 doi:10.20944/preprints202004.0410.v1

Article

A Self-Tuning NPID Control Method for FOPTD Processes

Hinsermu A. Garbaabaa[1], Million G. Geda[2], Minyamer G. Wase[3], Selvarasu Ranganathan[4], Gang-Gyoo Jin[5] and Yung-Deug Son[6,*]

[1] School of Electrical Engineering and Computing, Adama Science and Technology University (ASTU), Adama, Ethiopia; hialex98@gmail.com
[2] School of Electrical Engineering and Computing, **ASTU**, Adama, Ethiopia; mgaredow@gmail.com
[3] School of Electrical Engineering and Computing, ASTU, Adama, Ethiopia; minyamer1298@gmail.com
[4] School of Electrical Engineering and Computing, ASTU, Adama, Ethiopia; selvarasunaveen@gmail.com
[5] School of Electrical Engineering and Computing, ASTU, Adama, Ethiopia; gjin30@gmail.com
[6] Department of Mechanical Facility Control Engineering, Korea University of Technology and Education, Cheonan, Chungnam, 31253, Korea; ydson@koreatech.ac.kr
* Correspondence: ydson@koreatech.ac.kr; Tel.: +82-41-560-1297

Abstract: Owing to the time-varying characteristics and nonlinearities of industrial processes, control has higher difficulties and results in challenges for advanced technology. In this paper, a self-tuning controller that includes a nonlinear proportional-integral-derivative (NPID) control function as well as a self-tuning function is proposed for first-order plus time delay (FOPTD) process control. The NPID control function is implemented using the nonlinear PID controller whose optimum parameters are adapted by a neural network (NN). The self-tuning function is able to identify the process dynamics using a short period of process behavior and tune NPID parameters based on the identified parameters. The advantage of the proposed method is validated with a set of simulation works on three processes and the comparison results are presented.

Keywords: Self-tuning control; NPID controller; Neural network; FOPTD Process; Genetic algorithm; Parameter estimation

1. Introduction

The use of proportional-integral-derivative (PID) control has a long history in control engineering and despite significant advancements in the field of control, the PID controller is still used in a wide variety of control systems due to its simplicity, robustness, and applicability [1-4]. The key feature of designing a PID controller is the determination of three gains of PID controller, i.e. proportional gain K_p, integral gain K_i, and derivative gain K_d to achieve more desirable closed-loop performance. For industrial processes, tuning methods of Ziegler–Nichols [1], IMC [2] and so on are frequently used to find these gains. The relay feedback technique guarantees obtaining stable oscillations in a closed-loop system through direct field experiments [3]. There are several modified versions of these tuning rules available [4] and their improved responses have shown that they were successful for some types of processes.

However, industrial processes often present characteristics such as time-varying and nonlinearities and although being initially well adjusted, they should be periodically retuned to maintain the desired closed-loop behavior. For this reason, various methods have been proposed for implementing auto-tuning and self-tuning PID controllers [5-8]. An improved phase angle margin auto-tuning method for the first-order plus time delay (FOPTD) model was proposed [5]. An auto-tuning procedure of a PID controller with derivative filter based on evolutionary algorithms and its on-line implementation were presented [6]. A self-tuning controller based on the method of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

frequency identification and in-tended for controlling plants with time-varying parameters under arbitrary bounded exogenous disturbances was proposed [7]. Wahid and Hassan [8] proposed a self-tuning fuzzy PID controller for aircraft pitch control.

Recently, research works on nonlinear PID control using nonlinear characteristics to modify the traditional linear PID controller have been actively conducted [9-14]. NPID controllers can be various forms depending on the application strategy of nonlinearities. Seraji introduced a class of nonlinear PID controllers which consist of a nonlinear gain in cascade with a linear fixed gain PID controller to scale the error [9,10]. Korkmaz et al. suggested a nonlinear PID controller with three nonlinear gains characterized by the Gaussian error function, and also suggested a parameter tuning method using a genetic algorithm (GA) [11]. Han proposed a nonlinear PID controller with nonlinear gain functions combining $e(t)$, $\int e(t)dt$ and $\dot{e}(t)$ to achieve a better tracking and better noise rejection [12]. So et al. presented two nonlinear PID controllers with a nonlinear function described as a Takagi-Sugeno fuzzy model for a liquefied natural gas (LNG) regasification system [13]. Jin and Son introduced an NPID controller in which a nonlinear gain is coupled in series with the integral action to scale the error and three tuning rules for the FOPTD model based on dimensional analysis and a GA [14].

