ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 1 - 2026

# Optimization design of PID control system for single-jointed robotic arm based on particle swarm algorithm

*Yang Zhang*[1,*]

[1] Shanxi Vocational Technical College, Taiyuan, Shanxi, 030006, China

**SUMMARY:** *One solution to enhance the accuracy in the trajectory-tracking of a single-joint robot arm under PID control is to implement a fuzzy PID strategy based on fuzzy logic. Such a controller may describe the nonlinear behavior of the plant without relying on an accurate mathematical representation of the controlled object, but it also maintains the standard PID structure. To further meet the optimization demands of the trajectory tracking, the Dynamic Population Dead-Zone-Initiated Adaptive Particle Swarm Optimization algorithm (DP-DZIA-PSO) is applied to adjust the fuzzy PID parameters. On importing a six-degree-of-freedom robotic-arm dynamic model developed in SolidWorks into ADAMS, simulation and motion analysis are performed and the target angular velocity is used to compute the torque required by every joint and thus formulating a foundation of selecting the motor. Then, PSO, DZIA-PSO, and DP-DZIA-PSO are used to minimize the robotic-arm PID control system, and the comparative results of the three methods are obtained. The findings indicate that the enhanced particle-swarm method proposed herein provides a better Pareto front over the other two approaches. Because the controllers associated with various optimization solutions possess different control properties, appropriate solutions may be selected based on particular practical needs.*

***KEYWORDS:*** *DP-DZIA-PSO; fuzzy PID controller; six-degree-of-freedom robotic arm; trajectory tracking control*

## 1 Introduction

Due to the accelerated pace of industrial automation, digital technology has been rapidly integrated into the whole manufacturing process, which offers the technical foundation base of the industry and helps to hasten the manufacturing transformation and development, thus being a significant driver of economic growth [1, 2]. As one of the key parts of automated assembly line, robotic arms have been used in large volumes since they can increase the efficiency of production processes, reduce operational expenses, and ensure the quality of the products, which resulted in high demand on them in the sphere of manufacturing, logistics and warehousing [3]. The simple configuration and versatility of application scenarios of the single-jointed robotic arm are among other types, with a linkage and joint, which is a significant reference to the design of multi-jointed robotic arms[4]. Proportional-integral-derivative (PID) control has been popular in industrial automation due to its simple form, good stability, and high reliability [5].After decades of development, PID control has become popular in engineering systems. Nevertheless, in the engineering field, less than a half of the controllers have reached the user-satisfying control performance, and when dealing with the

*Zhangyang19836@163.com
https://doi.org/10.65102/is2026376
Published: 30 April, 2026

complex, nonlinear robotic arm system and complex environmental disturbances, PID controllers fail [6-8]. Moreover, parameter tuning and dynamic response are also the most challenging aspects of conventional PID controllers [9].

The famous Ziegler-Nichols (ZN) method has been used in tuning PID parameters since the controlled plant can be accessed; nonetheless, in engineering applications, it sometimes yields significant deviation, and the optimal compromise between fast response and stability of the system is not easy to obtain [10]. In recent years, intelligent optimization methods have found their way into an extensive variety of optimization problems. The use of genetic algorithms (GA), ant colony optimization, firefly optimization, and particle swarm optimization (PSO) has become increasingly popular as practical tools of PID parameter tuning on controlling robotic arms. According to [11], there was a GA that used the roulette-wheel selection, one-point crossover, and Gaussian mutation applied to a single-joint robotic arm that has minimized the main performance indicators of the PID controller and consequently enhanced the transient and steady-state behavior. A PID controller was constructed in reference to[12] based on an advanced GA in which rise time and settling time were minimized to 1.23 seconds and 1.81 seconds respectively and the horizontal line accuracy and diagonal accuracy were increased by 23.98 percent and 23.64 percent respectively. Reference [13] also optimized the PID sliding-surface parameters of a planetary rover via an enhanced GA, which resulted in more precise control, smoother movement of the rover, and greater flexibility and real-time operation. In [14], GA was used to adjust the PID parameters and then fuzzy control was used to optimize the secondary controller and achieve a maximum overshoot of 5.01 and improve the trajectory tracking and stability. Nevertheless, since the mechanism of GA-PID is based on the nature of the natural-selection mechanism, it could also experience a long convergence time, low efficiency of computation and sensitivity to some parameters of the controllers.

Reference [15] integrated the whale optimization algorithm with a multi-objective ocean predator algorithm for PID tuning, generating multiple candidate solutions and achieving more balanced overall behavior. That method shortens the frequency stabilization time and preserves high stability under load uncertainty. In [16], the invasive weed optimization algorithm was used to determine PID parameters, and it was further combined with an artificial neural network for parameter selection, making it possible to obtain zero maximum overshoot of the control system while reducing the rise time of some joints to 0.1 s. Reference [17] introduced a dynamically weighted artificial bee colony algorithm into robotic-arm PID design to provide a better control strategy, improve end-effector trajectory behavior, and strengthen resistance to external disturbances. In [18], the colony size of the artificial bee colony algorithm was modified to obtain optimal parameter values for a robotic-arm PID controller initialized by ZN tuning, and the optimized results showed improvements in several time-domain indicators along with better dynamic characteristics. Reference [19] combined a dynamic-weight artificial bee colony optimization algorithm with fuzzy theory to establish a real-time machine-learning-based control strategy for articulated robots, enabling online parameter adjustment with improved convergence performance. In [20], fuzzy technology was incorporated into a PID controller built on a social spider optimization algorithm, allowing the robotic arm to achieve zero steady-state error and more accurate positioning, which in turn improved coordination among all joints. Nevertheless, although these approaches reduce controller error indices and enhance several time-domain performance measures, maintaining a satisfactory balance between convergence speed and control accuracy remains difficult, and artificial bee colony optimization may still be trapped in local optima.

A robotic-arm PID controller was created using the SISTOOL PID auto-tuning program,

and ant colony optimization was introduced to find the best gain combination that enhanced the performance of the controller and its operating efficiency as described in Ref. [21]. Ant colony optimization was used as a basis of a PID tuning algorithm in reference [22], and it was found out that the mean absolute error and normalized root mean square error were much lower than fuzzy-PID and ZN-PID and that the tracking accuracy, convergence rate and stability were also higher. In[23], an optimal nonlinear PID control algorithm of micro-robot was developed using ant colony optimization which effectively eliminated the low performance of linear PID controllers in low-friction conditions or in the presence of system disturbances. Reference [24] used the firefly algorithm to optimize the PID parameters of robots with various degrees of freedom and integral time square error analysis revealed that the firefly approach was superior to the ZieglerNichols method in terms of suppressing overshoot and also reducing the rise time and settling time. In [25], both the traditional and upgraded versions of the firefly algorithms were applied to improve the PID control of the flexible beam structures. The firefly-based PID controller successfully reduced the uncontrolled vibration of the flexible beam by more than 96.2 percent compared to the heuristic method, and the improved firefly algorithm increased the attenuation ratio up to 98.8 percent. Nevertheless, ant colony optimization has its own disadvantages like slow convergence and dependence on parameters, and therefore the quality of search and accuracy may be reduced progressively in the course of the subsequent optimization.

In literature [26] a PID control strategy was introduced based on the combination of firefly optimization technique with a neural network. This approach minimized the integral time multiplied squared error and enhanced the precision of tracking, recovery and stability of work of robots without multiple repeated retraining. In literature [27], two neural networks were created to recognize the target system and propose PID parameters depending on the recognition outcomes which resulted in reduction of the number of parameter changes by 92.9 percent and increased efficiency in controller operation. Literature [28] used a radial basis function neural network algorithm to optimize a PID controller to enhance response time, reject disturbances and track behaviour of a bilateral upper-limb rehabilitation robotic arm of hemiplegics. Even though neural-network-based PID optimization has been proven to be more accurate and stable in terms of tracking and less frequent in terms of parameter adjustment than its counterparts, the overall control effect remains limited by a long training period, slow convergence, and sensitivity to initial weights.

Particle swarm optimization (PSO) is an evolutionary technique driven by swarm intelligence and has been widely applied in function optimization, neural-network training, pattern recognition, and fuzzy control. Owing to its simple structure, convenient implementation, high computational efficiency, fast convergence, and relatively low tendency to become trapped in local optima, PSO has shown favorable performance in many applications. Reference [29] presented an optimized PID controller based on PSO. Compared with a conventional PID controller, the PSO-based PID system reduced response time, enhanced resistance to external disturbances, improved control accuracy, and yielded more stable dynamic behavior. In [30], adaptive PSO was combined with an advanced Drosophila optimization algorithm to optimize the parameters of a nonlinear neural fractional-order PID controller for non-complete differential-drive mobile robots, reducing the mean square error to 0.00059 and decreasing energy consumption during circular motion. Reference [31] compared PID controllers tuned by Ziegler – Nichols, empirical tuning, PSO, and beetle antennae search over different test paths, and the results showed that PSO and beetle antennae search produced comparable errors, although PSO performed slightly better.

