Self-Tuning of PID Parameters Based on Adaptive Genetic Algorithm

Jimin Zhao\textsuperscript{a}, Miao Xi\textsuperscript{b}
Tianjin University of Science and Technology, Tianjin, China
\textsuperscript{a}zhaojiminemail@tust.edu.cn, \textsuperscript{b}973058690@qq.com

Abstract. Aiming at the problems existing in the PID parameter tuning of traditional genetic algorithms, a method of applying adaptive genetic algorithms to parameter tuning was proposed. It takes system overshoot and dynamic performance indicators as the objective function, optimizes the crossover and selection probability in the genetic algorithm, reduces the probability of the system entering a local optimum, and makes the system converge faster. Comparing the traditional manual tuning PID and the genetic algorithm (GA) PID controller with the adaptive genetic algorithm (AGA) PID controller, it is concluded that the use of adaptive genetic algorithm can improve the performance indicators of the system.

1. Preface
Since the production of PID controllers, it has become the controller with the highest frequency and widest range of use in automated production processes. Combining traditional PID with intelligent control algorithms can produce many improved PID controllers. Genetic algorithm, as a new type of intelligent control algorithm, is simple, efficient, and has advantages in seeking the global optimal solution. However, because the crossover probability and mutation probability are fixed in the traditional genetic algorithm, the early search efficiency is not high, the probability of generating a new individual is small, and the probability of the final solution becoming a local optimum is greatly increased. Therefore, this paper adopts an improved adaptive genetic algorithm to set the parameters of the PID control system and improve the overall performance index of the system.

2. PID control
PID control is a control method with the purpose of minimizing system deviation. It mainly uses three operations: proportional, integral, and derivative of the system deviation, and linearly combines them into control variables. The error is smaller. System deviation is generally expressed as:

\[ e(t) = r(t) - y(t) \]  

PID function:

\[ u(t) = K_P[e(t) + \frac{1}{T_I} \int_0^t e(t) dt + \frac{T_D}{T_I} \frac{de(t)}{dt}] \]  

Or written as a transfer function form:
In the middle, $k_p$ is the control ratio, $T_i$ is the integration time coefficient, $T_D$ is the differential time coefficient.

The traditional PID control principle is as follows:

$$G(s) = \frac{U(s)}{E(s)} = k_p \left(1 + \frac{1}{T_i s} + T_D s \right)$$

(3)

Assuming

$$\Delta u(k) = u(k) - u(k-1)$$

(5)

$$u(k-1) = k_p \left\{ e(k-1) + \frac{T_i}{T} \sum_{i=0}^{k-1} e(i) + \frac{T_d}{T} \left[ e(k-1) - e(k-2) \right] \right\}$$

(4)

It is stipulated that the coefficient $k_p$ before the k-1 error is the proportional operation coefficient of the system; the coefficient $k_i$ ($k_p/T_i$) of the sum of errors at all times is the integral operation coefficient of the system; the coefficient $E$ before the degree of error of the k-1 time is the differential operation coefficient get:

$$\Delta u(k) = k_p \left\{ e(k) - e(k-1) \right\} + k_i e(k) + k_d \left[e(k) - 2e(k-1) + e(k-2) \right]$$

(6)

3. Adaptive genetic algorithm tuning PID parameters

Genetic algorithm (GA) is a search algorithm that simulates the evolutionary laws and genetic forms of the natural world. It is a multi-parameter, multi-combination simultaneous optimization method that simulates the principle of "natural selection, survival of the fittest", and constantly updates the population with certain rules to obtain the optimal solution. The basic process of the genetic algorithm is as follows:

1. Initialize parameters and encode independent variables to form an initial population.
2. Select the appropriate fitness function according to the mathematical model and calculate the individual fitness.
3. The selection operation sorts the population individuals by fitness, and selects a fixed number of individuals with higher fitness to enter the next generation.
(4) Cross-computation, two different individuals exchange certain genes, and cross-generate new excellent individuals.

(5) Mutation calculation, in order to ensure the diversity of individuals, randomly change a certain gene with a small probability to generate a new individual.

(6) Generate new groups, set termination conditions, if conditions are met, stop, find the optimal solution; otherwise, continue to compare calculations, selections, crossovers, mutations, continue to find the best individuals, until they find, stop.

The improved adaptive genetic algorithm is applied to PID parameter tuning, and its structure is as follows:

![Figure 2. Adaptive genetic algorithm tuning PID parameters.](image)

### 3.1. Coding and decoding

There are many forms of genetic algorithm coding. Although real coding eliminates the decoding step but the genetic operation is complicated, the binary coding genetic operation is very convenient. The system needs to be set to the parameters are \([k_p, k_i, k_d]\), Suppose the interval in which the three parameters are located is \([k_{min}, k_{max}]\), and the accuracy of the solution is accurate to \(m\) decimal places, Then the binary code length \(n\) of each parameter shall satisfy:

\[
2^{n-1} \leq (k_{max} - k_{min}) \times 10^m \leq 2^n
\]  

(7)