This paper presents a self-tuning controller including a NPID control function as well as a self-tuning function that tries to maintain optimal closed-loop performance. The NPID control function is implemented using a NN that emulates one of the tuning rules of the nonlinear PID controller proposed by the authors [14]. The nonlinear PID controller consists of the linear PD-term and the nonlinear I-term employing a nonlinear gain in cascade with the integral action of a conventional PID controller. The optimum parameters are obtained based on dimensional analysis and a GA. With the operator's intervention, the tuning function identifies a FOPTD model for a given process for a short period, recommends the controller parameters based on the estimated process parameters, and updates them. The effectiveness of the proposed method is evaluated over a set of three processes.

1.1. Structure of the nonlinear PID controller

Through the previous work [14], the authors proposed a nonlinear PID (NPID) controller which consists of the linear PD-term and the nonlinear I-term employing a nonlinear gain in cascade with the integral action of a conventional PID controller. Figure 1 shows the closed-loop control system with the NPID controller.

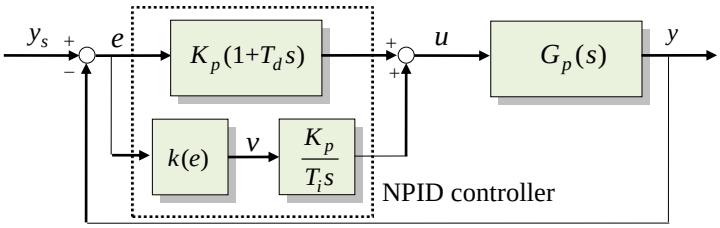


Figure 1. A NPID control system.

In Figure 1, $G_P(s)$ denotes the process; y_s, y and u are the setpoint (SP), the process variable (PV) and the control input, respectively; e is the error (e= SP – PV).

The time–domain equation of the NPID controller is given by

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int v(t)dt + T_d \frac{de(t)}{dt}], \quad (1)$$

where K_p, T_i and T_d are the proportional gain, the integral time and the derivative time, respectively. The scaled error $v(t)$ in the integral term is described by

$$v(t) = k(e)e(t). \quad (2)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

$k(e)$ is a nonlinear gain given by

$$k(e) = \exp(-\frac{e^2}{2\Delta y_s^2}), \tag{2}$$

where Δy_s denotes the SP change, that is, the difference between the current SP and the previous SP. An example of $k(e)$ is illustrated in Figure 2 with the typical values of Δy_s= 0.5, 1, and 1.5. Its shape looks like the bell-shaped curve with a center 0. A small Δy_s expects that e will be clustered around the center, whereas a large Δy_s does that e will be spread out over a large range of values. It can be expected that about 99.7% of values of e will be within ±3 times Δy_s.

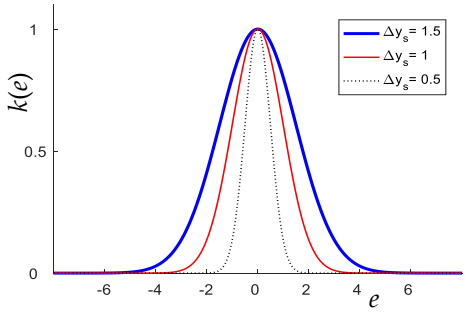


Figure 2. Shapes of $k(e)$ versus e for different Δy_s.

It was demonstrated in [14] that the use of $k(e)$ in the integral loop can enhance the controller performance in terms of swiftness and closeness of the response without excessive control effort.

1.2. Tuning of the NPID controller

The NPID controller given by Equations (1) - (3) can be applied to linear and nonlinear processes and properly tuned by the use of an evolutionary algorithm to minimize a given performance index. The authors proposed three tuning rules for processes that can be simply approximated to a FOPTD model as

$$\tau \frac{dy(t)}{dt} + y(t) = Ku(t - L). \tag{4}$$

where K, τ, and L denote the steady-state gain, the time constant, and the time delay of the process, respectively.

