International Journal of Recent Advances in Multidisciplinary Topics
Volume 6, Issue 11, November 2025
www.ijramt.com | E-ISSN: 2582-7839 | RESAIM Publishing (www.resaim.com)

A Noble PI-PD Controller Design Based on Extended State Space Predictive Functional Control Using Improved Grasshopper Optimization Algorithm

Chung Hyok Kang[1*], Kukhuan Jang[2], Jinsong Ri[3]
[1,2,3]*Faculty of Metallic Engineering, Kim Chaek University of Technology, Pyongyang, Democratic People's Republic of Korea*

***Abstract*: In proportional-integral-proportional-derivative (PI-PD) controller design based on extended state space predictive functional control (ESSPFC), the proper tuning of weighting matrix of cost function has a direct influence on the controller performance. However, the analytical method to determine it has not been unknown. To solve this problem, in this paper, an improved grasshopper optimization algorithm (GOA) is used to improve the performance of controller by optimizing the weighting matrix of cost function. To overcome the drawbacks of standard GOA, a linear shrinking coefficient is replaced by nonlinear shrinking coefficient, a mutation operation is adopted, and position updating formula is modified. The performance of proposed method is tested and compared with other methods based on PFC. Simulation results show that the proposed design method is much superior to other methods in terms of the set-point tracking, disturbance rejection and robustness.**

***Keywords*: PI-PD control, Predictive functional control, Grasshopper optimization algorithm, Extended State space model.**

1. Introduction

Traditional proportional-integral-derivative (PID) controllers may not obtain the desired control performance for large time delay processes [1]-[4].

Meanwhile, PI-PD controller is an improved version of PID controller, which can effectively improve the control performance of large time delay process [5]-[7]. However, while the PI-PD controller has such advantages, it also has some difficulty in parameter tuning. In some papers, the parameter tuning methods of PI-PD controller have been proposed. Reference [8] proposed a graphical parameter calculation method, in which the estimated values of parameters are only available and the parameters are fixed, so that they cannot be adapted to the operating conditions of the system. Reference [9] proposed a method of obtaining the parameters of PID controller and then introducing additional parameters to obtain the parameters of PI-PD controller, but its computational process is complex. The above facts show that the PI-PD controller has a wide range of applications but is limited due to the difficulty in parameter tuning.

Meanwhile, even if the parameters of PI-PD controller are determined by the above-mentioned methods, the PI-PD may not ensure the desired performance because of the large time delay and uncertainties of processes [10], [11].

Recently, predictive function control (PFC) has been widely applied in practice since it is suitable for coping with large time delays and uncertainties effectively [12]-[19]. Moreover, PFC does not require high computational capability and can be implemented in real-time conditions using low-cost hardware [20]. Therefore, combining PFC with PI-PD control may be a good choice.

Some papers combining PFC with PI-PD control are available. Reference [21] proposed a PI-PD design method based on PFC optimization for the case that the process model is given as first-order plus dead time (FOPDT) model. On the other hand, Reference [22] proposed a PI-PD design method based on PFC using an extended state space model. But they had not mentioned the method of tuning the weighting matrix Q of cost function which has a great influence on the controller performance. Some papers introduced methods for optimizing the weight matrix by using various optimization algorithms. Reference [23] proposed a weighted matrix optimization method using GA to improve the performance of PID designed based on state space PFC. In addition, Reference [24] suggested a weighted matrix optimization method using extremal optimization.

Recently, Grasshopper optimization algorithm (GOA) has been widely applied to optimization problems. Since GOA has a simple theory foundation, easy implementation and excellent search performance, many researchers are interested in it [25]-[30].

In this paper, to design the optimal PI-PD controller, the weight matrix Q in PI-PD controller design based on ESSPFC is optimized by using the improved GOA. The proposed controller simultaneously has a simple structure of the traditional PID controller with the excellent performance of PFC.

The rest of paper is organized as follows. Section 2 presents

*Corresponding author: kch9213@star-co.net.kp

the design method of PI-PD controller based on PFC by representing the plant as the extended state space model. Section 3 gives the theoretical basis of standard GOA, introduces some improvements of GOA, and describes the optimization method of weight matrix Q based on the improved GOA. The simulation results are presented in Section 4 and the conclusions is given in Section 5.

2. ESSPFC Based PI-PD Design

A. Extended State Space Model

The controlled process is supposed to be described as follows [22], [23].

$$\begin{aligned} &y(k+1)+F_1y(k)+F_2y(k-1)+\cdots+F_py(k-p+1)= \\ &=H_1u(k)+H_2u(k-1)+\cdots+H_qu(k-q+1) \end{aligned} \tag{1}$$

where $y(k)$ and $u(k)$ are the output and input of the process at time instant k, respectively, p and q are the output and input orders, respectively, and F_i (i=1, 2, …, p) and H_j (j=1, 2, …, q) are the corresponding coefficients of process model.

