the dataset through the mean square error (MSE). As a first step, the model development and validation will be lab-based (Matlab) and the model output will be compared with the actual system output.

The Predicted PID parameters will be tested on the real PID controllers. PID parameters will be changed and the controller response will be supervised in the real environment. Error metrics such as mean square error will be calculated to evaluate the quality of the result.

2.5. Evaluation Metrics

The integral error indices are used to measure and evaluate the system error. Integral Error indices are defined as a quantitative measure to evaluate the system's performance. They can be used both in System Identification and in the tuning of PID parameters.

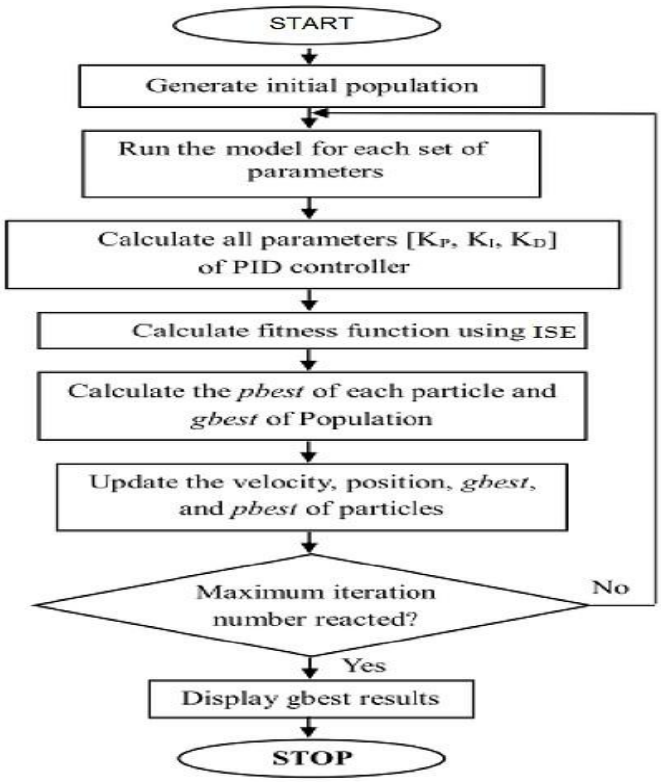


Figure 3.
PSO algorithm flow chart.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

They calculate the difference between the set point and the corresponding measured process variable (e(t) = SV − PV) over a period of time. Such metrics include [44]:

- Integral Absolute Error (IAE): $IAE = \int_0^t |e(t)|\, dt$
- Integral Time-Weighted Absolute Error (ITAE): ITAE = $ITAE = \int_0^t t|e(t)|\, dt$
- Integral Squared Error (ISE): $ISE = \int_0 t_{e^2(t)}\, dt$
- Mean Squared Error (MSE): $MSE = \int_0^t \frac{e^2(t)}{t} dt$

3. Data Collection and Feature Selection

In this project, we have considered using real data to build the model. We have got permanent access to real-time and historical data systems. Data access is through Plant Information Management System (Exaquntum), which saves all historical data for the plant.

Plant Information Management System (PIMS) is used for extracting data for each PID controller, each PID logic is associated with six parameters to calculate the output signal. Three fixed parameters for each PID controller; proportional, integral, and derivative values for PID, and another three variable parameters for each process; set point (SV), process variable (PV), and final control opening (MV). In Figure 5.a, an example of a level controller receives the feedback from a level transmitter as PV to control the level with 50% set point (SV). PID parameter values are extracted from the PID tuning panel as depicted in Figure 5.b.

For each PID controller, a separate Excel data file is collected. It contains the set points (SV), process variables (PV), manipulated variables (MV), time, and date, as shown by the excerpt in Figure 6. The other three parameters P, I, and D are fixed for each controller and will be manually added in the optimization stage. The Excel file will be generated for each PID controller separately. The datasets are collected for a long period, more than 6 months.

The input and output of PID controller and tuning parameters are all selected as features. Datasets are extracted with all these features as shown in Figure 5. The list of these features and their sources in control system are shown in Table 1.

Table 1.
Selected features.

Features list	
Feature	**Source**
- (SV) Set Point Value.	- PID controller SV tag number.
- (PV) Process Variable Value.	- Feedback transmitter signal tag number.
- (MV) Manipulate VariableValue.	- Final control element signal tag number.
- (P) Proportional Parameter Value.	- Function block setting.
- (I) Integral Parameter.	- Function block setting.
- (D) Derivative Parameter.	- Function block setting.
- Time Stamp.	- Network time server.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

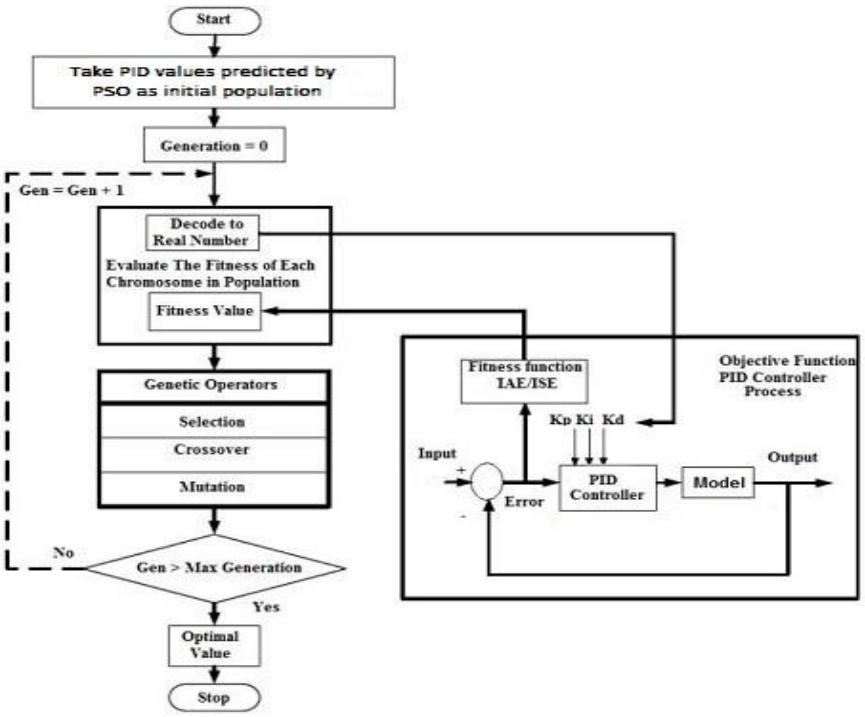
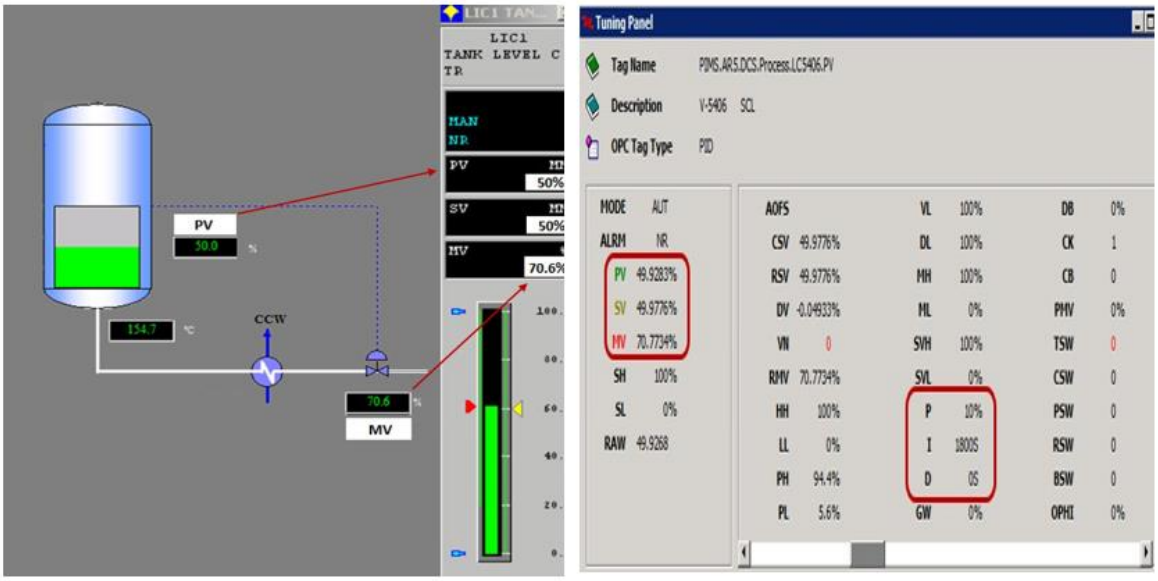