Defining dimensionless variable $t' = t/\tau$ and letting $u(\tau t')= \bar{u}(t')$, $y(\tau t')= \bar{y}(t')$, and $e(\tau t')= \bar{e}(t')$ to simplify notation gives Equation (1) and Equation (4) in the dimensionless form as

$$\bar{u}(t') = K_p[\bar{e}(t') + \frac{\tau}{T_i}\int \bar{v}(t')dt' + \frac{T_d}{\tau}\frac{d\bar{e}(t')}{dt'}], \tag{5}$$

$$\frac{d\bar{y}(t')}{dt'} + \bar{y}(t') = K\bar{u}(t' - \frac{L}{\tau}). \tag{6}$$

The tuning rules were obtained by solving optimization problems formulated to minimize the performance indices in terms of the integral of the square value of the error (ISE), integral of the absolute value of the error (IAE), and integral of the time weighted absolute value of the error (ITAE). They are listed in Tables 1-2 [14].

Table 1. Tuning rules for setpoint tracking based on ISE, IAE and ITAE ($0.01 \le L/\tau < 1$).

Performance	**Dimensionless parameters**		
Index	$\boldsymbol{KK_p}$	$\boldsymbol{T_i/\tau}$	$\boldsymbol{T_d/\tau}$
ISE	$1.2886(\frac{L}{\tau})^{-0.9182}$	$0.9217 + 0.2375\frac{L}{\tau}$	$0.4302(\frac{L}{\tau})^{0.9645}$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

IAE	$1.0350(\frac{L}{\tau})^{-0.9327}$	$0.9465+0.1398\frac{L}{\tau}$	$0.3527(\frac{L}{\tau})^{0.9406}$
ITAE	$1.0019(\frac{L}{\tau})^{-0.9457}$	$0.8723+0.1848\frac{L}{\tau}$	$0.3401(\frac{L}{\tau})^{0.9799}$

Table 2. Tuning rules for setpoint tracking based on ISE, IAE and ITAE ($1 \le L/\tau \le 3$).

Performance Index	Dimensionless parameters		
	KK_p	T_i/τ	T_d/τ
ISE	$1.3362(\frac{L}{\tau})^{-0.5488}$	$0.8419+0.3190\frac{L}{\tau}$	$0.4137(\frac{L}{\tau})^{0.75}$
IAE	$1.0822(\frac{L}{\tau})^{-0.5495}$	$0.8237+0.2692\frac{L}{\tau}$	$0.3331(\frac{L}{\tau})^{0.6831}$
ITAE	$1.0093(\frac{L}{\tau})^{-0.5198}$	$0.7813+0.2781\frac{L}{\tau}$	$0.2878(\frac{L}{\tau})^{0.7317}$

2. Design of a Self-Tuning Controller

In the previous section, a brief overview for the NPID controller and tuning rules for the FOPTD model was given. This section deals with the problem of designing a self-tuning control scheme that consists of the NPID controller, a neural net-work-based tuning rule emulator, and a process parameter estimator.

2.1. Structure of the self-tuning controller

In general, the fixed-parameter PID controller is robust to process parameter changes to some extent, but its performance may be degraded when the process transitions from one operating range to another. In this case, a mechanism is necessary for coping with changes in the process. Figure 3 shows the structure of the proposed self-tuning controller. If an operator activates the self-tuning function, self-tuning is performed for a predetermined time. It follows the procedure of (1) collecting process I/O data, (2) estimating parameters of the FOPTD model, (3) calculating and updating the values of the NPID controller parameters.

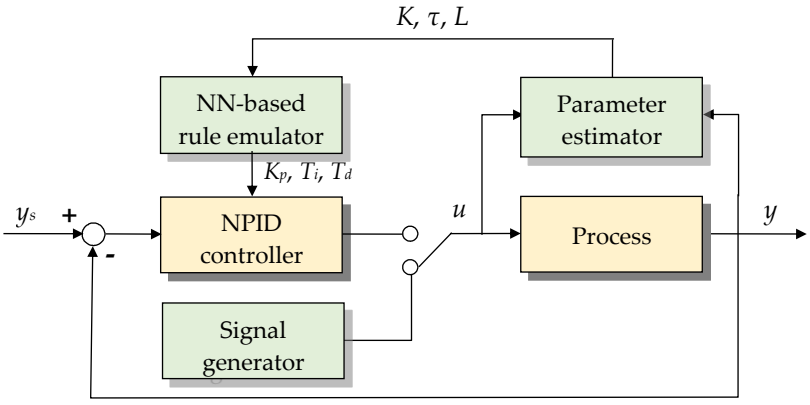


Figure 3. The proposed self-tuning NPID control system.