The process model can be expressed through the back shift operator Δ as follows:

$$\begin{aligned} &\Delta y(k+1)+F_1\Delta y(k)+\cdots+F_p\Delta y(k-p+1)= \\ &=H_1\Delta u(k)+H_2\Delta u(k-1)+\cdots+H_q\Delta u(k-q+1) \end{aligned} \tag{2}$$

As in [22, 23], a state space vector $\Delta x_m(k)^T$ is chosen as:

$$\Delta x_m(k)^T=\begin{bmatrix}\Delta y(k),\Delta y(k-1),\cdots,\Delta y(k-p+1), \\ \Delta u(k-1),\Delta u(k-2),\cdots,\Delta u(k-q+1)\end{bmatrix} \tag{3}$$

where the dimension of $\Delta x_m(k)^T$ is m=p+q-1.

Then a state space model can be obtained as follows:

$$\begin{cases}\Delta x_m(k+1)=A_m\Delta x_m(k)+B_m\Delta u(k) \\ \Delta y(k+1)=C_m\Delta x_m(k+1)\end{cases} \tag{4}$$

where

$$A_m=\begin{bmatrix} -F_1 & -F_2 & \cdots & -F_{p-1} & -F_p & H_2 & \cdots & H_{q-1} & H_q \\ 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

$$B_m=\begin{bmatrix}H_1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0\end{bmatrix}^T$$

$$C_m=\begin{bmatrix}1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0\end{bmatrix}$$

Defining the reference trajectory as $r(k)$, the output error is obtained as:

$$e(k)=y(k)-r(k) \tag{5}$$

From Eqs. (4) and (5), $e(k+1)$ is derived as follows.

$$e(k+1)=e(k)+C_mA_m\Delta x_m(k)+C_mB_m\Delta u(k)-\Delta r(k+1) \tag{6}$$

The following extended state variable $z(k)$ is introduced.

$$z(k)=\begin{bmatrix}\Delta x_m(k) \\ e(k)\end{bmatrix} \tag{7}$$

Then the extended state space model is as follows:

$$z(k+1)=Az(k)+B\Delta u(k)+C\Delta r(k+1) \tag{8}$$

where $A=\begin{bmatrix}A_m & 0 \\ C_mA_m & I\end{bmatrix}$; $B=\begin{bmatrix}B_m \\ C_mB_m\end{bmatrix}$; $C=\begin{bmatrix}0 \\ -1\end{bmatrix}$ and 0 is a zero vector with dimension $m \times 1$.

B. ESSPFC Dased PI-PD Design

The future state variable from time instant k is as follows [22].

$$z(k+P)=A^Pz(k)+\psi\Delta u(k)+\theta\Delta R \tag{9}$$

where $\theta=\begin{bmatrix}A^{P-1}C & A^{P-2}C & \cdots & C\end{bmatrix}$; $\psi=A^{P-1}B$;

$$\Delta R=\begin{bmatrix}\Delta r(k+1) & \Delta r(k+2) & \cdots & \Delta r(k+P)\end{bmatrix}^T$$

$$r(k+i)=\beta^iy(k)+\left(1-\beta^i\right)c(k)$$

Here P is the prediction horizon, β is the smoothing factor of reference trajectory, and $c(k)$ is the set-point at time instant k.

The cost function $J(k)$ is defined as

$$\min J(k)=z(k+P)^TQz(k+P) \tag{10}$$

where Q is the weight matrix.

The following incremental PI-PD controller is used.

$$\begin{aligned} u(k)&=u(k-1)+K_p(k)[e_1(k)-e_1(k-1)]+K_i(k)e_1(k) \\ &\quad-K_f(k)[y(k)-y(k-1)]\text{-}K_d(k)[y(k)-2y(k-1)+y(k-2)] \\ &=u(k-1)+K_p(k)[e_1(k)-e_1(k-1)]+K_i(k)e_1(k) \\ &\quad-K_f(k)[y(k)-y(k-1)]-K_d(k)[y(k)-y(k-1)] \\ &\quad+K_d(k)[y(k-1)-y(k-2)] \end{aligned} \tag{11}$$

$$e_1(k) = c(k) - y(k) \tag{12}$$

In Eq. (11), $K_p(k), K_i(k), K_f(k), K_d(k)$ are the proportional coefficient and the integral coefficient in the outer loop, the proportional coefficient and the differential coefficient in the inner loop at time instant *k*, respectively.

In Eq. (12), $e_1(k)$ is the error between the set-point and the actual output value at time instant *k*.

Eq. (11) can be simplified as follows:

$$u(k) = u(k-1) + w(k)^T E(k) \tag{13}$$

where

$$\begin{aligned}
&w(k) = \left[w_1(k),\ w_2(k),\ w_3(k),\ w_4(k)\right]^T\\
&E(k) = \left[E_1(k),\ E_2(k),\ E_3(k),\ E_4(k)\right]^T\\
&w_1(k) = K_p(k) + K_i(k)\\
&w_2(k) = -K_p(k)\\
&w_3(k) = -K_f(k) - K_d(k)\\
&w_4(k) = K_d(k)\\
&E_1(k) = e_1(k)\\
&E_2(k) = e_1(k-1)\\
&E_3(k) = y(k) - y(k-1)\\
&E_4(k) = y(k-1) - y(k-2)
\end{aligned} \tag{14}$$