Figure 4.
GA-PID tuning flow chart.



(a) PID Controller Face Plate (b) PID Tuning Panel

Figure 5.
PID tuning system: (a) PID controller face plate, (b) PID tuning panel.

4. Solution Details

Neural Networks are proposed to be used for the identification of the actual system model. Training methods and operational data can be adjusted until reaching a high accuracy model.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

	A	B	C	D	E
1	FC5124.SV.Value	FC5124.PV.Value	FC5124.MV.Value	TimeStamp	Date
2	8322.3740	8267.2969	74.8787	12:00 AM	09/01/18
3	8344.0654	8293.3359	75.2178	12:01 AM	09/01/18
4	8324.5889	8369.5820	75.0268	12:02 AM	09/01/18
5	8351.5889	8426.5908	75.2749	12:03 AM	09/01/18
6	8335.7852	8406.3818	74.9658	12:04 AM	09/01/18
7	8350.7061	8358.8525	75.1180	12:05 AM	09/01/18
8	8335.3330	8272.8418	75.2188	12:06 AM	09/01/18

Figure 6.
Extracted dataset.

4.1. Radial Basis Function Neural Networks (RBFNN)

In our case, Radial Basis Function neural network structure is proposed. It is a feed forward and fully connected neural network. RBFNN has a good approximation properties and shows high accuracy of approximation. Also, the connection weights from the hidden layer to the output layer are linear and we can use the linear algorithm to ensure the global convergence of the parameters [46–48]. Moreover, during the training time one part of the nodes is affected by a given input, and only the parameters need to be adjusted, which reduces computation and the training time [47].

RBFNN has three layers; an input layer, a nonlinear hidden layer, and a linear output layer, as shown by Figure 7. Nodes in each layer are fully connected to the previous layer. Inputs to input layer are connected directly to the hidden layer without weights. Euclidean distances between the centers and the network input are calculated, then send to hidden layer and pass through RBF units. Output layer nodes are weighted linear combinations of the hidden layer.

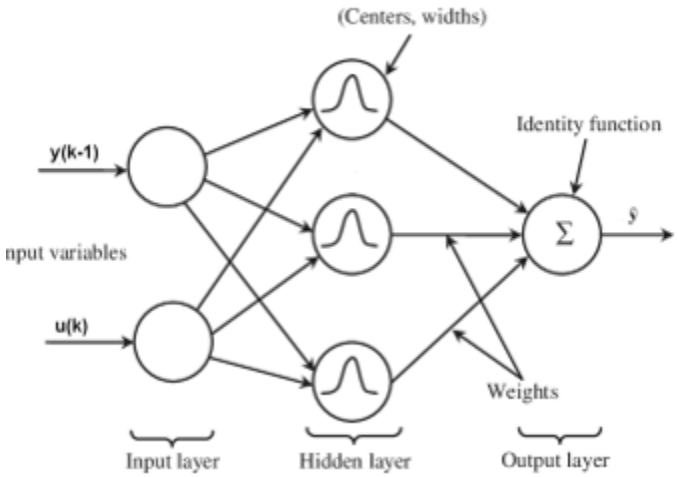


Figure 7.
RBF neural network architecture.

Input layer has 2 neurons and their inputs are $X = [u(k), y(k-1)]$, where u(k) and y(k) are the desired response and the actual output respectively (see Figure 2). The hidden layer contains 3 neurons

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

H = [h_1, h_2, h_3], is the radial basis vector, where hj is a Gaussian function, which can be described through the following formula:

$$h_j = \phi_j\big(X(k)\big) = \phi_j\big(X, C_j, \sigma_j\big) = exp\left(-\frac{\|x(k)-C_j\|^2}{\sigma_j^2}\right) \quad (2)$$

Where C_j and σ_j are Gaussian function parameters. The output layer contains one neuron and its output variable $\hat{y}$ is calculated through following equation:

$$\hat{y} = \sum_{j=1}^{m} w_j \Phi_j \quad (3)$$

where W_j is the weight of the j^{th} node and Φ_j is the j^{th} activation function.

RBFNN are defined by three important parameters: centers, widths and weights. These parameters are unknown and a learning strategy should be used to predict and optimize these parameters. Gradient descent (GD) is proposed to select an appropriate set of RBFNN parameters. The system is considered as identified when $\hat{y}$ matches y. Mean Square Error is used to measure the model approximation accuracy.

4.1.1. RBFNN System Identification Steps

RBFNN-based system Identification steps are as follows:

- Apply the input from the training data set to the input layer.
- Compute the output of the hidden layer.
- Compute the output and compare the output with actual output y and adjust the weight to minimize the error.
- Repeat steps 1 to 3 for all training set.
- Repeat steps 1 to 4 until the error tends to zero.

When the model is trained, validated and tested with high prediction accuracy, then it is ready for tuning step. The tuning algorithm will change the PID parameters based on the model output. Adjusting the parameters, consequently, will change the PID controller output signal, which will be reflected in the model. Model output error again will be sent back to tuning algorithm. Repeat this loop until you reach to minimum error or complete the maximum iteration number.

4.2. Tunning PID Parameters Optimization

In this section will explain our proposed optimization methods, include optimization steps, equations and hyper parameter for PSO and GA.

4.2.1. Particle swarm optimization (PSO)

Tuning PID parameters by PSO algorithm is the second step after system identification. It starts with an initial swarm of random particles, each particles candidate as solution (PID values) with defined range. Particles assigned with a randomized velocity Vi and position Xi.

The 3D search space is defined, and all its points represent a particular combination of K_p, K_i, K_d parameters values. Randomly each particle is scattered in the swarm and represented by 3xSwarm matrices. Depending on the information that get exchanged between particles, each particle adjusts and adapts its direction and remember its best fitness function (p_{best}) and the best global position obtained by any particle (g_{best}).