Reference [32] employed ZN, GA, ant colony optimization, and PSO to tune the PID

controller of a single-link underwater dual robotic arm. Among these methods, PSO achieved the best trajectory-tracking performance, and the integral of time-weighted absolute error of the dual arm was reduced by 96.44% and 98.6%. In [33], PSO was incorporated into a backpropagation-neural-network-based PID controller to strengthen the global search capability of the network, reduce controller overshoot to 1.37%, and improve response speed, control precision, and robustness. Reference [34] used intelligent PSO to optimize the gain of a PID controller and further refine the parameters of a sliding-mode controller, thereby constructing a proper and stable nonlinear controller for a robot manipulator and reducing vibration during motion. In [35], PSO was adopted to calculate the optimal tuning parameters of a PID controller for a six-degree-of-freedom robotic arm, which reduced uncertainty in the control system and provided higher efficiency, stronger stability, lower overshoot, and less vibration. Nevertheless, most existing studies remain at the level of theoretical simulation, practical implementation is still rather limited, and the robustness of PID control systems requires further continuous improvement.

In order to solve the problems mentioned above, this paper explores fuzzy PID control to track trajectories of a six degrees of freedom robotic arm with an articulated structure. Initially, the kinematics of the articulated system are discussed followed by formulation of the kinematics equations through D-H parameters and calculation of the forward and inverse kinematic solutions. The next step is to add fuzzy control rules to the traditional PID system and design a suitable fuzzy PID controller. An approach chosen to simplify the process of controller-parameter optimization is to use the DP-DIZA-PSO algorithm. Based on the classic PSO architecture, the approach is refined in four directions: inertia weight, the learning coefficient, number of individuals and initialization of particles. A dynamic simulation model of the six degree of freedom robotic arm is subsequently developed using ADAMS and the torque that should be supplied by each joint is calculated. Lastly, the efficacy and efficiency of DP-DIZA-PSO in optimizing fuzzy PIDs to robotic arms could be validated by comparing the behavior of various algorithms.

# 2 Method

## 2.1 Structural design of the robotic arm

D-H parameter approach has become one of the most popular methods to characterize a kinematic chain. One can consider a robotic arm as a spatial open-chain mechanism with multiple links connected via joints. Using a separate coordinate frame to every link allows expressing the geometric relationships between adjacent joints and links with greater precision through the D-H parameters. At the same time, the transformation matrix can be reduced to a simplified form to calculate the coordinate frame of another link, simplifying the development of both the forward and inverse kinematic solutions. The linkage model of the robotic arm based on D-H is shown in Figure 1.

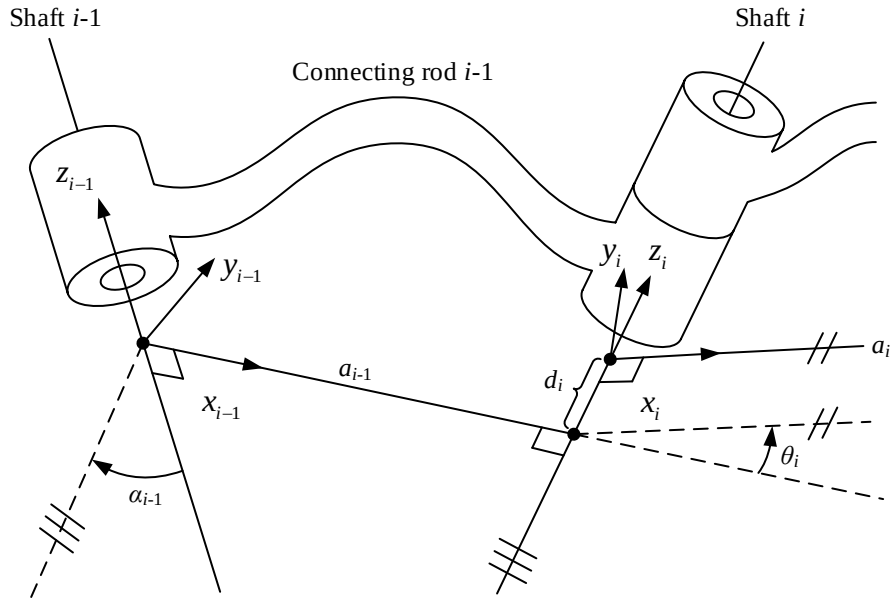



*Figure 1: The mechanical arm linkage D-H parameter model*

The coordinate frame of the $i$ th joint is established by choosing the axial direction of the $i$ th joint as $z_i$ -axis, the common normal between $z_i$ and $z_{i+1}$ that points to the successive joint as $x_i$ -axis and thereafter the $y_i$ -axis by using the right-hand rule. In this convention, four parameters of the connecting link may be specified:

(1)the distance measured between $z_{i-1}$ and $x_{i-1}$ and $z_i$, or the link length;

(2)the rotation between $z_{i-1}$ and $z_i$, i.e. $x_{i-1}$ the twist angle $\alpha_{i-1}$ ;

(3) the measurement of distance from $z_i$ to $x_i$ along $x_{i-1}$, i.e. joint offset $d_i$;

(4) $x_{i-1}$ to $x_i$ rotation around $z_i$, which is the joint angle $\theta_i$ .

In the present work, the six-degrees-of-freedom robotic arm is designed on the SolidWorks platform as per the requirement of D-H parameter modeling. There are six revolute joints in the manipulator, each of which is attached to the appropriate link. The first three joints are responsible with the control of the location to which the arm arrives, while the other three are used to control its orientation. Because the axes of the final three joints meet at the same point, the origins of the three coordinate frames are all at the same point. Therefore, some of the D-H parameters are zero and this simplifies the process of deriving forward and inverse kinematics solutions.

To simplify the transformation matrix, the robotic-arm D-H parameter model is arranged as shown in Fig. 2, when $z_{i-1}$ is parallel to $z_i$, the $\{i-1\}$ origin is taken at the place that makes $d_i = 0$, and when $z_{i-1}$ intersects $z_i$ at $a_{i-1} = 0$, the $\{i-1\}$ origin is taken at the intersection point of the $z_{i-1}$. Table 1 shows the parameter list of the robotic arm linkage $D-H$ model.

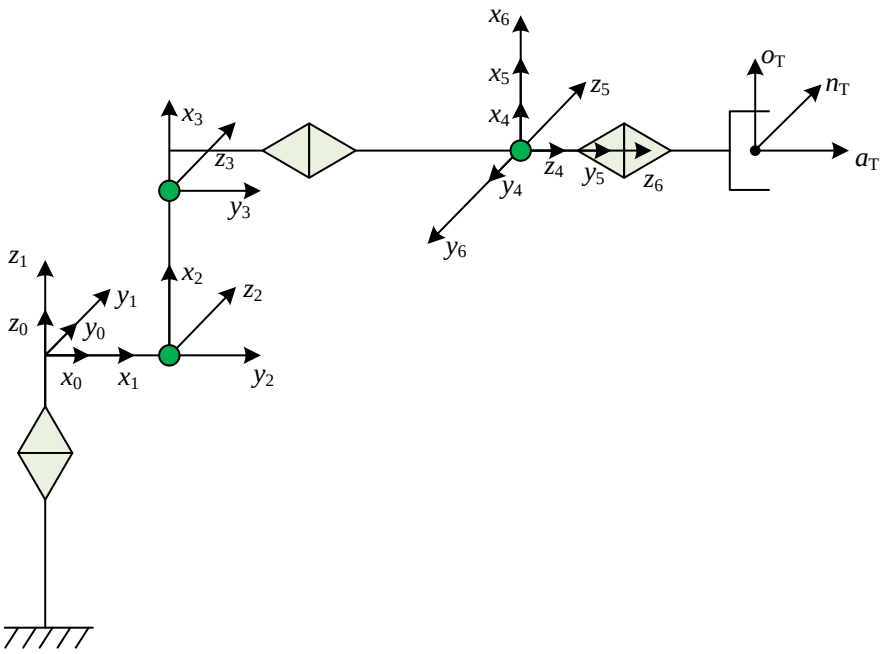



*Figure 2: Mechanical arm D-H parameter model*

*Table 1: Mechanical arm linkage D-H model parameter*

| I | $\theta_i$ (º) | $d_i$ (mm) | $\alpha_{i-1}$ (º) | $a_{i-1}$ (mm) | Range of joint transformation |
|---|---|---|---|---|---|
| 1 | $\theta_1$ | 0 | 0 | 0 | -180º~180º |
| 2 | $\theta_2$ | 0 | -100 | 60 | -160º~45º |
| 3 | $\theta_3$ | 0 | 0 | 300 | -80º~80º |
| 4 | $\theta_4$ | 350 | -100 | 100 | -180º~180º |
| 5 | $\theta_5$ | 0 | 100 | 0 | -90º~90º |
| 6 | $\theta_6$ | 0 | -100 | 0 | -270º~270º |
| T | 0 | 85 | 0 | 0 | |

## 2.2 Kinematic analysis of the robotic arm

### 2.2.1 Positive kinematic solution of the robotic arm

The forward kinematics of a robotic manipulator is concerned with finding out the position and orientation of the end effector after the arm configuration is determined and all the joint rotation variables are determined. Depending on the modeling data given by the $D-H$ parameters, the link coordinate frame $\{i\}$ transformation with respect to frame $\{i-1\}$ can be calculated by four consecutive transformations, which are the link transformations ${}_{i}^{i-1}T$ .

(1) Rotate $\alpha_{i-1}$ about axis $x_{i-1}$, so that $z_{i-1}$ becomes parallel to $z_i$ .

(2) Translate $a_{i-1}$ along $x_{i-1}$, so that $z_{i-1}$ lies on the same line as $z_i$ .

(3) Rotate $\theta_i$ around axis $z_i$ , allowing $x_{i-1}$ to turn parallel to $x_i$ .