Taking into account Eqs. (4) to (14) and taking a derivative of the cost function *J*(*k*), the following optimal control law is obtained.

$$w(k) = -\frac{\psi^T Q\left(A^P z(k) + \theta\Delta R\right)E(k)}{\psi^T Q\psi E(k)^T E(k)} \tag{15}$$

Then $K_p(k), K_i(k), K_f(k), K_d(k)$ are obtained as:

$$\begin{aligned}
&K_p(k) = w_1(k) + w_2(k)\\
&K_i(k) = -w_2(k)\\
&K_f(k) = -w_3(k) - w_4(k)\\
&K_d(k) = w_4(k)
\end{aligned} \tag{16}$$

It can be seen that $w(k)$ will be infinite when the control system is reaching the steady state since $e_1(k)$ in $E(k)^T E(k)$ will be almost zero. Then $K_p(k), K_i(k), K_f(k), K_d(k)$ will be infinite, which is unrealistic for PI-PD controller. So $K_p(k), K_i(k), K_f(k), K_d(k)$ are calculated according to the following equation by introducing a small permissible error limitation δ.

$$\begin{cases}
K_p(k) = K_p(k-1)\\
K_i(k) = K_i(k-1)\\
K_f(k) = K_f(k-1)\\
K_d(k) = K_d(k-1)
\end{cases} \quad |e_1(k)| \le \delta$$
$$\begin{cases}
K_p(k) = w_1(k) + w_2(k)\\
K_i(k) = -w_2(k)\\
K_f(k) = -w_3(k) - w_4(k)\\
K_d(k) = w_4(k)
\end{cases} \quad |e_1(k)| > \delta \tag{17}$$

As a result, the control input *u*(*k*) at time instant *k* is obtained from Eq. (13).

3. Optimal PI-PD Design by GOA

From Eq. (15), it can be seen that the weighting matrix *Q* has a great impact on the performance of controller.

The weight matrix *Q* is a diagonal matrix expressed as follow [23], [24].

$$Q = diag\{q_{j1}, q_{j2}, \cdots, q_{jp}, q_{jp+1}, \cdots, q_{jp+q-1}, q_{je}\},\quad 1 \le j \le P \tag{18}$$

where $q_{j1}, q_{j2}, \cdots, q_{jp}$ are the weighting factors associated with the response of process output and $q_{jp+1}, q_{jp+2}, \cdots, q_{jp+q-1}$ are the weighting factors associated with the control input. And q_{je} is the weighting factor on the output error. In general, q_{je} is selected as 1 [23].

In this study, $q_{j1}, q_{j2}, \cdots, q_{jp}, q_{jp+1}, q_{jp+2}, \cdots, q_{jp+q-1}$ are optimized by using the improved GOA to determine the optimal parameters of PI-PD controller.

A. Grasshopper Optimization Algorithm (GOA)

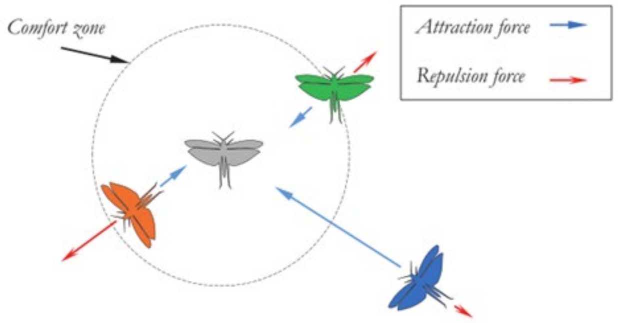

Fig. 1. Conceptual model of the interactions between grasshoppers

GOA is a meta-heuristic algorithm that mimics the swarm behavior of grasshoppers to find food sources in nature [31]. The food search process of the grasshopper's swarm consists of two stage: exploration and exploitation. In the exploration, grasshoppers explore the search area to find promising areas and in the exploitation, they exploit the promising areas to find the target area. The exploration and exploitation of grasshopper's swarm are controlled by the attraction and repulsion that are social interaction forces among grasshoppers.

Attraction forces allow grasshoppers to exploit the promising areas, while repulsion forces enable grasshoppers to explore the search area. The zone where the two kinds of forces are equal is called comfort zone. As the search process preforms, the comfort zone decreases, and finally the grasshopper swarm finds the optimal target area. A conceptual model of the interactions between grasshoppers is illustrated in Fig. 1.

GOA mimics the interaction forces between grasshoppers, and the position of the grasshopper with the best fitness is regarded as the closest to the target. The rests will move to the direction of the target under the social interaction between grasshoppers. Also with the position update of grasshoppers, to balance between exploration and exploitation (i.e. the global search and the local search), the comfort zone will decrease adaptively. Finally, all grasshoppers converge together and find the best solution.