The feedback error from system model is considered as fitness function input. Fitness function calculates the integral of model error to select the best set of PID parameters [39-41].

The particles are updated according to following equation:

$$V_i^d = WV_i^d\, max + c_1 r_1\big(X_i^d P_{best} - X_i^d\big) + c_2 r_2\big(X_i^d P_{good} - X_i^d\big) \quad (5)$$

Where, c1 and c2 are acceleration constant. r1 and r2 are random numbers between 0 and 1. W is the inertia weight. i is the index of the d-dimensional particle, and:

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

- $X_i^d = [x_i^1,\ x_i^2, \ldots\ x_i^d]$represents the present position for particles.
- $X_i^d p_{best} = \left[x_i^1 p_{best},\ x_i^2 p_{best}, \ldots\ x_i^d p_{best}\right]$, reprents the best positions visited by particles.
- $X_i^d p_{best} = [x_i^1 p_{best},\ x_i^2 p_{best}, \ldots\ x_i^d p_{best}]$, represents the best global position.
- $V_i^d = [v_i^1,\ v_i^2, \ldots\ v_i^d]$, represents the velocity of particles.
- $V_i^d max = [v_i^1 max,\ v_i^2 max, \ldots\ v_i^d max]$represents the upper bound velocity of the particle.

4.2.2. PSO-PID Tuning Steps

PSO-PID Tuning steps are as follows:

- **Step 1:** Start after setting the constants, Inertia weight factor W, Acceleration constants c_1 and
- c_2.
- **Step 2:** set the initial number of swarms in the 3D search space with $\lceil X_i^1, X_i^2, X_i^3 \rceil$ and $\lceil V_i^1, V_i^2, V_i^3 \rceil$ as initial position and velocity.
- **Step 3:** Evaluate the fitness function.
- **Step 4:** If the fitness value is greater than p_{best}, then go to step 5, else, go to step 8.
- **Step 5:** set p_{best} with present fitness value.
- **Step 6:** If the present fitness value is greater than g_{best}, then go to step 7, else, go to step 8.
- **Step 7:** set g_{best} with present value of fitness function.
- **Step 8:** Update the position and velocity values of the particles.
- **Step 9:** Exit if it meets the defined end criteria, else, go to step 3.

4.2.3. PSO Hyper-Parameters

The hyper parameters can be adjusted to start adapting the parameters. Start with low parameters, then adjuste depending on the results. The PSO-PID tuning listed in Table 2.

4.2.4. Genetic Algorithm

The GA used as hybrid function with PSO algorithm. It starts with the PSO optimization result as initial population. The genetic algorithm generates PID parameters and assigns the new values to controller, then the controller will react based on these new values and send new output to the final control. Integral square error function will calculate the error over time and send it to GA to evaluate the previous PID parameters values, repeat this loop until reach to the minimum fitness function value or max iteration. The PID parameters values corresponding with minimum error will be selected as optimal parameters [23–25].

Table 2.
PSO-PID tuning hyper parameters.

Variable	Description
n	Size of the swarm (# of birds)
Bird step	Max. no. of bird steps
dim	Dimension of the problem
c_1	Velocity constant
c_2	Velocity constant
w	Momentum of inertia
minKp	Minimum value of agent for Kp
maxKp	Maximum value of agent for Kp
minKi	Minimum value of agent for Ki
maxKi	Maximum value of agent for Ki

4.2.5. GA-PID Tuning Steps

The GA-PID Tuning algorithm consists of many steps and elements as follows:

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

- **Coding and Decoding:** The initial population is set by encoding the PID parameters to the binary as chromosomes with fixed length of zero and ones. The length of the strings depends on the PID values range. The required bits is calculated based on the following equation:

$$2^{(m_j-1)} < (b_j - a_j) * 10^4 < 2^{m_j} - 1 \quad (6)$$

Where m_j is the number of the bits, and b_j upper bound, a_j is the lower bound of PID parameters values.

- **Fitness Function:** Evaluate the fitness values of all chromosomes and find a group of the best chromosomes by converting their binary number to real number which represents the PID parameters. The encoding process is done for each chromosome from binary to real as the following equation:

$$X_j = a_j + PID - value * \frac{(b_j - a_j)}{2^{m_j}-1} \quad (7)$$

PID parameters set to pass to the controller, system feedback, and initial fitness value are computed. The performance criterion functions are used as fitness functions [26]. For each set of PID parameters new fitness value will be generated and measured by ISE performance metric.

- **New population:** In our case start by taking the PSO output as the initial population to get the new population by doing the selection, crossover, and mutation processes.
- **Selection:** select the two parents from the population depending on their fitness.
- **Crossover:** Crossover pairs of chromosomes in the new generation based on their probability and crossover rate.
- **Mutation:** Mutation is a swap between the chromosomes to maintain diversity in the genetic population by removing the low probability.
- **GA Hyper-Parameters:** GA Hyper Parameters can be adjusted to start adapting the parameters. Start with low parameters, then adjust depending on the results. The GA-PID tuning is listed in Table 3.

Table 3.
GA hyper parameters.

Parameter	Type/value
Generations	Number
Population size	Number
Encoding	Binary
Selection	Uniform
Mutation	Uniform

5. Experimental Results

The implementation of the system will start with system identification coding in section 5.1, followed by coding of PID tuning with PSO in section 5.2. In section 5.3, the integration of Matlab and Simulink is done to show real interaction between project components.

5.1. System Identification

RBF neural network with gradient descent is selected to perform system identification. The Monte-Carlo Simulation (MCS) Method is used to train the RBF network. The system identification Algorithm is shown in the Pseudo Code of Algorithm 1 RBFNN.

5.2. PID Tuning Parameters Optimization

The second part of the implementation is coding the optimization method for finding the optimal PID parameters. Particle Swarm Optimization (PSO) algorithm combined with hyperfunction is selected to solve bounded constrained problems with a fitness function. The following pseudo-code shows the main steps for the optimization process.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

Algorithm 1: RBFNN For System Identification

```
Data Importing;
Extract Features;
- SV, PV and MV;
Monte Carlo Simulation Initialization;
- Number Of Runs;
- Number Of Epochs;
RBFNN Training Parameters Initialization;
- Number Of Neurons;
- Spread of Gaussian Kernels;
for i ← 0 to Number Of Runs do
    Random initialization of the RBFNN Weights and Bias;
    for i ← 0 to Number Of Epochs do
        for i ← 0 to Number Of Samples do
            for i ← 0 to Number Of Neurons do
                Calculating the kernel matrix
            end
            Calculate RBFNN output;
            Calculate error of estimation;
            Update wights/Bias, Use Gradient Descent;
        end
        Calculate MSE, fitness Function;
        Accumulating MSE, Weights and Bias of each epoch;
    end
    Final training MSE, Weights and Bias after runs all Epochs;
end
```

5.3. Simulink Model Integration

Simulink is a block diagram environment for multidomain simulation. It supports system-level design, simulation, and continuous testing and verification of control systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB and helps to incorporate Matlab algorithms into models and export simulation results to Matlab for further analysis [56].