(4) Translate $d_i$ along axis $z_i$ , so that the origin of coordinate frame $\{i\}$ coincides with the origin of $\{i-1\}$ .

Express this in terms of the transformation matrix:

$$
\begin{aligned}
{}^{i-1}_{i}T &= Rot(x,\alpha_{i-1})Trans(x,a_{i-1})Rot(z,\theta_i)Trans(z,d_i) \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_{i-1} & -s\alpha_{i-1} & 0 \\ 0 & s\alpha_{i-1} & c\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

where $s$ is sin and $c$ is cos.

The transformation matrix between the coordinate systems is:

$$
\begin{aligned}
{}^{0}_{1}T &= \begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};{}^{1}_{2}T = \begin{bmatrix} c2 & -s2 & 0 & 58 \\ 0 & 0 & 1 & -5 \\ -s2 & -c2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{2}_{3}T &= \begin{bmatrix} c3 & -s3 & 0 & 280 \\ s3 & c3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};{}^{3}_{4}T = \begin{bmatrix} c4 & -s4 & 0 & 80 \\ 0 & 0 & 1 & 346 \\ -s3 & -c4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{4}_{5}T &= \begin{bmatrix} c5 & -s5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s5 & c5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};{}^{5}_{6}T = \begin{bmatrix} c6 & -s6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s6 & -c6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^{6}_{T}T &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 78 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2}
$$

$$
{}^{0}_{T}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T\,{}^{6}_{T}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}
$$

As a result, the manipulator endpoint coordinates may be calculated based on the rotation parameters of every joint, thus finishing a one-way mapping between joint space and Cartesian space.

### 2.2.2 Inverse solution of the kinematics of the robotic arm

Inverse kinematics is the inverse problem of forward kinematics. The aim is to find out the joint variables in joint space based on a given end-effector position and orientation in Cartesian space. It is often more complex than forward kinematics. In forward kinematics, there is a direct mapping between joint space and Cartesian space but in inverse kinematics there is the reverse transformation between Cartesian space and joint space. Moreover, the amount of possible inverse solutions can also differ and increase exponentially with the number of manipulator joints. A six-degree-of-freedom robotic arm might contain up to 16 inverse solutions.

At present, two main categories of inverse-kinematic methods are commonly adopted: analytical methods and numerical methods. The numerical method is to find the point with the smallest change from the current position by iterative method, which adopts a step-by-step approximation solution method, so the numerical method will only get a group of inverse solutions. Commonly used inverse solvers are KDL and TRAC-IK based on numerical methods, both of which are based on the Jacobi matrix using Newton-Raphson iteration method to find the kinematic inverse solution, which is characterized by fast convergence and self-correction.

Let $X$ denote the generalized position vector of the end position of the robotic arm and $\theta$ be the joint coordinate vector of the robotic arm to establish the kinematic differential equation:

$$\dot{X} = \begin{bmatrix} v \\ w \end{bmatrix} = \lim_{\Delta t \to 0} \frac{1}{\Delta t} \begin{bmatrix} d \\ \delta \end{bmatrix} \Rightarrow D = \begin{bmatrix} d \\ \delta \end{bmatrix} = \lim_{\Delta t \to 0} \dot{X} \Delta t \tag{4}$$

As per the definition of the Jacobian matrix, it depicts how the robotic arm Cartesian-space velocity maps to the joint space velocity or in other words, how the joint-space motion rate converts to Cartesian-space motion rate. It could be represented in the form of:

$$\dot{X} = J\dot{\theta} \tag{5}$$

It can be converted to:

$$D = \lim_{\Delta t \to 0} \dot{X} \Delta t = \lim_{\Delta t \to 0} J\dot{\theta} \Delta t = Jd\theta \tag{6}$$

Since $J$ may be a square matrix of non-full rank, $J$ may not have an inverse matrix, so the pseudo-inverse $J^+$ of $J$ is used:

$$d\theta = J^+ D \tag{7}$$

By performing a Householder-based singular value decomposition of the Jacobi matrix $J$:

$$J = U\Sigma^+ V^* \Rightarrow J^+ = V\Sigma^+ U^* \tag{8}$$

Get:

$$d\theta = V\Sigma^{+}U^{*}D \tag{9}$$

If the robotic arm moves from a previous position of $T_{pre}$ to a current position of $T_{cur}$, then there is:

$$T_{cur} = T_{pre}(I+\Delta) \Rightarrow \Delta = T_{pre}^{-1}T_{cur} - I \tag{10}$$

Since the robotic arm is moved from a previous positional attitude of $T_{pre}$ to a current positional attitude of $T_{cur}$ by translation and rotation, it is obtained:

$$\begin{aligned}
\Delta &= Trans(dx, dy, dz)\, Rot(k, d\theta) - I \\
&= \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta x & \delta y & 0 \\ \delta z & 1 & \delta z & 0 \\ -\delta y & \delta x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & -\delta x & \delta y & dx \\ \delta z & 0 & \delta z & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
\therefore T_{pre}^{-1}T_{cur} - I &= \begin{bmatrix} 0 & -\delta x & \delta y & dx \\ \delta z & 0 & \delta z & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{11}$$

By equalizing the values of the corresponding elements we can get $D = [dx\ dy\ dz\ \delta x\ \delta y\ \delta z]^{-1}$, and then from Eq. we can compute $d\theta$, and finally use the iterative method to set up the iteration by jumping out of the loop to end the iteration when $d\theta$ is less than a certain accuracy, thus obtaining the kinematics of the six-degree-of-freedom robotic arm Inverse solution.

Although both TRAC-IK and KDL solvers are based on numerical methods, they differ in their iterative algorithms. The KDL solver is a simple extension of Newton's convergence algorithm, which may fall into a local minimum. And TRAC-IK is using SQP nonlinear optimization method, which can better optimize the iteration, compared with KDL can quickly get the inverse solution, but also can solve the problem of KDL local minimum, can solve the situation that KDL can not get the inverse solution, have higher reliability and solving efficiency. In summary, so in this paper, the TRAC-IK solver is selected.

## 2.3 Design of Single Joint Controller

### 2.3.1 PID controller design

The PID algorithm has become popular in the field of industrial robots and numerous other control systems due to its ease of configuration, easy implementation, high reliability and good robustness. The core of PID control is that the error between the measured output $y$ and

the desired value $r$ is acted upon via proportional, integral, and derivative terms in a linear combination to produce the control input $u$, thus controlling the controlled plant. Figure 3 illustrates the schematic of the PID controller.

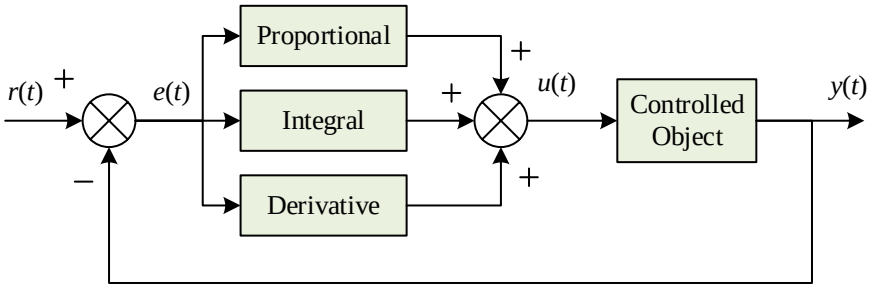



*Figure 3: schematic diagram of PID controller*

Among them:

$$e(t) = r(t) - y(t) \tag{12}$$

Control Law for:

$$u(t) = k_P \left( e(t) + \frac{1}{T_I} \int e(t)dt + T_D \frac{de(t)}{dt} \right) \tag{13}$$

The transfer function is of the form:

$$G(s) = \frac{U(s)}{E(s)} = k_P \left( 1 + \frac{1}{T_I s} + T_D s \right) \tag{14}$$

In Eqs. (13) and (14), $k_P$ denotes the proportional gain, $T_I$ represents the integral time constant, and $T_D$ stands for the derivative time constant. It should be noted that $k_I = k_P / T_I$, $k_D = k_P / T_D$.

The three PID correction term functions can be summarized like this:

Proportional term: The error is the difference between reference input and measured value and is transformed directly into a proportional control signal. To put it differently, as soon as a deviation occurs, the controller acts instantly to remove the deviation. A bigger proportional gain $k_P$ tends to reduce the response time of the system, but it also makes the system more susceptible to overshooting.

Integral term: It plays the major role in removing the steady-state error of the system. The bigger value of integral time constant $T_I$ means less strong integral action and smaller $T_I$ indicates more powerful integral effect.

The differential term: The term is primarily aimed at the dynamic properties of the system. It can show the direction of the deviation by reflecting how fast the system deviation changes and provide a useful anticipatory behavior to the control process, thereby reducing the regulation time as well as assisting in damping the oscillation.

In practical tuning, the parameters of the PID controller are generally chosen via experience. The integral coefficient $k_I$ is initially set to 0, after which the proportional

coefficient $k_P$ and differential coefficient $k_D$ are tuned, frequently by orders of magnitude. Oscillation can be reduced with an appropriate increase in $k_D$. With a steady-state deviation present, the integral parameter $k_I$ needs further adjustment until the system error achieves the required value.

### 2.3.2 Fuzzy PID controller design

(1) Fuzzy PID controller structure

The adaptive fuzzy PID controller takes the error $e$ and the error change $ec$ as inputs, which can satisfy the requirement of self-tuning of PID parameters by $e$ and $ec$ at different moments. Using fuzzy control rules to modify the PID parameters online, it constitutes an adaptive fuzzy PID controller, the structure of which is shown in Figure 4.