The position updating formula of GOA to mimic such a search process can be modelled mathematically as follows [31], [32].

$$X_i^d(t+1)=c\left(\sum_{j=1,\ j\neq i}^{N} c\frac{ub_d-lb_d}{2}s\left(\left|X_j^d(t)-X_i^d(t)\right|\right)\frac{X_j(t)-X_i(t)}{d_{ij}}\right)+\hat{T}_d \tag{19}$$

where t is thecurrent iteration, N is the swarm size, $X_i^d(t)$ and $X_j^d(t)$ are the position of the ith and the jth grasshoppers in t iteration in the d-dimensional search space, respectively, ub_d and lb_d represent the upper and lower boundaries in the d-dimensional search space, respectively, d_{ij} is the distance between the ith and the jth grasshoppers, defined as $d_{ij}=\left|X_j(t)-X_i(t)\right|$, $\hat{T}_d$ indicates the d-dimensional target position (i.e. the best solution found so far), s is a function to indicate the strength of interaction forces (attraction and repulsion forces), define as follows.

$$s(r)=fe^{\frac{-r}{l}}-e^{-r} \tag{20}$$

where f is the intensity of attraction, l is the attractive length scale, r represents the distance between grasshoppers.

In Eq. (19), c is the shrinking factor to balance exploration and exploitation. If c is large, the dominant activity of GOA is exploration i.e., the global search; if c is small, the dominant activity is exploitation i.e., the local search. To converge the GOA according the number of iterations, c is set as follows.

$$c=c_{\max}-t\frac{c_{\max}-c_{\min}}{T} \tag{21}$$

where $c_{\max}$, $c_{\min}$ are the maximum and the minimum values of c, respectively, T represents the maximum number of iterations. Usually, $c_{\max}=1$, $c_{\min}=0.00001$ [32], [33].

The search process of GOA is similar with PSO. The difference is that in PSO, each particle updates the position by its current position, the global best and the personal best, but as you can see in Eq. (19), each grasshopper updates the position not only by its current position and the global best, but also by the positions of all other grasshoppers. Thus, GOA has a higher search efficiency since the information of all grasshoppers is used in the searching process.

B. *Improvement of GOA*

While GOA has the advantages of simple theoretical basis, easy implementation and fast convergence speed, it also has some disadvantages.

The disadvantage is, first, the linear shrinking factor used in GOA cannot control the search process which consists of the exploration and the exploitation effectively. Grasshoppers cannot converge rapidly to the promising area in the exploration stage and they just wander around the target position in the exploitation stage. The second disadvantage is that GOA is easy to fall into the local optimum.

To overcome these drawbacks of GOA, some modifications of GOA are made. First, instead of the linear shrinking factor, the following nonlinear shrinking factor is introduced.

$$c_v=c_{\min}+\left(c_{\max}-c_{\min}\right)\times\left(\cos^3\left(\frac{\alpha}{2}\right)\right),\quad \alpha=\pi\times\frac{t}{T} \tag{22}$$

Fig. 2 shows the curves of linear and nonlinear shrinking factors according to the iteration number.

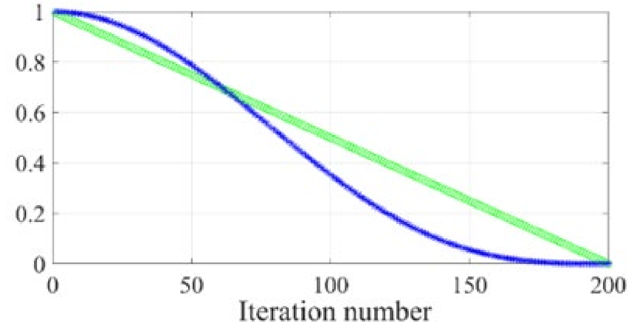

Fig. 2. Curves of linear and nonlinear decreasing factors

By this nonlinear decreasing factor, Grasshoppers can converge rapidly to the promising area in the exploration stage and they don't wander around the target position in the exploitation stage.

In grey wolf optimizer (GWO), the movement direction of individuals is decided by the positions of the best three individuals. The research results show that the algorithm with such mechanism have more superiorities in avoiding the local optimum than the algorithm whose direction is only depended on the best solution [34], [35]. Hence, the position updating formula in Eq. (19) is modified as follows:

$$X_i^d(t+1)=c_v\left(\sum_{j=1,\ j\neq i}^{N} c_v\frac{ub_d-lb_d}{2}s\left(\left|X_j^d(t)-X_i^d(t)\right|\right)\frac{X_j(t)-X_i(t)}{d_{ij}}\right)+ +\frac{\hat{T}_\alpha+\hat{T}_\beta+\hat{T}_\gamma}{3} \tag{23}$$