As shown in Figure 8, there are two subsystems; PID controller and identified process model (RBFNN). PID controller subsystem takes error e as input and calculates PID controller output u with the help of optimized PID gains which are received from the PSO Matlab function through the sinTrack function.

5.4. Model Outputs

The final output after completing the optimization and simulation process will show in the command lines, the iteration number, evaluation counter, best fitness function, average of evaluated iteration, and stall iterations counter as shown in Table 4.

- f-count: Number of fitness function evaluations.
- Best f(x): Best evaluated fitness function value.
- Mean f(x): Mean of fitness function values.
- fval: Final fitness value.

5.5. Model Validation on Real Process

The validation step for each study case is done in the real PID controller itself. Official approval is obtained for conducting our validation. The PID parameters were predicted by our model and are

applied instead of the actual PID parameters. Predicted PID parameters show notable enhancement in controlling and more stability in process.

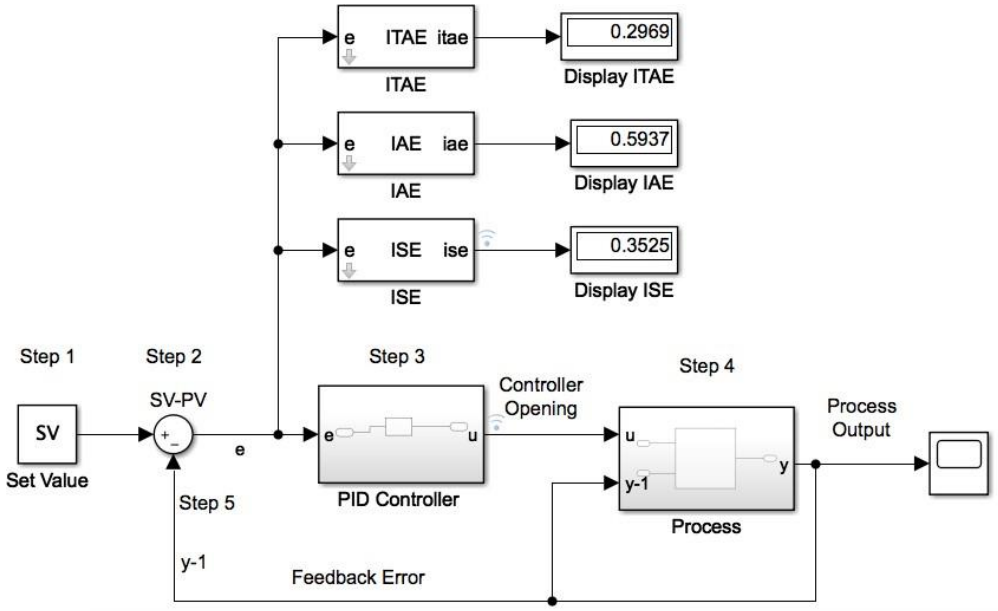


Figure 8.
Integrated simulation block diagram.

Table 4.
Optimization outputs.

Iteration	f-count	Best f(x)	Mean f(x)	Iterations Stall
22	1150	0.7447	0.7794	4
KP	KI	KD	fval	eflag
16.8705	1703.63.99	0	0.7447	-4

5.5.1. Predicted PID Validation: Case 1

In this case, the process variable (PV) has high oscillating around the set value (SV), because of the process under high pressure and continuous high flow of liquid. The PID controller tries to track the PV changing but still, there is a gap and overshooting. As shown in Figure 9 the controller opening (pink color trend) does not react correctly with high change in process variable (blue color trend). The actual PID controller parameters are P = 200, I = 10 and D = 0, repealed with the model prediction PID parameters P = 200.3, I = 22, and D = 0 as shown in Figure 10. The controller response becomes better and minimizes the overshooting range. Process variable change frequency becomes slower than before and close to the set value. The predicted PID increases the Integral parameter (I) to avoid the reaction with instant process overshooting. PID controller will react based on more accumulation errors to ensure its action is not caused by instant overshoot due to process disturbances.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

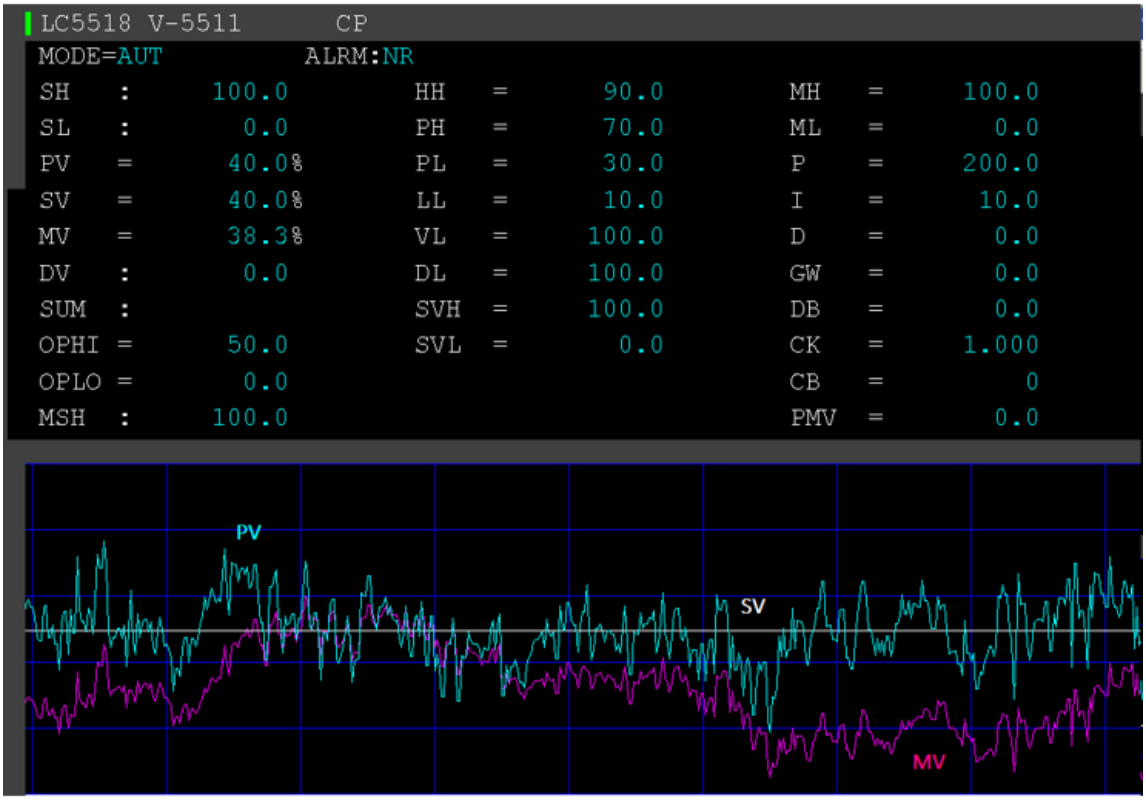


Figure 9.
Actual PID response.