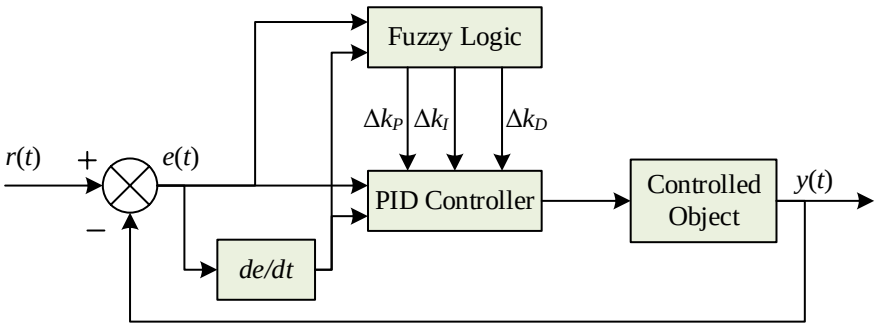



*Figure 4: Fuzzy PID controller structure diagram*

(2) Determination of affiliation function

A new FIS file of type Mamdani is created with the error $e$ and the rate of change of the error $ec$ as inputs, and the output quantities $\Delta k_P$, $\Delta k_I$, and $\Delta k_D$ through the fuzzy rule editor.

Take the fuzzy subsets of the inputs $e$, $ec$ and output quantities $\Delta k_P$, $\Delta k_I$, and $\Delta k_D$ as $\{NB, NM, NS, Z, PS, PM, PB\}$, which represent respectively represent negative large, negative medium, negative small, zero, positive small, positive medium, positive large. The argument domain is $[-3,3]$ and the quantization level is $\{-3,-2,-1,0,1,2,3\}$. The input quantities $e$ and $\Delta k_P$ affiliation functions are chosen to be Gaussian, and the output quantities $\Delta k_P$, $\Delta k_I$, and $\Delta k_D$ are chosen to be triangular.

(3) Setting of fuzzy rules

According to the effects of $k_P$, $k_I$, $k_D$ on the output characteristics, it can be generalized that the fuzzy rules can be set at different $e$ and $ec$, as well as at different $\Delta k_P$, $\Delta k_I$, $\Delta k_D$, so that the fuzzy rules can be obtained.

*Table 2: Fuzzy rules*

| *ec* \ *e* | NB | NM | NS | Z | PS | PM | PB |
|---|---|---|---|---|---|---|---|
| NB | PB/NB/PS | PB/NB/PS | PM/NB/Z | PM/NM/Z | PS/NM/Z | PS/Z/PB | Z/Z/PB |
| NM | PB/NB/NS | PB/NB/NS | PM/NM/NS | PM/NM/NS | Z/Z/Z | Z/Z/NS | Z/Z/PM |
| NS | PM/NM/NB | PM/NM/NB | PM/NS/NM | PS/NS/NS | Z/Z/Z | NS/PS/PS | NM/PS/NM |
| Z | PS/NM/NB | PS/NM/NM | PS/NS/NM | NS/Z/NS | NS/PS/Z | NM/PS/PS | NM/PM/PM |
| PS | PS/NS/NB | PS/NS/NM | Z/Z/NS | NS/PS/NS | NS/PS/Z | NM/PM/PS | NM/PM/PS |
| PM | Z/Z/NM | Z/Z/NS | NS/PS/NS | NM/PM/NS | NM/PM/Z | NM/PB/PS | NB/PB/PS |
| PB | Z/Z/PS | NS/Z/Z | NS/PS/Z | NM/PM/Z | NM/PB/Z | NB/PB/PB/ | NB/PB/PB |

## 2.4 Optimization of fuzzy PID controller based on improved PSO

The fuzzy PID controller can be effectively used to overcome the limitations of the conventional PID controller to dynamically adjust the parameters, its parameters cannot be tuned manually, which complicates the optimization process. Not only does this introduce a significant amount of subjectivity, but also makes it difficult to get the optimal parameter match, which will impact control performance. Also, the controller design cycle increases, which negatively impacts robustness and overall control quality. The approach adopted by this paper to address these problems is to come up with an optimization method of the controller that is based on an advanced particle swarm strategy. As a result, to optimize the fuzzy PID controller, an improved PSO algorithm is suggested accordingly.

### 2.4.1 Improved Particle Swarm Algorithm

This dead-zone-initiated adaptive particle swarm optimization algorithm, known as DZIA-PSO, has been proposed earlier in order to address the natural weaknesses of the PSO. Nevertheless, in the given algorithm, the population size, inertia weight and learning factors cannot be changed, and their values should be defined prior to the program run. With increasing the population size to a higher number, the chance to find the optimal solution goes up, but the computational load also becomes much higher. In the case of the robotic arm that is discussed in this paper the use of the fitness function implies that one call to the entire control system is made during each execution of the fitness function, which consumes a lot of time. To overcome these weaknesses, the present research additionally suggests a dynamic-population dead-zone-initiated adaptive particle swarm optimization algorithm, which is represented by DP-DZIA-PSO. The following four concepts: dead-zone initiation, dynamic population, inertia weight, and learning factor are presented and enhanced in this approach.

(1) “Dead zone” activation strategy

The dead-zone initiation policy is to define a circle of a certain radius in the middle of the local region, i.e., the dead zone, and then to check if a particle belongs to such an area based on the relevant regulations. When the particle comes into this range, it is re-initialized, allowing it to leave the local region. Hereby, when a particle enters the dead zone, it is restarted at a point outside the neighborhood of the local optimum. The exact procedure is stated as follows:

In particle iteration, to find out if a particle has ended up in a local optimum, the optimal solution of a particle is used to update its state. The detailed procedure is stated below:

Take the $k$ th step of evolutionary iteration as an example:

Order:

$$F_k^{\max} = \max\left\{ p_i^k \right\} \tag{15}$$

$$F_k^{\text{mean}} = \frac{1}{N}\sum_{i=1}^{N} p_i^k \tag{16}$$

where $p_i$ is the maximum adaptation value possessed by the $i$ th particle during the iteration history, and $F_k^{\max}$, $F_k^{\text{mean}}$ each represent the global optimal adaptation value and the average individual optimal adaptation value during that iteration.

Order:

$$H_1 = F_{k+1}^{\max} > F_k^{\max} \tag{17}$$

$$H_2 = F_{k+1}^{mean} > F_k^{mean} \tag{18}$$

Particle initialization:

$$X_{i,k}^{reset} = P_N + \left(\sigma_{i,1} - \sigma_{i,2}\right) X_k^{\max} \tag{19}$$

Particle initialization constraints limit the boundary:

$$X_k^{\max} = \frac{x^{\max}}{\max\left\{1, \left(K^{\max} - 1\right)\right\}} \tag{20}$$

$$x^{\max} = \left[ x_1^{\max}, x_2^{\max}, \cdots, x_r^{\max} \right] \tag{21}$$

$X_{i,k}^{\text{reset}}$ is the particle $i$ position after reinitialization. $P_N$ is the particle local extreme position. $\sigma_{i,1}$, $\sigma_{i,2}$ are random numbers between [0, 1]. $X_k^{\max}$ is the range of positions reinitialized at the $k$ th iteration. $K$ is the number of dead zones. $K^{\max}$ is the maximum number of dead zones, where $k^{\max} > 1$ is to be satisfied. $x^{\max}$ is the dynamic range of the particle in $R$ -dimensional space. $x_r^{\max}$ is the maximum position that the particle can be assigned in $r\left(1 < r < R\right)$ space.

After determining whether the particles evolve normally or not, the Sharing function is used to take exclusion operation for the particles in the dead zone. The specific steps are as follows:

$$sharing\left(d(i,j)\right) = \begin{cases} 1 - \dfrac{d(i,j)}{d_r}, \cdots d(i,j) < d_r \\ 0, \cdots d(i,j) \geq d_r \end{cases} \tag{22}$$

$d_r$ is the radius of the dead zone, given by the user. $d(i,j)$ is the distance between particles $i$ and $j$ . exclusion operation:

$$X_{i-new}^{n} = \frac{X_i^n}{1 - sharing(x)} \tag{23}$$

$X_i^n = \left( X_i^1, X_i^2, \cdots X_i^n \right)$ are the coordinates of the position of the particle $i$ at the $k$ th step. $X_{i-\text{new}}^{n}$ is the coordinate where particle $i$ is redistributed.

(2) Dynamic population strategy

Dynamic population strategy is added to the DZIA-PSO algorithm to adaptively change the size of the population during the particle evolution iteration process, so as to improve the convergence speed of the algorithm.

The main method of the dynamic population strategy is to perform evolutionary state estimation (ESE). Take the $k$ th step of evolutionary iteration as an example to illustrate the specific implementation steps of the DP strategy.

Calculate the average distance between the current particle $i$ and all particles:

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2} \tag{24}$$

where $N$ and $D$ denote the population size size and particle dimension, respectively.

The average distance $d_i$ of the globally optimal particle is denoted as $d_g$. If $d_g$ is smaller than $d_i$ of all other particles, the population is in a "converged" state. If $d_g$ is larger than the $d_i$ of all other particles, the population is in a "jumping out" state. If it is in between, it is not processed.