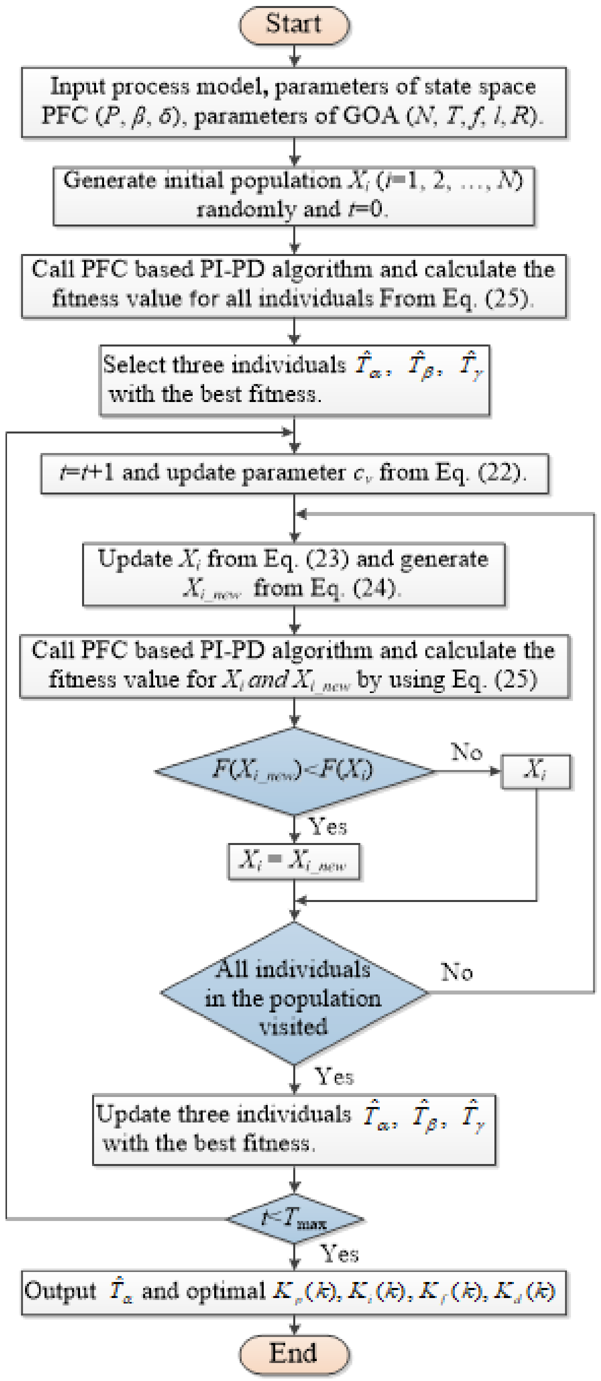
Start
Input process model, parameters of state space PFC (P, β, δ), parameters of GOA (N, T, f, l, R).
Generate initial population X_i (i=1, 2, …, N) randomly and t=0.
Call PFC based PI-PD algorithm and calculate the fitness value for all individuals From Eq. (25).
Select three individuals $\hat{T}_\alpha$, $\hat{T}_\beta$, $\hat{T}_\gamma$ with the best fitness.
t=t+1 and update parameter c_v from Eq. (22).
Update X_i from Eq. (23) and generate X_{i_new} from Eq. (24).
Call PFC based PI-PD algorithm and calculate the fitness value for X_i and X_{i_new} by using Eq. (25)
$F(X_{i\_new})<F(X_i)$
No
X_i
Yes
$X_i = X_{i\_new}$
All individuals in the population visited
No
Yes
Update three individuals $\hat{T}_\alpha$, $\hat{T}_\beta$, $\hat{T}_\gamma$ with the best fitness.
$t<T_{max}$
Yes
Output $\hat{T}_\alpha$ and optimal $K_p(k)$, $K_i(k)$, $K_f(k)$, $K_d(k)$
End

Fig. 3. Flowchart of algorithm to determine the optimal parameters of PI-PD

where $\hat{T}_\alpha$, $\hat{T}_\beta$, $\hat{T}_\gamma$ are the positions of three grasshoppers with the best fitness values so far

Also, to prevent GOA from falling into local optimum, the following mutation operation is introduced at each step of the search process.

$$X_{i\_new} = \frac{\hat{T}_\alpha + \hat{T}_\beta + \hat{T}_\gamma}{3} \times \left(\left(0.5 - rand(0,\ 1)\right) * R + 1\right) \tag{24}$$

where R is a parameter that limits the range of individuals generated by the mutation operation. If R is higher, the global search ability of algorithm is enhanced

In the improved GOA, X_i is replaced by X_{i_new} if the fitness value of X_i calculated by Eq. (23) is larger than the X_{i_new} fitness value calculated by Eq. (24), otherwise X_i is retained.

C. Optimal PI-PD design by optimization of weighting matrix

The fitness function used in GOA is defined as follows:

$$F(X_i) = \sum_{k=0}^{T_{sim}/T_s} k\left|e_1(k)\right| \tag{25}$$

where T_{sim} is the simulation time period, T_s is the sampling time, $X_i = \left\lfloor q_{j1}, q_{j2}, \cdots, q_{jp}, q_{jp+1}, q_{jp+2}, \cdots, q_{jp+q-1} \right\rfloor$ means the search individual (weight factor vector).

The flowchart of algorithm to determine the optimal parameters of PI-PD by the optimization of the weight matrix Q using the improved GOA is as Fig. 3.

Meanwhile, the flowchart of PFC based PI-PD control algorithm is given in Fig. 4.