Integral Squared Error (ISE) and Integral Absolute Error (IAE) are calculated to compare the rate of error (SV-PV) for the actual PID parameters and model prediction as illustrated in Table 5.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

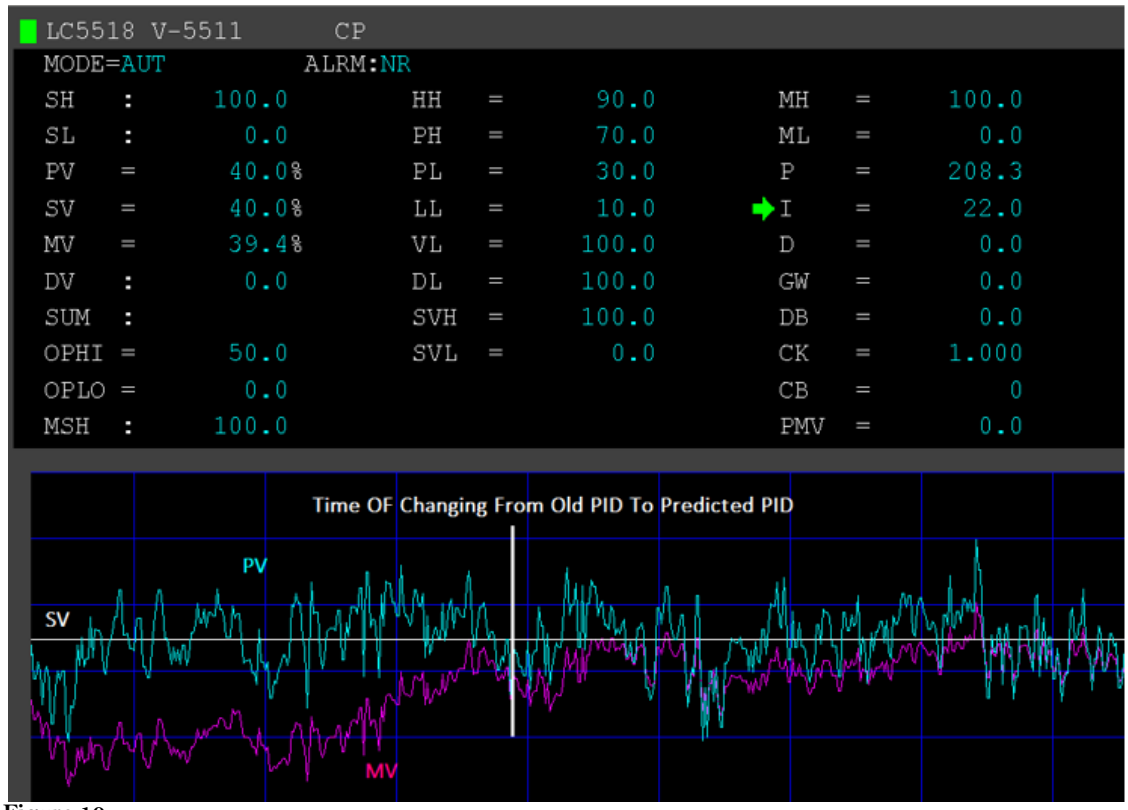


Figure 10.
Predicted PID response.

Standard Deviations for PV and MV are collected from the Plant Information Management System (PIMS) after validation. SD compares the two PIDs responses by measuring the stability of the process variable and controller opening. Predicted PID shows less SD than actual PID parameters as shown in Tables 6 and 7.

Table 5.
Calculated ISE and IAE (case 1).

PID	ISE	IAE
Actual PID	2.332	5.441
Predicted PID	1.886	1.373

Table 6.
Actual and predicted PID PV values (case 1).

	Min value	Average value	Max value	Standard dev.
Predicted PV values	38.041	39.949	42.396	0.941
Actual PV Values	38.041	39.945	42.396	1.009

Table 7.
Actual and predicted PID MV values (case 1).

	Min value	Average value	Max value	Standard dev.
Predicted MV values	36.345	38.223	39.447	0.861
Actual MV Values	34.781	37.886	41.605	1.595

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

5.5.2. Predicted PID Validation: Case 2

In this case, the problem was observed is process variable (PV) has short continuous oscillating around the set value (SV), as depicted in Figure 11. The actual PID parameters are P = 99.3, I = 50 and D = 0, replaced with Predicted PID parameters are P = 91.2, I = 50 and D = 0. Figure 12 shows that predicted PID parameters decrease the proportional parameter (P), which makes the controller's action less aggressive to minimize the oscillating of PV.

Integral Squared Error (ISE) and Integral Absolute Error (IAE) are calculated to measure the rate of error (SV-PV) for the actual PID parameters and model prediction as illustrated in Table 8.

Table 8.
Calculated ISE and IAE (case 2).

PID	ISE	IAE
Actual PID	0.08147	0.2854
Predicted PID	0.01151	0.1073

SD for PV and MV are collected from the Plant Information Management System (PIMS) after validation to compare both PIDs parameters response. predicted PID shows SD less than actual PID parameters as shown in Tables 9 and 10.

Table 9.
Actual and predicted PID PV values (case 2).

	Min value	Average value	Max value	Standard dev.
Predicted PV values	39.858	40.022	40.140	0.132
Actual PV values	39.566	39.954	40.716	0.304

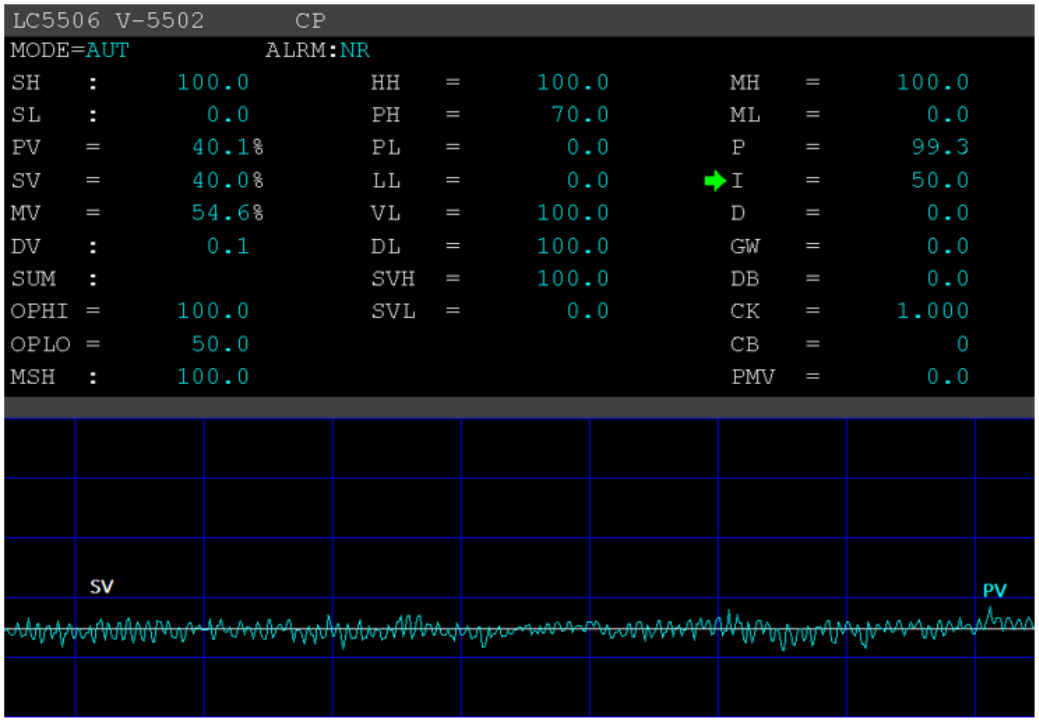


Figure 11.
Actual PID response.