2.2. Process parameter estimator

If the operator switches to the self-tuning mode with the normal control function disabled, a signal from the signal generator is applied to the model as well as the process. In the real world it is

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

good practice to perform estimation with as big a step signal as possible, over the operating range, and in both the increasing and decreasing directions.

The FOPTD model in Equation (4) as an approximation of complex processes is rewritten as follows:

$$(1+\tau s)Y(s) = Ke^{-Ls}U(s)\,. \tag{7}$$

A discrepancy, that is, the error between the process and the model due to modelling errors, parameter variations or disturbances can be represented by

$$E(s) = (1+\tau s)Y(s) - Ke^{-Ls}U(s)\,. \tag{8}$$

Figure 4 shows a block diagram for estimating K, τ, and L.

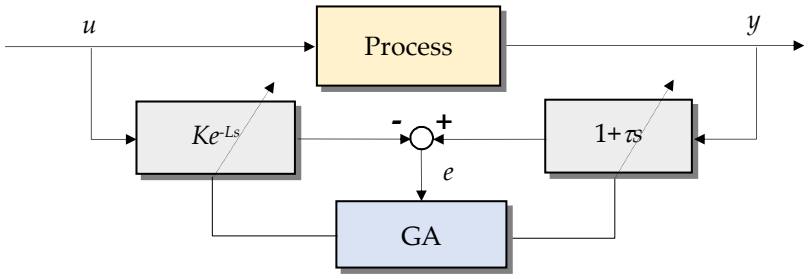


Figure 4. Block diagram for process parameter estimation.

While both the input and the output signals of the process are applied to the model connected in parallel with the process, a GA continuously adjusts K, τ, and L such that the dynamics of the model is as close as possible to that of the process. In this way the GA searches for the parameters of the model to minimize the following objective function.

$$J(\boldsymbol{\phi}) = \int_{(k-W+1)T_s}^{kT_s} |e(t)|\,dt \tag{9}$$

where $e(t)$ is the error between the process and the model, $\boldsymbol{\phi} = [K\ \tau\ \ L]^{\mathrm{T}} \in R^3$ is the parameter vector to be adjusted, T_s is the sampling time, and W is the size of the data window, which is properly compromised between the precision of the estimate and the computation time.

2.3. NN-based tuning rule emulator

In general, self-tuning controllers need a mechanism that has the ability to calculate controller parameters based on process information. To do this, a neural network (NN) is used to learn the NPID controller settings with a set of precalculated data. As a tuning rule emulator, Figure 5 shows the structure of an NN with one input, three outputs, and one hidden layer of neurons. The input of the NN is $x(= L/\tau)$ and the outputs are $y_1(= KK_p)$, $y_2(= T_i/\tau)$, and $y_3(= T_d/\tau)$. The number of neurons in the hidden layer is 20. As activation functions that determine the characteristics of neurons in this paper, a hyperbolic tangent function in the range [-1, 1] for the hidden layer and a linear function for the output layer are employed.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

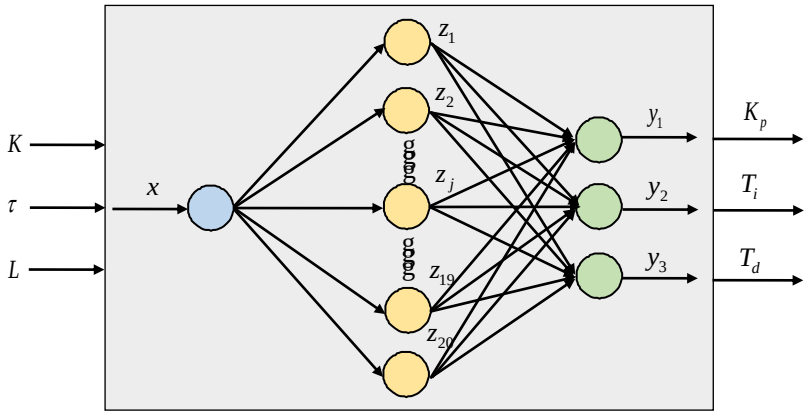


Figure 5. Structure of the NN-based emulator

When an input x is applied to the NN, the output z_j, $j\in[1, 20]$ of the hidden layer is obtained as follows:

$$z_j = f(x \cdot w_j^a + b_j), \quad j\in[1,20] \tag{10}$$

where w_j^a is the weight between the jth neuron and the input layer and b_j is the bias of the jth neuron of the hidden layer. Thus, the final output y_j is

$$y_j = g(\sum_{i=1}^{20} z_i w_{ji}^b + b_j), \quad j\in[1,3]. \tag{11}$$

where w_{ji}^b is the weight between the jth neuron of the output layer and the ith neuron of the hidden layer, and b_j is the bias of the jth neuron of the output layer.

Supervised learning of a NN adjusts its weights and biases with a set of training data to reduce the difference between the network output and the target output. For learning of the NN-based emulator, we ran the program 20 times with different random seeds to obtain optimum KK_p, T_i/τ, and T_d/τ which minimize the IAE performance index to the changes of L/τ from 0.01 to 3. A total of 580 data sets were used. The learning rate was set to 0.01, and learning was terminated when the *rms* learning error was 0.01 or less. Best validation performance was 1.1701×10^{-5} at epoch 300.

Figure 6 shows the difference between the outputs of the NN emulator and those of the tuning rules in Tables 1-2. It can be seen that the results of the NN emulator are similar to those of the tuning rules.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

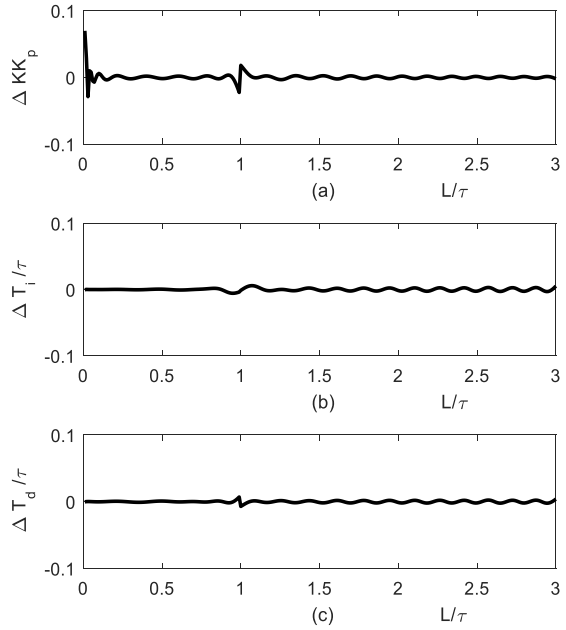


Figure 6. Error between the target curve and the trained curve (IAE): **(a)** ΔKK_p; **(b)** $\Delta T_i/\tau$; **(c)** $\Delta T_d/\tau$.

3. Simulation Results

For the purpose of simulation work, three processes are considered and the responses of the proposed method tuned in terms of IAE performance index are compared with those of the PID settings proposed by the Ziegler-Nichols setting (hereafter referred to as PID-ZN) [1], the IMC setting (hereafter referred to as PID-IMC) [2], Anwar, Shamsuzzoha and Pan (hereafter referred to as PID-ASP) [15], Peng and Wu (hereafter referred to as PID-Peng)[16], and Yang, Xu and Chiu (hereafter referred to as PID-YXC)[17].

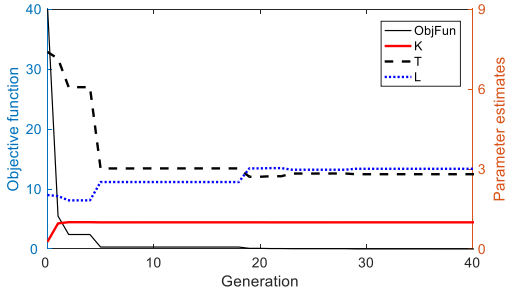
The comparative analysis is obtained in terms of various performance specifications such as rise time (t_r), overshoot (M_p), settling time (t_s), the integral of absolute error $IAE = \int_0^\infty |e(t)|dt$ and the total variation $TV = \sum_{i=1}^{\infty} |u(i+1) - u(i)|$ of the control input where $u(i)$ denotes the control input at the ith instance. The smaller the IAE value, the faster the response speed and the higher accuracy, and the smaller the TV value, the smoother the control input.