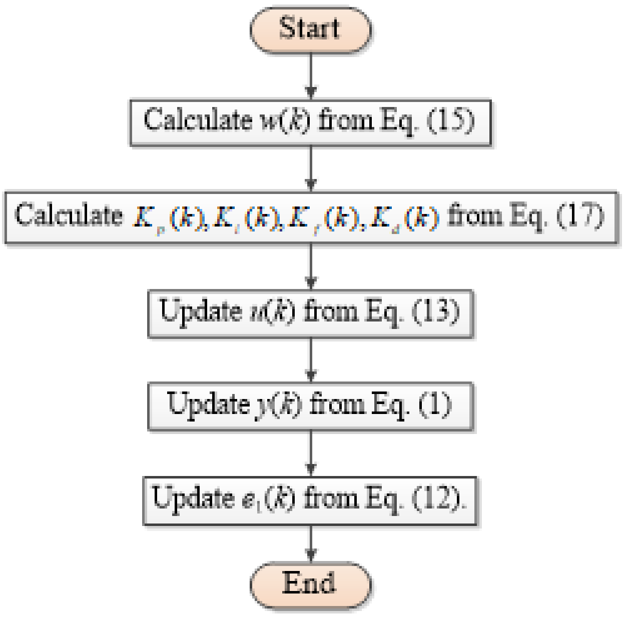
Start
Calculate $w(k)$ from Eq. (15)
Calculate $K_p(k), K_i(k), K_f(k), K_d(k)$ from Eq. (17)
Update $u(k)$ from Eq. (13)
Update $y(k)$ from Eq. (1)
Update $e_1(k)$ from Eq. (12).
End

Fig. 4. Flowchart of PFC based PI-PD control algorithm

Table 1
Tuning parameters of controllers

Method	Tuning parameters
PFC-PIPD	P=6, T_s=0.5, β=0, δ=10^{-4}
ESSPFC-PIPD	P=6, T_s=0.5, β=0, δ=10^{-4}, Q=diag(0, 0, 0, 0, 0, 0, 1)
GA-ESSPFC-PIPD	P=6, T_s=0.5, β=0, δ=10^{-4}, Q= diag (4.3, 0, 0, 0, 0, 0, 1)
GOA-ESSPFC-PIPD	P=6, T_s=0.5, β=0, δ=10^{-4}, Q= diag (1.5, 0, 0, 0, 0, 0, 1)

4. Simulation Results and Discussion

The simulation is carried out for the temperature control process of crude oil in a fluidized catalytic cracking introduced in [21].

Fig. 5 shows the controlled process.

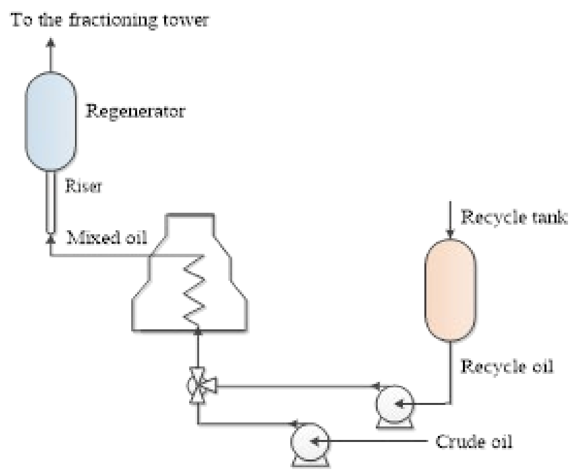

Fig. 5. Process of the temperature control in a fluidized catalytic cracking unit

The process model is expressed by the following FOPDT model [21].

$$G(s)=\frac{1}{4s+1}e^{-2.5s} \tag{26}$$

To illustrate the advantage of the proposed controller (GOA-ESSPFC-PIPD), the controller proposed in [21] (PFC-PIPD), the controller proposed in [22] (ESSPFC-PIPD) and the PI-PD controller designed by the method mentioned in [23] (GA-ESSPFC-PIPD) are introduced for comparison.

The set-point is set as 1, and at time instant k=100, a disturbance with amplitude of -0.05 is added to the output. Table 1 shows the tuning parameters for different controllers.

Fig. 6 shows the simulation results for the nominal model.

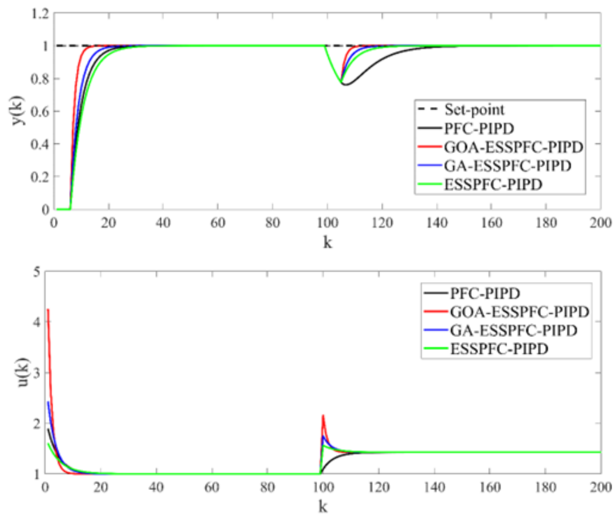

Fig. 6. Closed-loop input and output responses for the nominal model

From Fig. 6, we can obviously see that GOA-ESSPFC-PIPD is better than the others in the set-point tracking and disturbance rejection.