Table 10.
Actual and predicted PID MV values (case 2).

	Min value	Average value	Max value	Standard dev.
Predicted MV values	52.928	53.375	53.750	0.210
Actual MV Values	53.985	54.436	54.962	0.305

5.5.3. Predicted PID Validation: Case 3

In this case, the controller has a low response to the process change. If the deviation between the PV and SV increases, it takes a long time to open the controller and eliminate the deviation. This means the controller response is very low compared to the process dynamics. The actual proportional value (P) is 10, the integral is 1800 and the Derivative(D) is 0 as shown in Figure 13. The set value (SV) is 50%, we have changed it to check the controller response before validating our prediction PID parameters. As shown in Figure 13, SV changes from 50% to 50.5%, PID controller shows a slow response.

Actual PID parameter values have changed to our predicted PID parameters as shown in Figure 14, the new values I = 16.8705, P = 1703.6399, and D = 0. The SV again changed from 50.5% to 50% to check the controller response. The PID controller response has become faster than before, it starts opening directly after changing the set value. Increasing the proportional parameter (P) makes the controller reaction more aggressive, and decreasing the Integral (I) minimizes the time the controller should wait until response to the error.

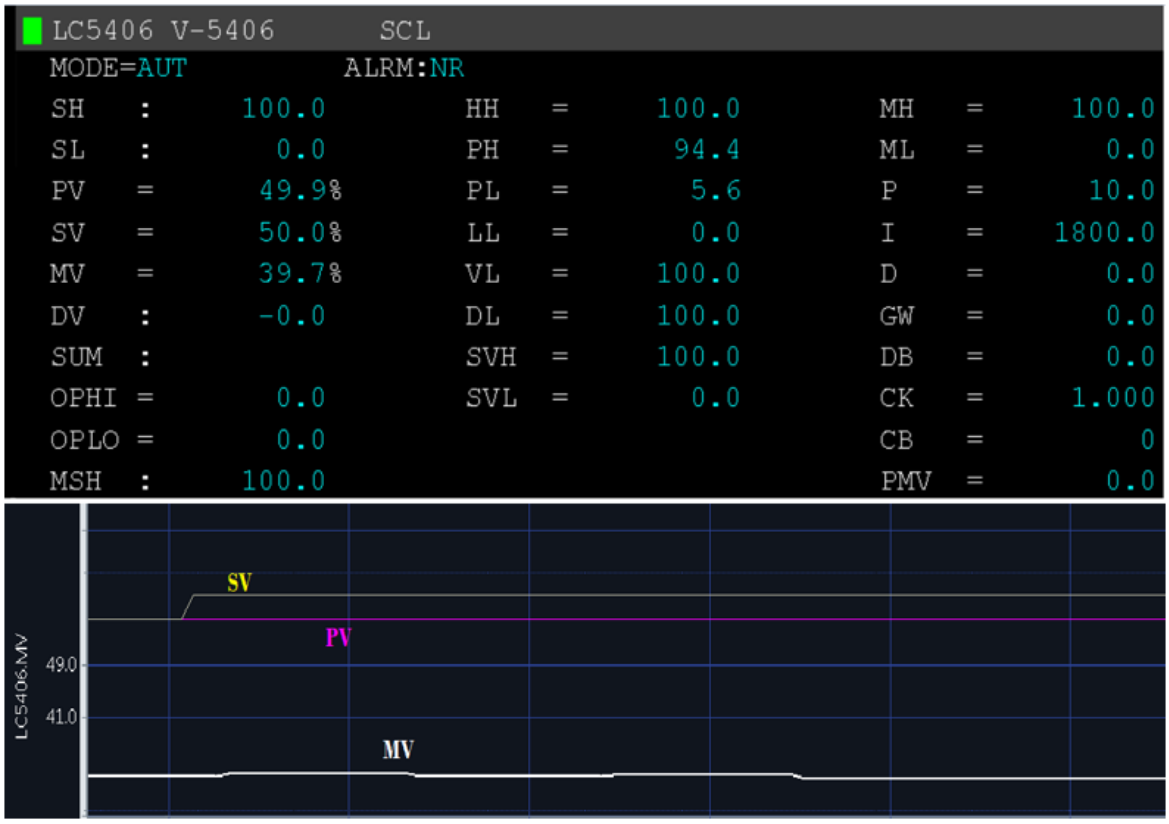


Figure 13.
Actual PID response.

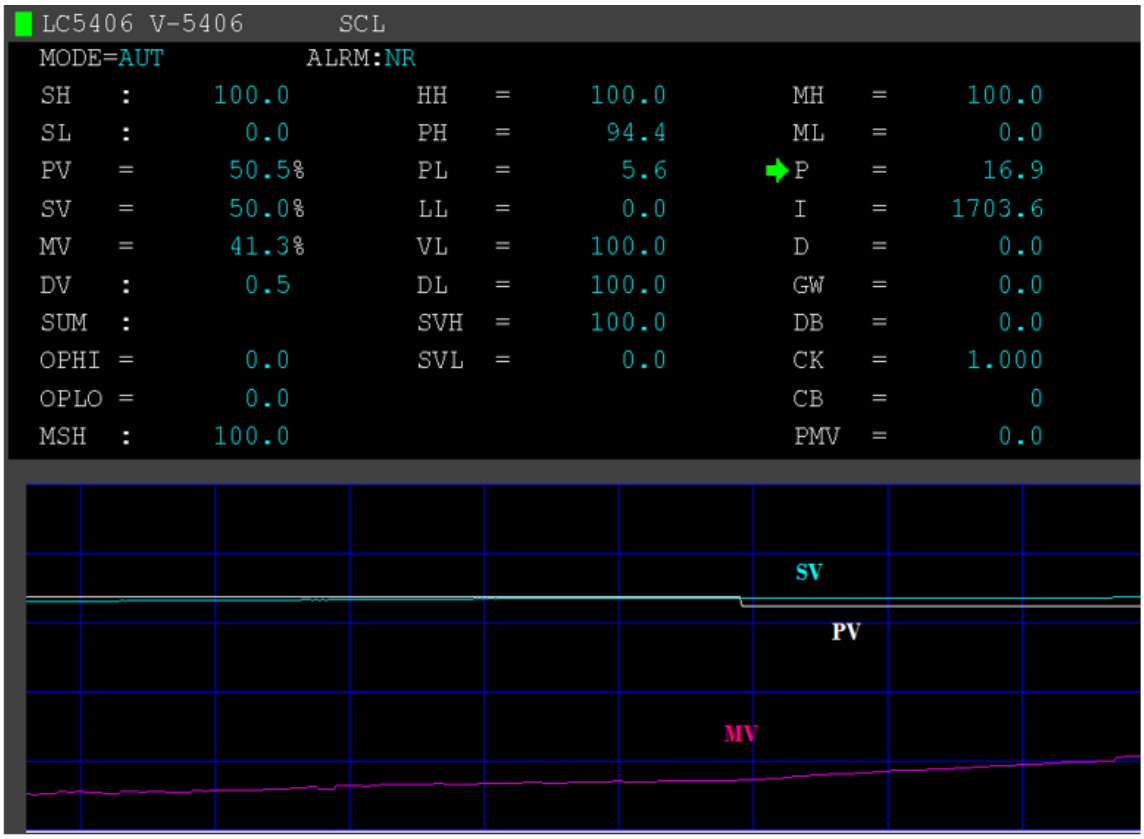


Figure 14.
Predicted PID response.