After determining the state of the particles, particles can be added or deleted according to the state. The deletion of particles obeys the principle of randomness, but the current position and the historical optimal position of the deleted particles cannot be optimal.

(3) Inertia weight improvement

In order to improve the performance of the algorithm, the improved PSO algorithm adopts a strategy to adaptively adjust the inertia weight $w$ with the current adaptation value. The formula is as follows:

$$w = \begin{cases} w_{\min} + \dfrac{(w_{\max} - w_{\min})(f - f_{\min})}{f_{ave} - f_{\min}} & f \le f_{ave} \\ w_{\max} & f > f_{ave} \end{cases} \tag{25}$$

$w_{\min}$, $w_{\max}$ are the set minimum and maximum inertia weights, respectively. $f$ is the adaptation value of the current particle. $f_{\min}$, $f_{\text{ave}}$ are the minimum adaptation value and the average adaptation value of the particle swarm, respectively.

(4) Learning factor improvement

The learning factors $c_1$ and $c_2$ represent the learning degree of own experience and group experience, respectively. The strategy to adjust the learning factor is to adjust $c_1$ and $c_2$ according to the relative difference between the adaptation value of each particle and the adaptation value of the local optimal particle and the global optimal particle:

$$\begin{cases} c_1 = k_1 \left| \dfrac{f(x_i^k) - f(p_{iN})}{f(p_{iN})} \right| \\ c_2 = k_2 \left| \dfrac{f(x_i^k) - f(p_{gN})}{f(p_{gN})} \right| \end{cases} \tag{26}$$

$k_1$, $k_2$ are the inertia factor weights of $c_1$ and $c_2$ respectively, ranging between $[0, 3.5]$. $f\left(x_i^k\right)$ is the fitness value of particle $i$ at the $k$ th iteration.

The complete flowchart of DP-DZIA-PSO algorithm is shown in Fig. 5.

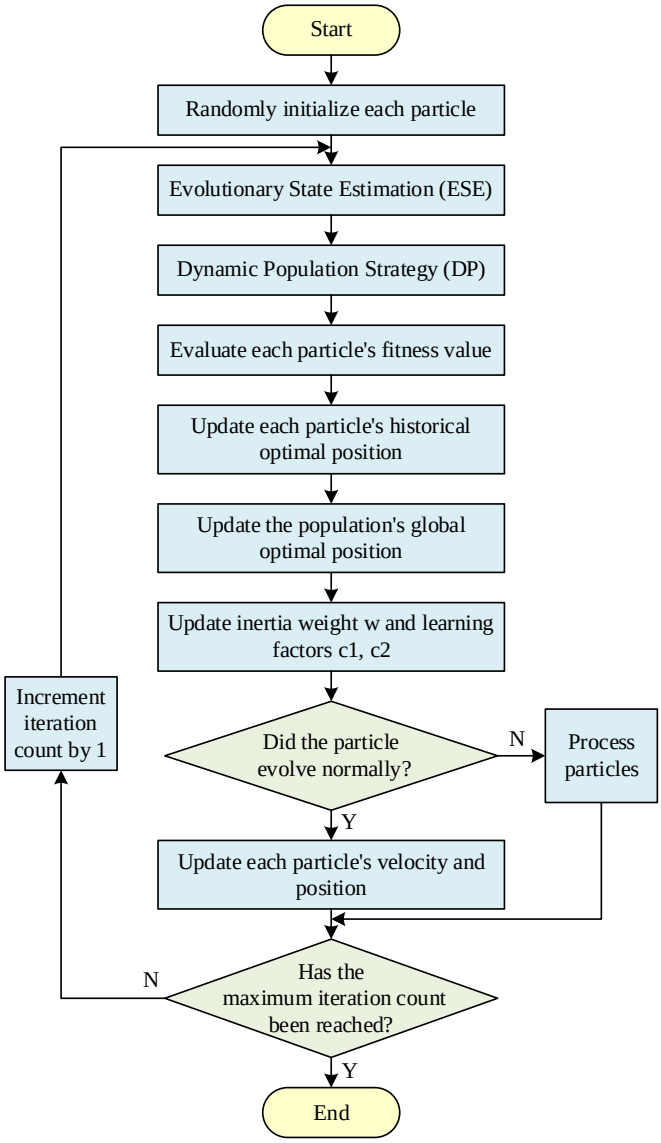



*Figure 5: DP-DZIA-PSO algorithm process*

### 2.4.2 Fuzzy PID controller optimization

In this study, the adjustable parameters of the controller are tuned offline with a particle swarm method to obtain an optimal parameter set and thereby improve control precision.

Subsequently, the fuzzy-control parameters in the fuzzy PID controller are further tuned through the DP-DZIA-PSO algorithm, and the corresponding schematic is presented in Fig. 6.

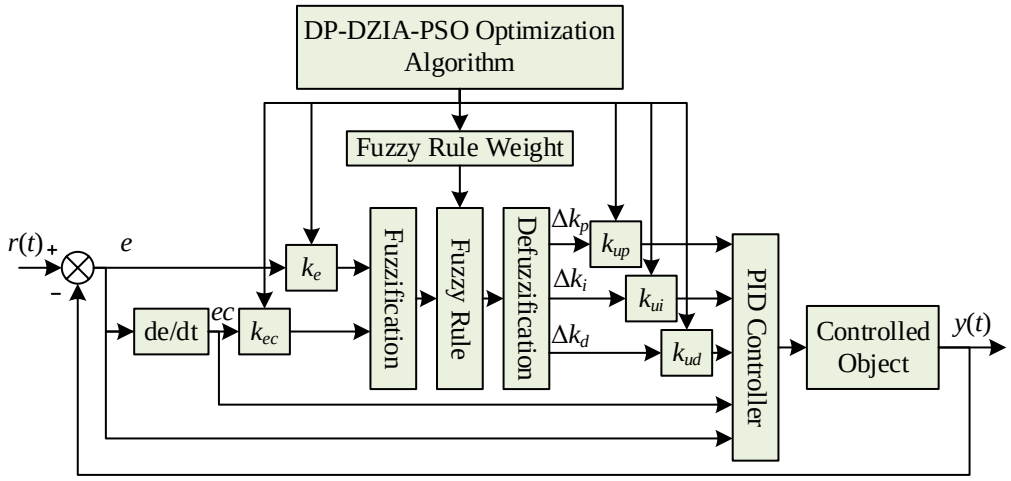



*Figure 6: Optimization principle diagram*

The parameters optimized by the improved particle swarm algorithm are the fuzzy rule weights and quantization scale factors $k_e$, $k_{ec}$, $k_{up}$, $k_{ui}$, and $k_{ud}$ in the fuzzy PID controller.

The particle swarm algorithm searches for the optimal solution on the basis of the fitness function. Since ITAE exhibits better performance in evaluating both dynamic and static characteristics of the control system, the ITAE index is selected as the fitness function in this study.

The procedure for optimizing fuzzy-controller parameters with the DP-DZIA-PSO algorithm can be divided into two stages:

(1) The DP-DZIA-PSO algorithm performs parameter reinitialization and updating for variables such as population size and learning factor according to the corresponding rules.

(2) The Simulink program, which acts on the controlled plant, calculates and compares the adaptation values of the particles.

Each particle in the DP-DZIA-PSO optimization process is subjected to a sequence of processes, which include initialization of the rule weights, quantization of the velocities and positions of the scaling factors, write the optimized parameters into the fuzzy controller to produce an updated controller, evaluate the fitness values, identify the best solution, renew the particle velocity and position. The operations are repeated until the maximum iteration number defined is obtained.

Figure 7 depicts the workflow of optimizing fuzzy-controller parameters using the enhanced particle swarm algorithm. Initially, the evolutionary state estimation (ESE) is applied to the particles to enable adapting the size of the population. Then, the fuzzy-controller parameters are set and the particles update their flight velocity and position. The initialized parameter set obtained at the current stage is then assigned to the fuzzy controller, and the Simulink program is run once to obtain the ITAE index at this time, i.e., the adaptive value, as a way of updating local and global optimums, as well as the inertia weights and learning factors. Finally, the particles are re-initialized or not according to the corresponding criteria, so that an iterative update is completed. The above process is looped until the maximum number of iterations is reached, and then the optimization search is finished.