Suppose \(k_{min} = 0, k_{max} = 20, m = 3\) in the above formula, then each parameter corresponds to a binary code length of 15, each individual contains three parameters, each individual has a code length of 45:

\[
\begin{pmatrix}
k_{p1}, \ldots, k_{p15}
k_{i1}, \ldots, k_{i15}
k_{d1}, \ldots, k_{d15}
\end{pmatrix}
\]  

(8)

Serialize the binary corresponding to each parameter to a decimal number during decoding \(x_p, x_i, x_d\):

\[
(k_{p1}, \ldots, k_{p15})_2 = (\sum_{t=1}^{15} k_{pt} \cdot 2^t)_{10} = x_p
\]  

(9)

\[
(k_{i1}, \ldots, k_{i15})_2 = (\sum_{t=1}^{15} k_{it} \cdot 2^t)_{10} = x_i
\]  

(10)

\[
(k_{d1}, \ldots, k_{d15})_2 = (\sum_{t=1}^{15} k_{dt} \cdot 2^t)_{10} = x_d
\]  

(11)

The actual \(k_p, k_i, k_d\) parameters are:

\[
k_p = k_{min} + x_p \cdot \frac{k_{max} - k_{min}}{2^{n-1}}
\]  

(12)
\[ k_i = k_{min} + \hat{x}_i \cdot \frac{k_{max}-k_{min}}{2^{n-1}} \quad (13) \]
\[ k_d = k_{min} + \hat{x}_d \cdot \frac{k_{max}-k_{min}}{2^{n-1}} \quad (n = 15) \quad (14) \]

### 3.2. Fitness function

In order to meet the requirements of PID control, the mathematical model of the fitness function is specified as follows:

1. The fitness function contains the absolute value of the system deviation \( e(t) \), and in order to achieve different emphasis control schemes, a certain proportion operation is performed on the deviation with a coefficient of.

2. The fitness function contains the control quantity input \( u(t) \) at that moment, and the square of the input is used as a factor of the fitness function, and a proportional operation is also performed on the item with a coefficient of.

3. The fitness function contains the rise time \( t(u) \) of the system, and the term is proportionally calculated with a coefficient of.

4. Once the system has an overshoot, use the overshoot as an important component of the fitness function, and use a relatively large scale factor as a penalty function to avoid overshoot.

From the above points, the objective function of the system can be obtained:

If \( ey(t) \geq 0 \)
\[ J = \int_0^\infty (w_1 |e(t)| + w_2 u^2(t))d(t) + w_3 \cdot t(u) \quad (15) \]

If \( ey(t) < 0 \)
\[ J = \int_0^\infty (w_1 |e(t)| + w_2 u^2(t) + w_4 |ey(t)|)d(t) + w_3 \cdot t(u) \quad (16) \]

Where \( ey(t) = y(t) - y(t-1) \), \( y(t) \) is the output, and the fitness function is set to the inverse of the objective function, so the fitness function is \( F = \frac{1}{J} \).

### 3.3. Cross and Mutation

The reason why the genetic algorithm can perform global optimization is to rely on crossover and mutation to constantly generate new individuals, and the newly generated offspring and parent individuals are simultaneously selected for survival.

The cross probability is positively related to the rate of new individual generation, but when the cross probability is too large, too many individuals in each generation will be changed, which is not conducive to searching. If the mutation probability is too small, it will not be easy to generate completely new individuals. It will make genetic algorithms become blind and irregular. In view of the characteristics of crossover and genetic probability (AND), Srinivas et al. Proposed an adaptive genetic algorithm (Adaptive GA, AGA), in which the two probabilities can be changed according to the fitness.

In the adaptive genetic algorithm, the values of \( P_c \) and \( P_m \) belong to the following formula:

\[ P_c = \begin{cases} \frac{k_3(f_{max}-f)}{f_{max}-f_{avg}} & f \geq f_{avg} \\ k_2 & f < f_{avg} \end{cases} \quad (17) \]
\[ P_m = \begin{cases} \frac{k_3(f_{max}-f)}{f_{max}-f_{avg}} & f \geq f_{avg} \\ k_4 & f < f_{avg} \end{cases} \quad (18) \]
In the formula, the maximum fitness of each generation group is $f_{\text{max}}$, and the average fitness of each generation group is $f_{\text{avg}}$. In the cross operation, the fitness of the individual with the larger fitness value in the two individuals is $f$. In the mutation operation, the fitness value of the individual involved in the mutation is $f$, where $k_1, k_2, k_3, k_4$ is the constant of the slave and $(0, 1)$.

This improvement solves to some extent the effects of constant crossover and mutation probability, but there are still some problems: the crossover and mutation probability of the most adaptive individual in each generation of the population is basically 0, which leads to the algorithm search Individuals with large fitness in the early stage cannot be changed, reducing the convergence speed, and also increasing the probability of eventually obtaining a local optimal solution. Therefore, based on the above formula, the cross-probability and mutation probability of the individual with the highest fitness are increased, thereby Disadvantages of optimizing traditional genetic algorithms.