Integral Squared Error (ISE) and Integral Absolute Error (IAE) are calculated to measure the rate of error (SV-PV) for the actual PID parameters and model prediction as illustrated in Table 11.

Table 11.
Calculated ISE and IAE (case 3).

PID	ISE	IAE
Actual PID	0.06969	0.246
Predicted PID	0.01796	0.134

Standard Deviation (SD) for PV and MV are collected from Plant Information Management System (PIMS) after validation to compare between both PIDs parameters. predicted PID shows SD less than actual PID parameters as shown in Tables 12 and 13.

Table 12.
Actual and predicted PID PV values (case 3)

	Min value	Average value	Max value	Standard dev.
Actual PV values	50.236	50.346	50.448	4.295E-02
Predicted PV Values	50.132	50.199	50.236	5.005E-02

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576

Table 13.
Actual and predicted PID MV values (case 3).

	Min value	Average value	Max value	Standard dev.
Actual MV values	39.395	39.652	39.810	0.160
Predicted MV Values	38.932	39.157	39.395	0.135

6. Discussion and Conclusions

This research considered using machine learning for PID tuning optimization through system identification, instead of conventional methods that are based on signal processing, heuristic tuning, or mathematical models. For simulation, datasets obtained for various PID controllers from real plants were used. Radial Bias Networks are trained for each tuning case to simulate the dynamic of the process model to perform the PID parameters optimization on them. Validation and testing show that RBFNN can simulate the actual process with minimal errors.

Tuning PID parameters using ML algorithms as optimization methods, has been shown through simulation and experiments to provide substantial improvement in the controller response and process stability. The performance indexes based on error criteria prove that the error rate for the controller tuned using our model is less than the controller-tuned conventional methods. PID parameters obtained from model predictions applied in the live and real production process.

The optimization outcomes minimized the dead time, processed overshooting, and increased the controller response. Live trends and standard deviation calculations were collected from a real control system after Implementation. The implementation is verified and approved by control systems and plant operation department's managers and engineers, and its outcomes are considered as permanent improvement change.

Acknowledgement:

This work was supported by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia.

References

[1] Ribeiro, J. M., Santos, M. F., Carmo, M. J., Silva, M. F. (2017). Comparison of PID controller tuning meth- ods: Analytical/classical techniques versus optimization algorithms. 2017 18th International Carpathian Control Conference (ICCC). doi:10.1109/carpathiancc.2017.7970458.

[2] Automatic tuning and adaptation for PID controllers - A survey. (1993). Control Engineering Practice, 1(6), 1081. doi:10.1016/0967-0661(93)90079-7.

[3] Leva, A. (2018). PID-based controls in computing systems: A brief survey and some research directions. IFAC-PapersOnLine, 51(4), 805-810. doi:10.1016/j.ifacol.2018.06.178

[4] Development of the PID controller. (1993). IEEE Control Systems, 13(6), 58-62. doi:10.1109/37.248006

[5] Portillo, J., Marcos, M., Orive, D. (1999). Auto-tuner: Tuning commercial PID controllers on demand in industrial environments. 1999 European Control Conference (ECC). doi:10.23919/ecc.1999.7099395

[6] oshi, M., Uniyal, J., Juneja, P. K. (2016). Robustness analysis of controller designed for first, second and third order delayed process model. 2016 International Conference on Emerging Trends in Communication Technologies (ETCT). doi:10.1109/etct.2016.7882925

[7] Xu, Y., Wang, D., Lin, S., Zhang, W. (2011). A simple PID controller tuning strategy for first order plus dead time model. 2011 International Conference on Electronics, Communications and Control (ICECC). doi:10.1109/icecc.2011.6066750

[8] Nath, U. M., Dey, C., Mudi, R. K. (2017). Design of modified model-based adaptive control system for FOPDT processes. 2017 4th International Conference on Power, Control mEmbedded Systems (ICPCES). doi:10.1109/icpces.2017.8117667

[9] Alina-Simona, B., Nicolae, P., Daniel, M. (2011). Using an internal model control method for a distillation column. 2011 IEEE International Conference on Mechatronics and Automation. doi:10.1109/icma.2011.5986231

[10] Tang, W., Liu, Z., Wang, Q. (2017). DC motor speed control based on system identification and PID auto tuning. 2017 36th Chinese Control Conference (CCC). doi:10.23919/chicc.2017.8028376