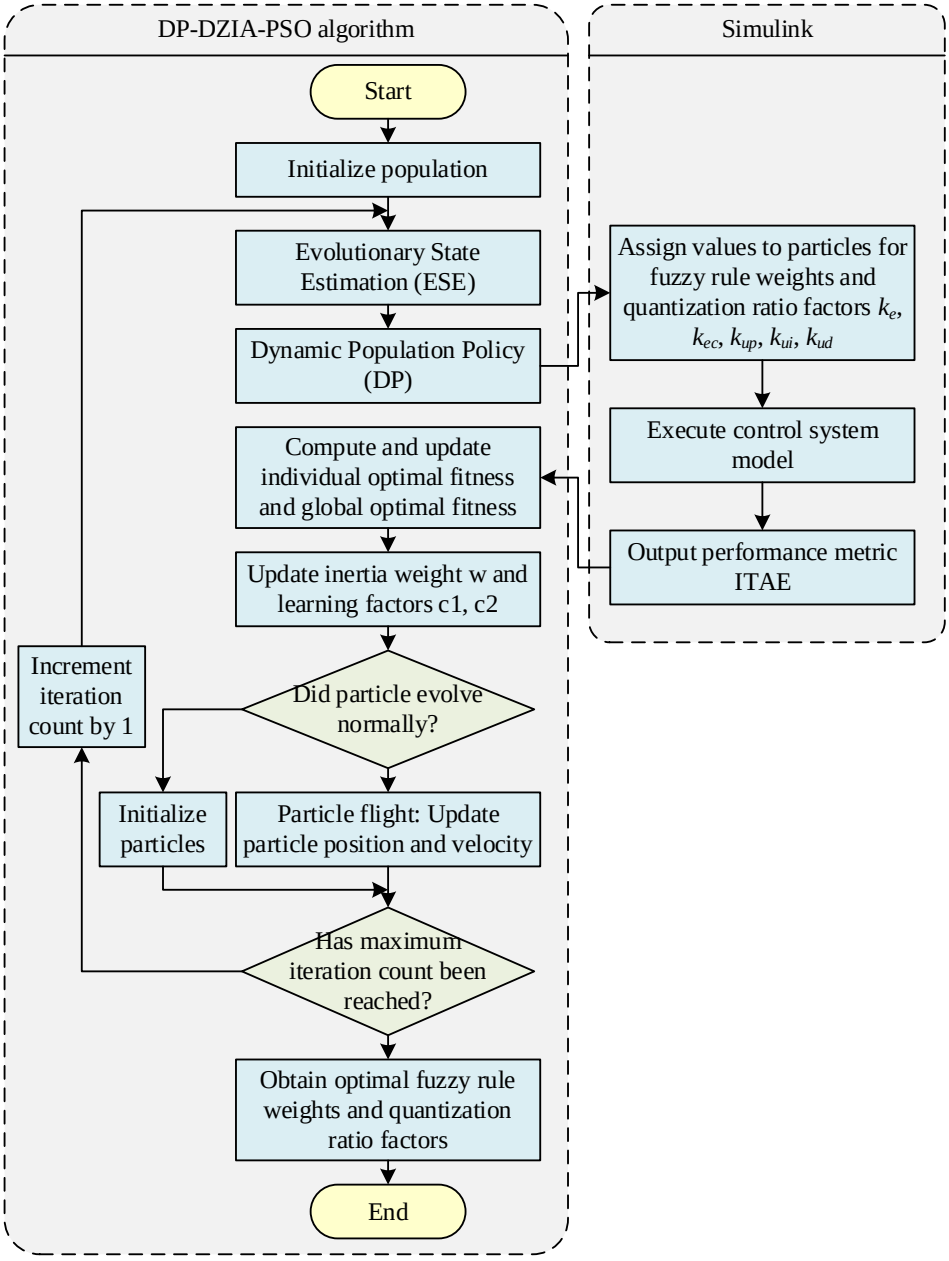



*Figure 7: DP-DZIA-PSO algorithm optimises fuzzy controller process*

# 3 Results

## 3.1 Dynamics Simulation

### 3.1.1 Simulation Parameter Settings

The solver of ADAMS uses the Lagrange equation method, which can be used to establish the dynamics equations and simulate and analyze the dynamics and kinematics of the virtual prototype. Here, ADAMS is used to analyze the moments required for the motion of each joint.

Because ADAMS is not suitable as a direct three-dimensional modeling tool, the model created in SolidWorks is used. In order to lessen the computational load, the six-degree-of-

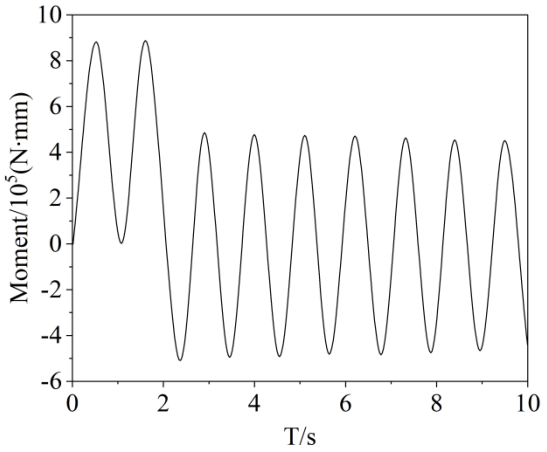
freedom robotic arm model built is simplified prior to importing it into ADAMS. The robotic arm model is constrained and all the parts are assembled into a single part. As per the design requirements, every joint of the robotic arm is driven by one of the motors. Hence, a rotary vice is installed on every one of the six joints, and the base is made stationary. The specific constraints are outlined in Table 3.

*Table 3: Model constraints*

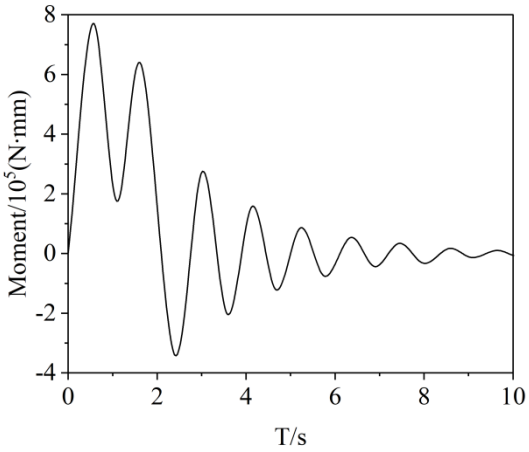
| Serial number | Motor pair | ADAMS Constraint name | Connection component |
|---|---|---|---|
| 1 | Fixed pair | Joint_1 | Base and earth |
| 2 | Rotating pair | Joint_2 | Shoulder joint and arm |
| 3 | Rotating pair | Joint_3 | The arm joints and the small arm |
| 4 | Rotating pair | Joint_4 | Small arms and wrist 1 |
| 5 | Rotating pair | Joint_5 | Wrist 1 joint and wrist 2 |
| 6 | Rotating pair | Joint_6 | Wrist 2 joint and wrist 3 |
| 7 | Rotating pair | Joint_7 | Wrist 3 joint and end finger unit |

After the constraints are added, the material of the six-degree-of-freedom robotic arm needs to be defined, considering the actual application environment, the said six-degree-of-freedom robotic arm is made of titanium alloy, density $\rho = 4.5\times10^{3} kg/m^{3}$, elastic modulus $E = 1.0\times10^{3}$GPa, Poisson's ratio $u$ = 0.33. Due to the limitations of the actual use of the working conditions, it is necessary to add a constant damping to the joints of the robotic arm and to ignore the effect of the gravitational acceleration.

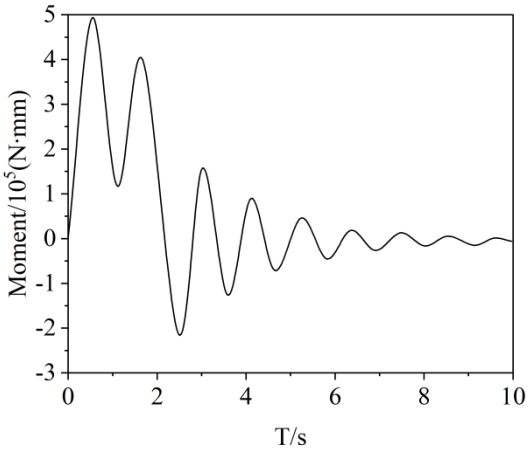
Because the six-degree-of-freedom robotic arm prototype is driven by motors, the drive is added at each rotating vice, and the corresponding motion function of each joint can be calculated according to the end motion speed (180mm/s) and the length of each arm in the design requirements. After inputting the driving function of each motor, a complete dynamics simulation model of the six-degree-of-freedom robotic arm prototype is established, and then the running time of 10s is selected, and the torque required for the rotation of each joint is obtained through simulation measurements, and the torque curves of each joint are shown in Fig. 8, and Figs. (a) ~ (f) are the torque curves of joints 1~6, respectively.