3.1. Process 1

Consider a 1st-order process with a time delay:

$$G_p(s) = \frac{e^{-3s}}{1+2.8s} \tag{12}$$

When the operation mode is switched to the parameter estimation, the estimator starts by applying a step signal to both the process and the model for a predetermined time. Figure 7 illustrates the parameter estimation process.



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

Figure 7. Evolutionary parameter estimation.

For the process in (12), three parameter values K= 1, τ= 2.8001, L= 3.0004 were obtained at the 40th generation. With these values, the emulator gives the controller parameters K_p= 1.1097, T_i= 3.1321, and T_d= 1.0437. When the self-tuning function has been performed, the NPID controller was reset with these new parameters. The proposed method was compared with the three methods. The PID settings for this process are K_p= 1.285, T_i= 7.2191, and T_d = 1.0895 for PID-ZN [1], K_p = 0.651, T_i = 2.8840, and T_d = 0.8568 for PID-Peng [16], and K_p = 1.1837, T_i = 3.9326, and T_d = 0.7380 for PID-IMC [2]. Figure 8 shows a comparison of the responses.

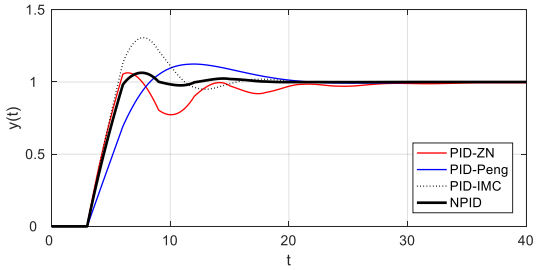


Figure 8. The unit step responses of the four methods on Process 1.

It is obvious that the response is faster when using the NPID controller, without significant increase of the overshoot. More precisely compared with the other methods in Table 3, the overshoot of the NPID controller is smaller while IAE is least, and both the settling time and TV are reasonable.

Table 3. Quantitative comparison of the setpoint tracking performances for Process 1.

Tuning Method	SP tracking performance				
	t_r	M_p	t_s	*IAE*	*TV*
PID-ZN	2.5275	6.3244	26.5434	5.7320	5.3678
PID-Peng	4.5585	19.3425	12.4371	6.2311	1.8533
PID-IMC	2.3678	30.7155	14.6769	5.5235	4.0635
NPID-IAE	2.7694	6.3836	15.2289	4.7034	3.4146

The NPID controller can mitigate integrator windup to some extent, especially by scaling down large errors. Figure 9 shows the unit step responses of the NPID controller and the PID controller when the value of the control input is limited to ±1.3. Here again, the performance of NPID controller is better than those of the PID controllers tuned by the other methods.

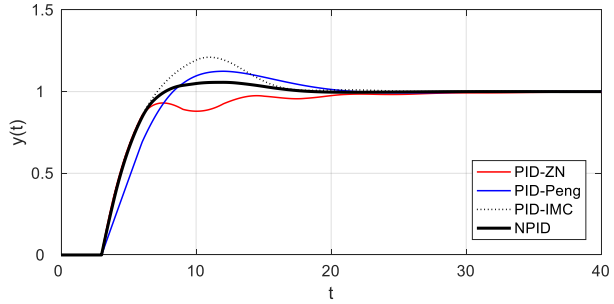


Figure 9. The unit step responses under input saturation on Process 1.