The improved probability is as follows:

\[
P_c = \begin{cases} 
P_{c1} - \frac{(P_{c1} - P_{c2})(f - f_{\text{avg}})}{f_{\text{max}} - f_{\text{avg}}} & f \geq f_{\text{avg}} \\ P_{c1} & f < f_{\text{avg}} \end{cases} \quad (19)
\]

\[
P_m = \begin{cases} 
P_{m1} - \frac{(P_{m1} - P_{m2})(f_{\text{max}} - f)}{f_{\text{max}} - f_{\text{avg}}} & f \geq f_{\text{avg}} \\ P_{m1} & f < f_{\text{avg}} \end{cases} \quad (20)
\]

Where $P_{c1} = 0.9, P_{c2} = 0.6, P_{m1} = 0.1, P_{m2} = 0.001$, $P_{c2}$ and $P_{m2}$ are the individual crossover and mutation probabilities with the highest fitness.

The system flow is as follows:

![Adaptive genetic algorithm](image)

**Figure 3.** Adaptive genetic algorithm.

4. Simulation and results

In order to verify the feasibility of adaptive genetic algorithm tuning PID parameters, Matlab is used for simulation joint tuning. Manually tuning PID parameters, traditional genetic algorithm tuning PID parameters and adaptive genetic algorithm tuning PID parameters are the three steps of the parameters. The response is compared, the sampling time is set to 1ms, and the transfer function is as follows:

\[
G(s) = \frac{133}{s^2 + 25s}
\]

The parameters are set as follows:

1. Initial population size $n=50$, number of iterations $T=50$.
2. The crossover probability and the mutation probability are adaptively adjusted according to the expressions given by (19) and (20).
(3) The selection operation uses the roulette model, and the probability that a certain body \( x \) is inherited to the next generation is:

\[
P_x = \frac{F_x}{F_{\Sigma}}
\]  

(21)

Where \( F_{cx} \) is the fitness value of the \( x \)th individual of a generation, and \( F_{\Sigma} \) is the sum of the fitness of all individuals in a generation.

Comparing the results of adaptive genetic algorithm and traditional genetic algorithm from the two aspects of convergence algebra and parameter results, the results are as follows:

![Figure 4. Algorithm's convergence algebra.](image)

![Figure 5. PID parameters obtained by each algorithm.](image)

The convergence speed of the adaptive genetic algorithm (AGA) in Figure 4 is significantly faster than that of the traditional genetic algorithm. The traditional genetic algorithm converged in the 23rd generation, and the adaptive genetic algorithm converged in the 17th generation. From Figure 5, the final parameters of the two algorithms can be seen. The results are relatively close, and it can be concluded that AGA is more efficient than GA, thus proving the feasibility of adaptive genetic algorithm for PID parameter tuning.
Table 1. Step response characteristics obtained by each algorithm.

| Algorithm       | $Y_{max}$ | $Y_{o}$ | $t_{max}$ | $\sigma/\%$ |
|-----------------|-----------|---------|-----------|-------------|
| AGA             | 1.0157    | 1.0153  | 0.9947    | 0.00042     |
| GA              | 1.0245    | 1.0240  | 0.9951    | 0.00050     |
| Traditional PID | 1.0319    | 1.0314  | 0.9950    | 0.00053     |

Figure 6. Step response curves of three algorithms.

It can be obtained from Figure 6 that the step response overshoot obtained by the traditional manual tuning PID algorithm is large; the step response obtained based on the PID parameter tuning of GA is smaller than the manual tuning, but its peak time is larger; And the step response based on AGA PID parameter tuning is better than the other two algorithms in terms of adjustment time, peak time, and overshoot.

5. Conclusion
PID controller has been widely used in the field of industrial control due to its simple principle, easy implementation and wide application. It has been the research direction of experts for how to determine PID parameters to make the system stronger. On the premise of referring to the existing tuning technology, this paper proposes parameter tuning of PID controller based on adaptive genetic algorithm. After improving the traditional genetic algorithm, the PID parameters were optimized, and after the experiment, it was concluded that the algorithm improved the dynamic performance and steady-state accuracy of the system.

References
[1] Gupta A, Goindi S, Singh G, et al. Optimal design of PID controllers for time delay systems using genetic algorithm and simulated annealing [C] // 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). IEEE, 2017.
[2] Devanshu A. Genetic Algorithm Tuned PID Controller for Process Control [C] // International Conference on Inventive Systems and Control. IEEE, 2017.
[3] Neath M J, Swain A K, Madawala U K, et al. An Optimal PID Controller for a Bidirectional Inductive Power Transfer System Using Multiobjective Genetic Algorithm [J]. IEEE Transactions on Power Electronics, 2013, 29 (3): 1523-1531.
[4] Pan I, Das S, Gupta A. Tuning of an optimal fuzzy PID controller with stochastic algorithms for networked control systems with random time delay [J]. ISA Transactions, 2011, 50 (1): 28-36.
[5] Zhang J, Zhuang J, Du H, et al. Self-organizing genetic algorithm based tuning of PID controllers [J]. Information Sciences, 2009, 179 (7): 1007-1018.
[6] Gaing Z L. A particle swarm optimization approach for optimum design of PID controller in AVR system [J]. IEEE Transactions on Energy Conversion, 2004, 19 (2): 384-391.