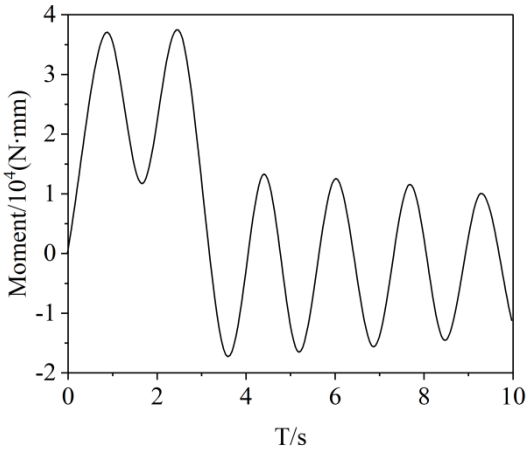

(a) Joint 1



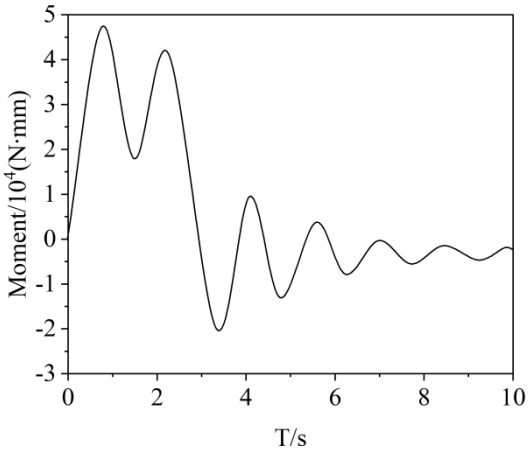

(b) Joint 2



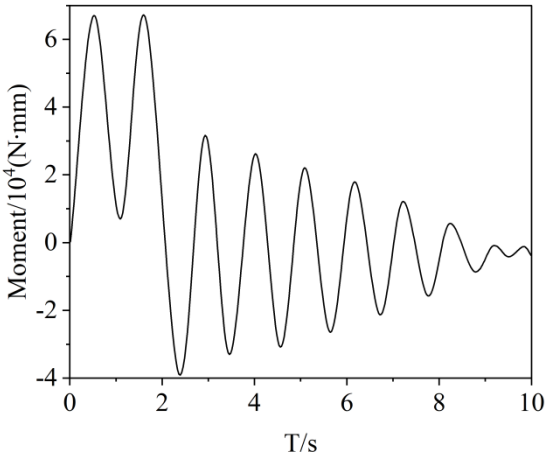

(c) Joint 3




(d) Joint 4




(e) Joint 5




(f) Joint 6

*Figure 8: Each joint moment curve*

### 3.1.2 Analysis of simulation results

The simulation findings of the six-degree-of-freedom robotic arm six-joint simulation reaching the target speed in ADAMS are presented in Figure 8. Examining the individual curves, it can be seen that every joint achieves the desired speed, and then the required torque curve slows down. It is due to the fact that the joint starts accelerating at zero speed to the highest target speed, and assuming that gravity is not considered, the drag torque does not change. Consequently, as the joint achieves the highest speed and goes into steady state, the necessary torque begins to reduce and eventually approach zero. Hence, upon achieving the maximum velocity, the joints run with a fixed velocity and the required torque reduces to zero over time. With the angular velocity plots of all the joints combined, it is possible to determine that the angular velocity keeps increasing until it becomes steady, thus the relevant torque profile can be extrapolated within reason. The torque curve peak value is used as the basis of motor selection of each joint and the resultant findings are as shown in table 4. The maximum moments required for the six-degree-of-freedom robotic arm joints 1~6 are $9\times10^5$N·mm, $8\times10^5$N·mm, $5\times10^5$N·mm, $4\times10^4$N·mm, $5\times10^4$N·mm, and $7\times10^4$N·mm, respectively.

*Table 4: Needs maximum torque for each joint*

| Joint | Maximum torque required /(N·mm) |
|---|---|
| Joint 1 | $9\times10^5$ |
| Joint 2 | $8\times10^5$ |
| Joint 3 | $5\times10^5$ |
| Joint 4 | $4\times10^4$ |
| Joint 5 | $5\times10^4$ |
| Joint 6 | $7\times10^4$ |

## 3.2 Analysis of optimization results

In order to evaluate the effectiveness of various optimization techniques, PSO, DZIA-PSO and DP-DZIA-PSO are used to optimize a fuzzy PID controller to track the trajectories. Parameter settings of these three optimization algorithms are provided in Table 5.

*Table 5: Three optimization algorithm parameters*

| Algorithm | PSO | Algorithm | DZIA-PSO | DP-DZIA-PSO |
|---|---|---|---|---|
| Population size | 200 | Population size | 200 | 200 |
| Iteration number | 100 | Iteration number | 100 | 100 |
| Selection probability | 0.9 | Inertia term | 0.6 | 0.6 |
| Cross probability | 0.9 | Maximum speed | 0.08 | 0.08 |
| Mutation probability | 0.07 | Mutation probability | 0.07 | 0.07 |

Fig. 9 shows the Pareto optimal fronts after optimization with the help of three algorithms. One can see that the Pareto front created by PSO is far worse than that created by the enhanced particle swarm algorithms, but the fronts found by DZIA-PSO and DP-DZIA-PSO are much alike. Hence, the enhanced particle swarm algorithm is more suitable to optimize fuzzy PID controllers. As shown in Fig. 9, the Pareto front of DP-DZIA-PSO is spread across a larger area, whereas the solution set based on DZIA-PSO is placed within a smaller area. Then, DZIA-PSO can output a solution set which is nearer to the actual Pareto frontier and the solutions related to it will be spread across a wider and more uniform area. Three representative solutions on the Pareto fronts of DZIA-PSO and DP-DZIA-PSO i.e. point A, B, and C in Fig. 9 are chosen to form the respective fuzzy PID controllers. The values of the parameters of the fuzzy PID controllers relating to optimization points A, B, C are given in Table 6. Additional information on the detailed fuzzy-rule parameters of the fuzzy PID controllers related to these three points of optimization are available in Tables 7 to 9.

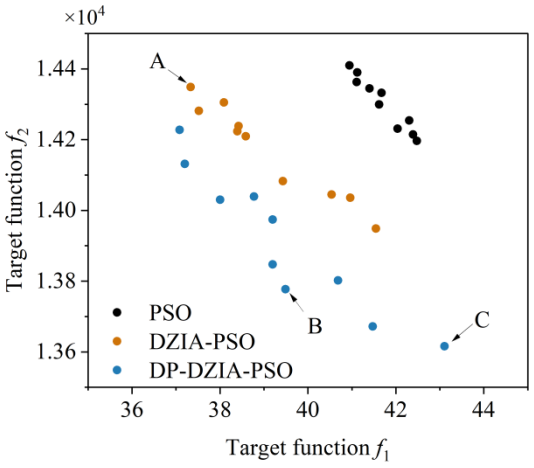



*Figure 9: Multi-objective optimized pareto front end*

*Table 6: Optimises the parameters of the fuzzy PID controller for the three points*

| Optimization points | Fuzzy variable | Domain of Discourse | The vertex position of the unilateral triangle of the membership function $X_1, X_2, X_3$ | | |
|---|---|---|---|---|---|
| A | $e$ | 4.341 | 0.797 | 2.288 | 3.958 |
| | $ec$ | 9.9 | 0.042 | 2.479 | 9.947 |
| | $\Delta K_P$ | 4.008 | 1.153 | 2.155 | 4.143 |
| | $\Delta K_I$ | 10 | 9.324 | 9.951 | 9.48 |
| | $\Delta K_D$ | 10 | 8.417 | 9.452 | 10 |
| B | $e$ | 3.581 | 0.308 | 1.561 | 3.422 |
| | $ec$ | 3.589 | 0.058 | 0.589 | 3.483 |
| | $\Delta K_P$ | 3.295 | 3.254 | 3.164 | 3.14 |
| | $\Delta K_I$ | 10 | 9.675 | 9.916 | 9.986 |
| | $\Delta K_D$ | 9.954 | 9.208 | 9.846 | 9.92 |
| C | $e$ | 9.231 | 0.236 | 1.095 | 9.156 |
| | $ec$ | 4.756 | 0.092 | 0.634 | 4.759 |
| | $\Delta K_P$ | 2.86 | 0.08 | 0.336 | 2.926 |
| | $\Delta K_I$ | 10 | 9.379 | 9.887 | 9.884 |
| | $\Delta K_D$ | 9.619 | 8.806 | 9.674 | 9.465 |

*Table 7: Optimization point A for fuzzy PID controller fuzzy rules*

| $\Delta K_P, \Delta K_I, \Delta K_D$ | | $e$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NB | NM | NS | Z | PS | PM | PB |
| $ec$ | NB | NB/NM/PB | NB/PM/PB | NB/PS/PB | NM/PS/PB | Z/PB/PB | Z/PB/NB | Z/NB/NB |
| | NM | NS/PM/PB | NS/PB/PB | NS/Z/PB | NS/PM/PB | NS/NB/PB | NS/NB/PB | NS/PS/NB |
| | NS | NS/PB/PB | NS/NM/PB | NS/Z/PB | NS/PS/PB | NS/PB/PB | NS/Z/PB | NS/PM/NB |
| | Z | NS/PB/PB | NS/Z/PB | Z/NB/PB | NS/NB/PB | NS/NB/PB | NS/Z/PB | NS/NB/NB |
| | PS | NS/PB/PB | NS/NM/PB | NS/NM/PB | NS/NB/PB | NS/NB/PB | NS/NB/PB | NS/NS/NB |
| | PM | NS/NB/PB | NS/PM/PB | NS/PB/PB | NS/PB/PB | NS/PB/PB | PS/NB/PB | NS/NS/PB |
| | PB | PM/Z/PB | PM/NB/PB | PM/PB/PB | PS/PB/PB | PB/NB/PB | PB/Z/PB | PB/NB/PB |

*Table 8: Optimization point B for fuzzy PID controller fuzzy rules*

| $\Delta K_P, \Delta K_I, \Delta K_D$ | | $e$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NB | NM | NS | Z | PS | PM | PB |
| $ec$ | NB | NB/NS/PB | NB/NM/PB | NB/PM/PB | NB/NB/PB | NB/PB/PB | PB/PS/NB | PB/NB/NB |
| | NM | NB/Z/PB | NB/NB/PB | NB/NB/PB | NB/PB/PB | NB/NS/PB | NB/Z/PB | PB/PB/NB |
| | NS | NB/PS/PB | NB/NB/PB | NB/NS/PB | NB/NB/PB | NB/PB/PB | NB/NB/PB | PB/NB/NB |
| | Z | NB/PB/PB | NB/PB/PB | NB/PB/PB | NB/NB/PB | NB/PB/PB | NB/NB/PB | PB/PB/NB |
| | PS | NB/PB/PB | NB/PM/PB | NB/NB/PB | NB/NB/PB | NB/NB/PB | NB/NB/PB | PB/NM/NB |
| | PM | NB/NB/PB | NB/Z/PB | NB/NB/PB | NB/NB/PB | NB/NM/PB | NB/Z/PB | NB/PB/NB |
| | PB | NB/PB/PB | NB/PB/PB | NM/PB/PB | NB/PB/PB | NB/PB/PB | NB/PS/PB | NB/NB/NB |