To evaluate the robustness of controllers, the following models with the maximum of 30% uncertainty from the nominal model are introduced [21].

$$\text{Case 1:}\quad G(s)=\frac{0.99}{3.7s+1}e^{-2.85s} \tag{27}$$

$$\text{Case 2:}\quad G(s)=\frac{1.11}{4.46s+1}e^{-2.14s} \tag{28}$$

$$\text{Case 3:}\quad G(s)=\frac{1.21}{4.48s+1}e^{-3.07s} \tag{29}$$

The simulation results for three cases are given in Figs. 7-9.

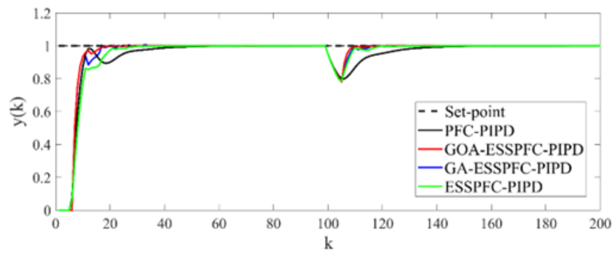

Fig. 7. Closed-loop output responses for case 1

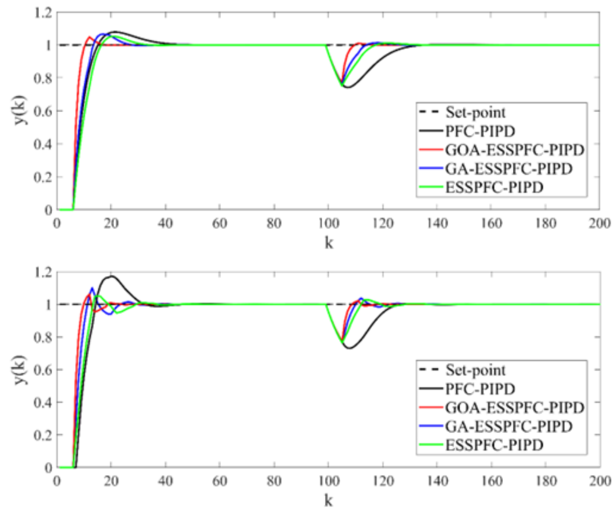

Fig. 9. Closed-loop output responses for case 8

From Figs. 7-9, we can see that GOA-SPFC-PIPD is better than the others in the robustness.

5. Conclusion

A noble PI-PD controller design based on the extended state space PFC and the improved GOA have proposed. The difficult parameter tuning has solved by combining PI-PD and PFC. To overcome the drawbacks of standard GOA, some modifications of GOA have made. And the optimal parameters of PI-PD has been obtained by the optimization of weighting matrix Q using the improved GOA. Simulation results show that the proposed design method is much superior to the others in terms of the set-point tracking, disturbance rejection and robustness.

Acknowledgements

The authors would like to thank the editors and reviewers for their careful comments and excellent suggestions about our paper.

References

[1] O. Garpinger, T. Hägglund, K. J. Åström, "Performance and robustness trade-offs in PID control," *J. Process Control*, 24 (5), pp. 568–577, 2014.

[2] R. D. Zhang, A. Xue, F.R. Gao, "Temperature control of industrial coke furnace using novel state space model predictive control," *IEEE Trans. Ind. Inform.*, 10(4), pp. 2084-2092, 2014.

[3] J. Garrido, F. Vázquez, F. Morilla, "Inverted decoupling internal model control for square stable multivariable time delay systems," *J. Process Control*, 24 (11), pp. 1710–1719, 2014.

[4] M. Miccio, B. Cosenza, "Control of a distillation column by type-2 and type-1 fuzzy logic PID controllers," *J. Process Control*, 24 (5), pp. 475–484, 2014.

[5] M. Shamsuzzoha, M. Lee, "Analytical design of enhanced PID filter controller for integrating and first order unstable processes with time delay," *Chem. Eng. Sci.*, 63 (10), pp. 2717–2731, 2008.

[6] D. Atherton, "PI-PD, an Extension of Proportional–Integral–Derivative Control," *Measurement and Control*, 49 (5), pp. 161–165, 2016.

[7] P. K. Padhy, S. Majhi, "Relay based PI–PD design for stable and unstable FOPDT processes," *Computers and Chemical Engineering*, 30, pp. 790–796, 2006.

[8] N. Tan, "Computation of Stabilizing PI-PD Controllers," *Int. J. Control, Autom. Syst.*, 7 (2), pp. 175–184, 2009.

[9] K. I. Tsai, C. C. Tsai, "Design and experimental evaluation of robust PID and PI-PD temperature controllers for oil-cooling machines," *In Proceedings of the 9th World Congress on Intelligent Control and Automation, Taipei, Taiwan*, pp. 535-540, 2011.

[10] R.D. Zhang, R. Lu, A. Xue, F. R. Gao, "New min-max linear quadratic fault-tolerant tracking control for batch processes," *IEEE Trans. Autom. Control.*, 61 (10), pp. 3045–3051, 2016.