[11] Serrano, V., Tsakalis, K. (2016). A study on the on-line system identification and PID tuning of a buck converter. 2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC). doi:10.1109/icnsc.2016.7479008
[12] Schei, T. S. (1992). Closed-loop Tuning of PID Controllers. 1992 American Control Conference. doi:10.23919/acc.1992.4792692
[13] Ziegler J. G. and Nichols N. B., (1942).Optimum settings for automatic controllers, Transactions of the American society of Mechanical Engineers, , pp. 759-768.
[14] Zhang, X., Yang, L., Li, Y. (2012). Single neuron PID controller based on adaptive Smith predic- tor for networked control systems. IEEE 10th International Conference on Industrial Informatics. doi:10.1109/indin.2012.6301217
[15] Hu, J., Zhang, H., Wu, G. (2015). Simulation of networked control system based on single neuron adaptive PID. 2015 IEEE International Conference on Information and Automation. doi:10.1109/icinfa.2015.7279442
[16] Liu, G. (2000). Neural-learning control of nonlinear dynamical systems. IEE Seminar Learning Systems for Control. doi:10.1049/ic:20000346
[17] Zheng, Z., Wang, N. (2002). Model-free control based on neural networks. Proceedings. International Conference on Machine Learning and Cybernetics. doi:10.1109/icmlc.2002.1175425.
[18] Wabgaonkar, H., Stubberud, A. (1992). Introduction to Non-linear Control Using Artificial Neural Networks. Control and Dynamic Systems High Performance Systems Techniques and Applications, 427-522. doi:10.1016/b978-0-12-012753-5.50016-8
[19] Mahmud, K. (2013). Neural network based PID control analysis. 2013 IEEE Global High Tech Congress on Electronics. doi:10.1109/ghtce.2013.6767259.
[20] Lin, H., Changlin, M., Feng, L. (2007). Study of Adaptive PID Controller Based on Single Neuron and Genetic Optimization. 2007 8th International Conference on Electronic Measurement and Instruments. doi:10.1109/icemi.2007.4350432
[21] Zhang, M., Li, W. (2006). Single Neuron PID Model Reference Adaptive Control Based on RBF Neural Network. 2006 International Conference on Machine Learning and Cybernetics. doi:10.1109/icmlc.2006.258358
[22] Patel, R., Kumar, V. (2015). Multilayer Neuro PID Controller based on Back Propagation Algorithm. Procedia Computer Science, 54, 207-214. doi:10.1016/j.procs.2015.06.023.
[23] Fan, L., Joo, E. M. (2009). Design for auto-tuning PID controller based on genetic algorithms. 2009 4th IEEE Conference on Industrial Electronics and Applications. doi:10.1109/iciea.2009.5138538
[24] Xiao, L., Han, C., Xu, X., Huang, W. (2010). Hybrid genetic algorithm and application to PID controllers. 2010 Chinese Control and Decision Conference. doi:10.1109/ccdc.2010.5498979
[25] Psichogios, D. C., Ungar, L. H. (1991). Direct and indirect model based control using artificial neural networks. Industrial Engineering Chemistry Research, 30(12), 2564-2573. doi:10.1021/ie00060a009.
[26] Sasaki, M., Matsuzaki, H., Koizumi, K. (1995). Optimal PID controller with saturation for impact machines by genetic algorithms. SICE 95. Proceedings of the 34th SICE Annual Conference. International Session Papers. doi:10.1109/sice.1995.526667
[27] Cheng, J., Ai, L., Xiong, W. (2012). Parameter optimization of PID controller for boiler combustion system based on adaptive genetic algorithm. International Conference on Automatic Control and Artificial Intelligence (ACAI 2012). doi:10.1049/cp.2012.1273
[28] Parvathy, R., Devi, R. R. (2014). Gradient descent based linear regression approach for modeling PID parameters. 2014 International Conference on Power Signals Control and Computations (EPSCICON). doi:10.1109/epscicon.2014.6887482
[29] Howell, M. (2000). The application of continuous action reinforcement learning automata to adaptive PID tuning. IEE Seminar Learning Systems for Control. doi:10.1049/ic:20000343
[30] Huang, X., Xiao, S. (2017). Self-Augmenting Strategy for Reinforcement Learning. Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence - CSAI 2017. doi:10.1145/3168390.3168392
[31] Carlucho, I., Paula, M. D., Villar, S. A., Acosta, G. G. (2017). Incremental Q -learning strat- egy for adaptive PID control of mobile robots. Expert Systems with Applications, 80, 183-199. doi:10.1016/j.eswa.2017.03.002.
[32] Carlucho, I., Paula, M. D., Villar, S. A., Acosta, G. G. (2017). Incremental Q -learning strat- egy for adaptive PID control of mobile robots. Expert Systems with Applications, 80, 183-199. doi:10.1016/j.eswa.2017.03.002
[33] M. I. Jordan, and T. M. Mitchell (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245). (http://science.sciencemag.org/content/349/6245/255)
[34] Radac, M., & Precup, R. (2016). Improving model reference control performance using model-free VRFT and Q-learning. 2016 20th International Conference on System Theory, Control and Computing (ICSTCC). doi:10.1109/icstcc.2016.7790632
[35] Silveira, A. S., Coelho, A. A., Gomes, F. J. (2012). Model-Free Adaptive PID Controllers Applied to the Benchmark PID12. IFAC Proceedings Volumes, 45(3), 370-375. doi:10.3182/20120328-3-it-3014.00063
[36] Gao, Z., Huang, Y., Han, J. (n.d.). An alternative paradigm for control system design. Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228). doi:10.1109/.2001.980926
[37] Fliess, M., Join, C. (2013). Model-free control. International Journal of Control, 86(12), 2228-2252. doi:10.1080/00207179.2013.810345
[38] Peischl, B., Wotawa, F. (2003). Model-based diagnosis or reasoning from first principles. IEEE Intelligent Systems, 18(3), 32-37. doi:10.1109/mis.2003.1200725
[39] Ouyang, P., Pano, V. (2015). Comparative Study of DE, PSO and GA for Position Domain PID Controller Tuning.

Algorithms, 8(3), 697-711. doi:10.3390/a8030697.
[40] Anand, A., Suganthi, L. (2018). Hybrid GA-PSO Optimization of Artificial Neural Network for Forecast- ing Electricity Demand. doi:10.20944/preprints201711.0190.v2.
[41] Ouyang, P., Pano, V. (2015). Comparative Study of DE, PSO and GA for Position Domain PID Controller
[42] Tuning. Algorithms, 8(3), 697-711. doi:10.3390/a8030697.
[43] Mhaskar, H., Hahm, N. (n.d.). System identification using neural networks. Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop. doi:10.1109/nnsp.1996.548338.
[44] Tsoi, A. (n.d.). System identification using neural networks. Proceedings. ICCON IEEE International Conference on Control and Applications. doi:10.1109/iccon.1989.770552.
[45] Zhuang, M., Atherton, D. P. (1991). Optimal PID Controller Settings Using Integral Performance
[46] Criteria. 1991 American Control Conference. doi:10.23919/acc.1991.4791963.
[47] Bratton, D., Kennedy, J. (2007). Defining a Standard for Particle Swarm Optimization. 2007 IEEE Swarm Intelligence Symposium. doi:10.1109/sis.2007.368035
[48] Liu, J. (2013). Radial Basis Function (RBF) Neural Network Control for Mechanical Systems. doi:10.1007/978-3-642-34816-7
[49] Kostanic, I., Ham, F. (n.d.). Radial basis function neural network for regulation of nonlinear systems. Proceedings of SOUTHEASTCON 96. doi:10.1109/secon.1996.510101
[50] Ko, C. (2012). Identification of non-linear systems using radial basis function neural networks with time-varying learning algorithm. IET Signal Processing, 6(2), 91. doi:10.1049/iet-spr.2011.0025.
[51] Isermann, R., Münchhof, M. (n.d.). Identification of Dynamic Systems. Retrieved from https://www.springer.com/de/
[52] Chen, Liu, Wen-Cheng. (2015, June 09). Water Quality Modeling in Reservoirs Using Multivariate Linear Regression and Two Neural Network Models.
[53] Duriez, T., Brunton, S., Noack, B. R. (n.d.). Machine Learning Control – Taming Nonlinear Dynamics and Turbulence. Retrieved from https://www.springer.com/us/book/9783319406237.
[54] Hussain, M. A. (1999). Review of the applications of neural networks in chemical process control — simu- lation and online implementation. Artificial Intelligence in Engineering, 13(1), 55-68. doi:10.1016/s0954- 1810(98)00011-9
[55] Ismail, M. M. (2012). Adaptation of PID controller using AI technique for speed control of isolated steam turbine. 2012 Japan-Egypt Conference on Electronics, Communications and Computers. doi:10.1109/jec- ecc.2012.6186962
[56] Henson, M. A., Seborg, D. E. (1992). Nonlinear Adaptive Control of a pH Neutralization Process. 1992 American Control Conference. doi:10.23919/acc.1992.4792608.
[57] Yeh, W., Lin, Y., Chung, Y. Y., Chih, M. (2010). A Particle Swarm Optimization Approach Based on Monte Carlo Simulation for Solving the Complex Network Reliability Problem. IEEE Transactions on Reliability, 59(1), 212-221.
[58] Getting Started with Simulink. (2015). Modeling of Digital Communication Systems Using Simulink®, 1-26. doi:10.1002/9781119009511.ch1.

Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 8, No. 6: 7252-7272, 2024
DOI: 10.55214/25768484.v8i6.3576