3.2. Process 2

Consider a 5th-order process with a time delay:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

$$G_p(s) = \frac{e^{-8s}}{(2s+1)^3(s+1)^2} \tag{13}$$

Similar to the previous simulation, the parameters of the FOPTD model obtained by the estimator result in K= 1, τ= 4.1195, and L= 12.2296. With these estimates, the NN-based emulator produces K_p= 0.5951, T_i= 6.6865, and T_d= 2.8856. The PID settings for this process are K_p= 0.543, T_i= 9.8727, and T_d= 4.2357 for PID-ASP [15], K_p= 0.63, T_i= 10.5, and T_d= 2.7698 for PID-YXC [17], and K_p= 0.56, T_i= 8.8889, and T_d= 2.35 for PID-IMC [2].
Figure 10 shows a comparison of the unit step responses of the PID and NPID control systems.

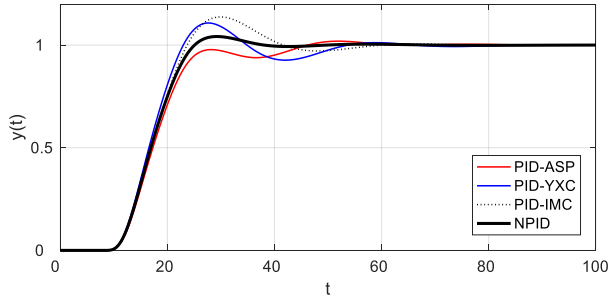


Figure 10. The unit step responses of the four methods on Process 2.

It is shown in Figure 10 that the response of the NPID controller is clearly better than those of the others. Table 4, which lists the quantitative results of the aforementioned methods, proves this.

Table 4. Quantitative comparison of the setpoint tracking performances for Process 2.

Tuning Method	SP tracking performance				
	t_r	M_p	t_s	*IAE*	*TV*
PID-ASP	13.5305	-	43.5887	18.6414	1.3733
PID-YXC	10.6384	10.7793	50.5933	18.4808	1.6732
PID-IMC	11.2270	13.7229	52.7756	18.9329	1.4917
NPID-IAE	12.0363	4.1820	34.2391	17.6170	1.2722

3.3. Process 3

Finally, consider a 20th-order process:

$$G_p(s) = \frac{1}{(s+1)^{20}} \tag{14}$$

In a similar fashion, the process is estimated as the FOPTD model and the result gives K= 1, τ= 5.1428, and L= 15.2924. Based on these values, the NPID controller settings set by the NN emulator are K_p= 0.6205, T_i= 8.3513 and T_d= 4.2367.
Similarly, the performance of the NPID controller was compared with those of three PID controllers. The settings of PID-ASP, PID-YXC and PID- IMC are K_p= 0.525, T_i= 9.5455 and T_d= 3.1619 [15], K_p= 0.62, T_i= 11.9231 and T_d= 3.5645 [17], K_p= 0.55, T_i= 10 and T_d= 3.4 [2], respectively. The unit step responses are shown in Figure 11.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 April 2020 doi:10.20944/preprints202004.0410.v1

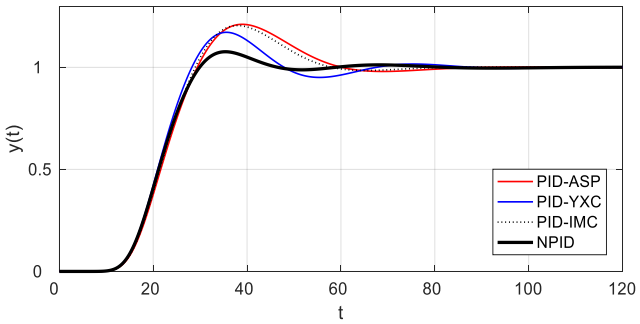


Figure 11. The unit step responses of the four methods on Process 3.

Again, it can be seen that the response of the NPID controller is superior to those of the other methods. Table 5, which summarizes the quantitative results, also shows that the NPID controller provides better performances than the PID controllers tuned by the other methods.

Table 5. Quantitative comparison of the setpoint tracking performances for Process 3.

Tuning Method	SP tracking performance				
	t_r	M_p	t_s	*IAE*	*TV*
PID-ASP	14.2455	21.0913	69.4321	12.7703	1.5654
PID-YXC	13.3755	17.1826	63.4094	11.8870	1.6955
PID-IMC	13.9138	20.5076	55.0202	12.4261	1.5644
NPID-IAE	14.5507	7.6566	43.5139	11.1414	1.3757

3. Conclusion

In this paper, a self-tuning controller composing the NPID control function and a self-tuning function was proposed for process control. The NPID control function was implemented using the nonlinear PID controller whose optimum parameters were adapted by a NN. The self-tuning function was able to identify the process dynamics using a short period of process behavior and tune NPID parameters based on the identified parameters. Simulation works on the three processes showed that the proposed method provided better performance in terms of various specifications than the other methods.