[11] S. Wu, R. D. Zhang, R. Lu, F. R. Gao, "Design of dynamic matrix control based PID for residual oil outlet temperature in a coke furnace," *Chemom. Intell. Lab. Syst.*, 134, pp. 110–117, 2014.

[12] R. D. Zhang, S. Wang, "Support vector machine based predictive functional control design for output temperature of coking furnace," *J. Process Control*, 18, pp. 439–448, 2007.

[13] H. Zhang, W. Lu, J. H. Yang, "Multi-model Switching Predictive Functional Control of Boiler Main Steam Temperature," *Inform. Technol. J.*, 12 (3), pp. 391-396, 2013.

[14] R. D. Zhang, H. Zou, A. Xue. "GA based predictive functional control for batch processes under actuator faults," *Chemom. Intell. Lab. Syst.*, 137, pp. 67-73, 2014.

[15] J. Zhang, "A non-minimal state space formulation based predictive functional control design for inverse-response processes," *Chemom. Intell. Lab. Syst.*, 139, pp. 70–75, 2014.

[16] Y. Wang, Q. Jin, R. Zhan, "Improved fuzzy PID controller design using predictive functional control structure," *ISA Transactions*, 71 (2), pp. 1-10, 2017.

[17] J. A. Rossiter, R. Haber, K. Zabet, "Pole-placement predictive functional control for over-damped systems with real poles," *ISA Transactions*, 61, 229–239, 2016.

[18] R. D. Zhang, S. Wu, F. R. Gao, "Improved PI controller based on predictive functional control for liquid level regulation in a coke fractionation tower," *J. Process Control*, 24 (3), pp. 125–132, 2014.

[19] J. M. Zhang, "Design of a new PID controller using predictive functional control optimization for chamber pressure in a coke furnace," *ISA Transactions*, 67, pp. 321-328, 2016.

[20] R. D. Zhang, A. Xue, R. Lu, P. Li, F. R. Gao, "Real-time implementation of improved state space MPC for air supply in a coke furnace," *IEEE Trans. Ind. Electron.*, 61 (7), pp. 3532–3539, 2014.

[21] H. Zou, H. Li, "Improved PI-PD control design using predictive functional optimization for temperature model of a fluidized catalytic cracking unit," *ISA Transactions*, 67, pp. 336-443, 2016.

[22] H. S. Li, J. M. Zhang, "Improved PID design using new state space predictive functional control optimization based structure," *Chemom. Intell. Lab. Syst.*, 151, pp. 95-102, 2016.

[23] J. L. Tao, Z. H. Yu, Y. Zhu, "PFC based PID design using genetic algorithm for chamber pressure in a coke furnace," *Chemom. Intell. Lab. Syst.*, 137, pp. 155–161, 2014.

[24] K. D. Lu, W. N. Zhou, G. Q. Zeng, "Design of PID controller based on a self- adaptive state-space predictive functional control using extremal optimization method," *Journal of the Franklin Institute*, pp. 1-24, 2018.

[25] A. F. Algamluoli, N. H. Abbas, "Speed controller design for three phase induction motor based on dynamic adjustment grasshopper optimization algorithm," *International Journal of Electrical and Computer Engineering* (*IJECE*), 11 (2), pp. 1143-1157, 2021.

[26] A. Saxena, "A comprehensive study of chaos embedded bridging mechanisms and crossover operators for grasshopper optimization algorithm," *Expert Systems with Applications*, 132, pp. 166–188, 2019.

[27] J. Luo, H. Chen, Q. Zhang, Y. Xu, "An Improved Grasshopper Optimization Algorithm with Application to Financial Stress Prediction," *Applied Mathematical Modelling*, 64, pp. 654-668, 2018.

[28] A. I. Omar, S. Aleem, E. ElZahab, "An improved approach for robust control of dynamic voltage restorer and power quality enhancement using grasshopper optimization algorithm," *ISA Transactions*, 95, pp. 110–129, 2014.

[29] A. Saxena, S. Shekhawat, R. Kumar, "Application and development of enhanced chaotic grasshopper optimization algorithms," *Modelling & Simulation in Engineering*, pp. 1–14, 2018.

[30] X. Yue, H. Zhang, "Grasshopper optimization algorithm with principal component analysis for global optimization," *Journal of Supercomputing*, 7 (7), pp. 5609–5635, 2019.

[31] S. Saremi, S. Mirjalili, A. Lewis, "Grasshopper Optimization Algorithm: Theory and application," *Advances in Engineering Software*, 105, pp. 30-47, 2017.

[32] A. A. Ewees, M. A. Elaziz, E. H. Houssein, "Improved grasshopper optimization algorithm using opposition-based learning," *Expert Syst. Appl.*, 112, pp. 156–172, 2018.

[33] Y. Yan, H. Z. MA, Z. D. Li, "An Improved Grasshopper Optimization Algorithm for Global Optimization," *Chinese Journal of Electronics*, 30 (3), pp. 451-459, 2021.

[34] S. Mirjalili, S. H. Saremi, S. M. Mirjalili, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Systems with Applications*, 47, pp. 106-119, 2016.

[35] M. Kohli, S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *J. Comput. Des. Eng.*, 5 (4), pp. 458–472, 2018.