Acknowledgments: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07048954).

Author Contributions: G. Jin conceptualized the idea of this research work. The simulation works were carried out by M. Geda and M. Wase under the supervision of G. Jin and Y. Son. Results were analyzed by Y. Son and G. Jin. The paper was written by H. Garbaabaa, G. Jin, S. Ranganathan and Y. Son with different degrees of contribution.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ziegler, J.G.; Nichols, N.B. Optimum settings for automatic controllers. *Trans. American Society of Mechanical Engineers*. **1942**, 64, 759–768.
2. Rivera, D. E.; Morari, M.; Skogestad, S. Internal model control: PID controller design. *Ind. Eng. Chem. Process Des*. Dev. **1986**, 25, 252-265.
3. Åström, K. J.; Hägglund, T.; Hang, C. C. and Ho, W. K. Automatic Tuning and Adaption for PID Controllers. *A Survey. Control Eng. Practice*. **1993**, 1, 699-714.
4. O'Dwyer, A. Handbook of PI and PID Controller Tuning Rules. *second ed. Imperial College Press*, London, **2006**.

5. Zeng, D.; Zheng, Y.; Luo, W.; Hu, Y.; Cui, Q.; Li, Q.; Pen, C. Research on Improved Auto-Tuning of a PID Controller Based on Phase Angle Margin. *Energies*, **2019**, 12, 1704-1719.
6. Reynoso-Meza, G.; Sanchis, J.; Herrero, J. M.; Ramos, C. Evolutionary auto-tuning algorithm for PID controllers. In: 2012 IFAC Conference on Advances in PID Control, Brescia, Italy, 28-30 March 2012.
7. Aleksandrov, A. G.; Palenov, M. V. Self-tuning PID/I controller. *Automation and Remote Control 10*, **2011**, 4-18.
8. Wahid, N.; Hassan, N. Self-Tuning Fuzzy PID Controller Design for Aircraft Pitch Control. In: 2012 Third International Conference on Intelligent Systems Modelling and Simulation, Kota Kinabalu, Malaysia, 19-24 February 2012.
9. Seraji, H. A new class of nonlinear PID controllers. In: Proceedings of the 5th IFAC Symposium on Robot Control 1997 (SYROCO '97), Nantes, France, 65-71 September 1997.
10. Seraji, H. A new class of nonlinear PID controllers with robotic applications. *Journal of Robotic Systems*, **1998**, 15, 161–181.
11. Korkmaz, M.; Aydogdu, O.; Dogan, H. Design and performance comparison of variable parameter nonlinear PID controller and genetic algorithm based PID controller. In: Proc. Int. Symp. Innovations in Intelligent Systems and Applications (INISTA), Trabzon, Turkey, 1-5 July 2012.
12. Han, J. From PID to Active Disturbance Rejection Control. *IEEE Transactions on Industrial Electronics*, **2009**, 56, 900-906.
13. So, G.; Yi, H.; Son, Y.; Jin, G. Temperature Control of a Regasification System for LNG-fuelled Marine Engines Using Nonlinear Control Techniques. *International Journal of Control, Automation and Systems*, **2018**, 16, 3047-3054.
14. Jin, G.; Son, Y. Design of a Nonlinear PID Controller and Tuning Rules for First-Order Plus Time Delay Models. *Studies in Informatics and Control*, **2019**, 28, 157-166.
15. Anwar, M. N.; Shamsuzzoha, M.; Pan, S. A Frequency Domain PID Controller Design Method Using Direct Synthesis Approach. *Chemical Engineering*, **2015**, 40, 995-1004.
16. Peng, H., Wu, S.-C. *Journal of Cent. South Univ. Technology* (China) , **2000**, 7, 165-169.
17. Yang, X.; Xu, B.; Chiu, M. S.; PID controller design directly from plant data. *Ind. Eng. Chem. Res*, **2011**, 50, 1352-1359.