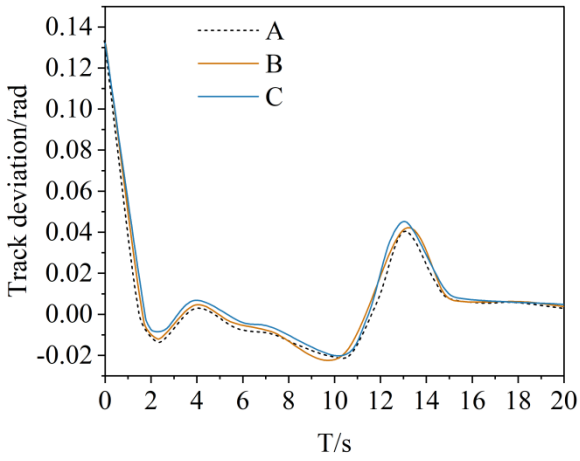
*Table 9: Optimization point C for fuzzy PID controller fuzzy rules*

| $\Delta K_P, \Delta K_I, \Delta K_D$ | | $e$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | NB | NM | NS | Z | PS | PM | PB |
| $ec$ | NB | NB/PS/NB | PB/PS/NB | PB/NS/PB | PB/NB/NB | PB/PS/NB | PB/PB/PB | PB/Z/NB |
| | NM | NB/PB/PB | NB/NB/PB | NB/PM/PB | NB/PB/PB | NB/NB/PB | NB/NB/PB | NB/PB/NB |
| | NS | NB/Z/PB | NB/PS/PB | NB/PB/PB | NB/NB/PB | NB/PB/PB | NB/NB/PB | NB/NB/NB |
| | Z | NB/Z/PB | NB/PB/PB | NB/PM/PB | NB/NB/PB | NB/PB/PB | NB/NB/PB | NB/NB/NB |
| | PS | NB/PB/PB | NB/PS/PB | NB/NB/PB | NB/NB/PB | NB/NBPB | NB/NB/PB | NB/NB/NB |
| | PM | NB/NM/PB | NB/NB/PB | NB/NB/PB | NB/NB/PB | NB/Z/PB | NB/Z/PB | NB/PM/NB |
| | PB | NB/PB/PB | NB/PB/PB | NB/PB/PB | NB/PB/PB | NB/PB/PB | NB/NS/PB | NB/NB/NB |

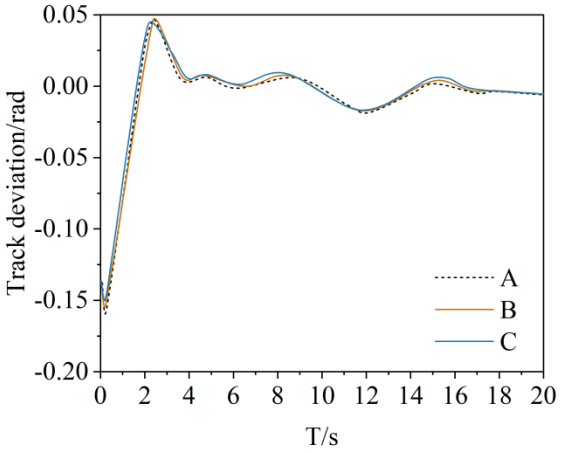
These fuzzy PID controllers are applied in simulations of articulated-robot trajectory tracking based on the aforementioned optimization points. The performance indices of the given controllers, namely, objective function 1, objective function 2, joint-angle deviation and controller-output patterns are presented in Table 10. Figures 10 depicts the deviations of the trajectory-tracking of the six joints of the articulated robot arm under the fuzzy PID controllers related to various optimization points, with Figs. (a) (b) (c) (d) (e) (f) representing Joint 1, Joint 2, Joint 3, Joint 4, Joint 5, and Joint 6 respectively. Figure 11 shows the torque changes generated by the controller at each joint, and Figs. (a) (b) (c) (d) (e) (f) also represent robot-arm Joints 1, 2, 3, 4, 5, and 6 respectively. Figure 12 illustrates the end-effector tracking deviation in Cartesian space. According to Table 10 and Figs. 10-12, the lowest trajectory-tracking deviation is observed by the fuzzy PID controller associated with optimization point A, and the lowest controller output is observed by the controller associated with optimization point C. The fuzzy PID controller associated with point B shows a middle level of performance when it comes to trajectory-tracking deviation and the controller output.

*Table 10: Optimization points corresponding to fuzzy PID controller control effect*

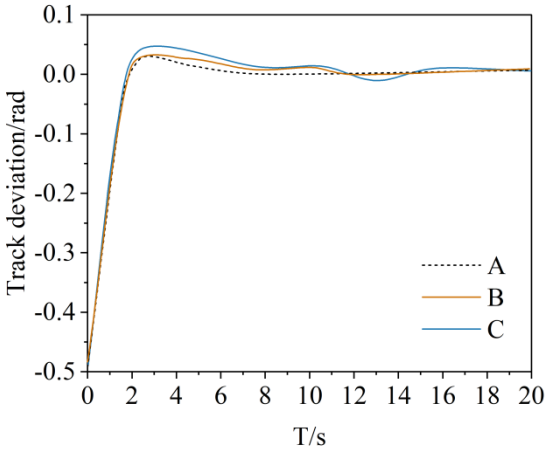
| Optimization points | A | B | C |
|---|---|---|---|
| Objective function 1 | 32.641 | 36.254 | 41.126 |
| Objective function 2 | 15354 | 15137 | 14662 |
| Joint Angle deviation norm | 4.135 | 4.211 | 4.259 |
| Controller output norm | 137.812 | 135.043 | 133.844 |



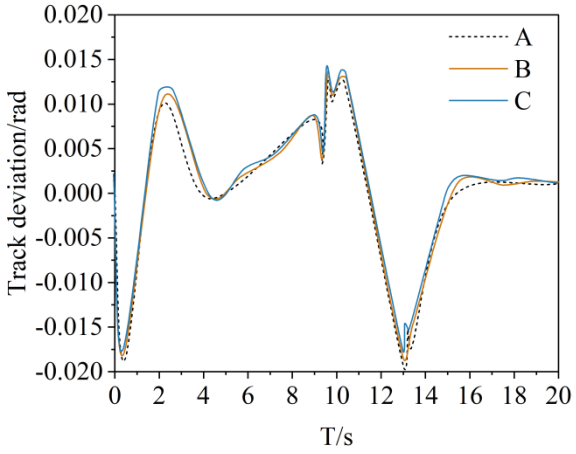

(a)Joint 1



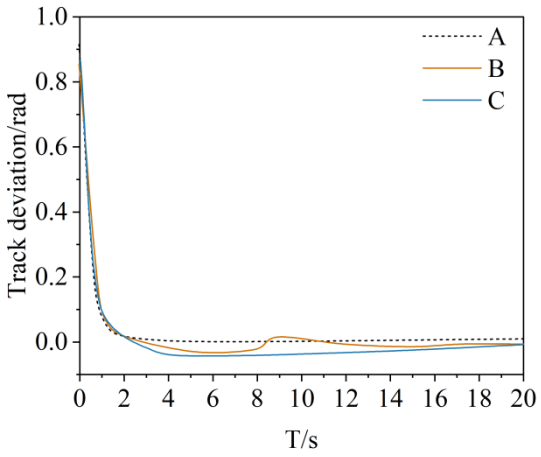

(b)Joint 2



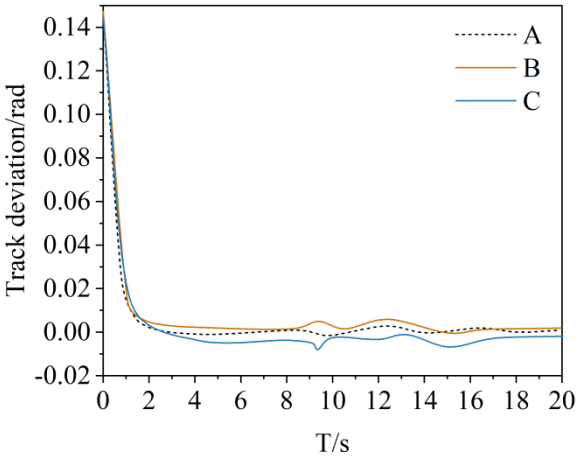

(c)Joint 3




(d)Joint 4

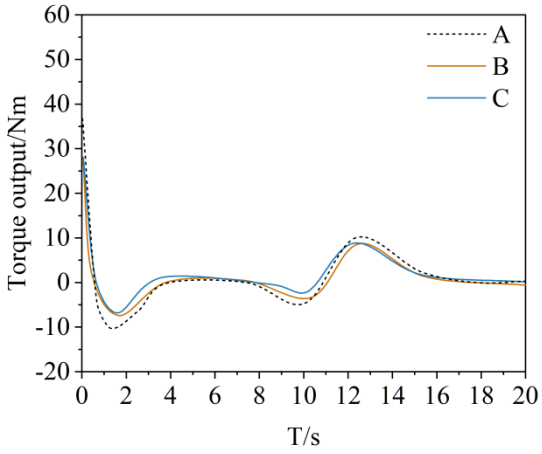



(e)Joint 5

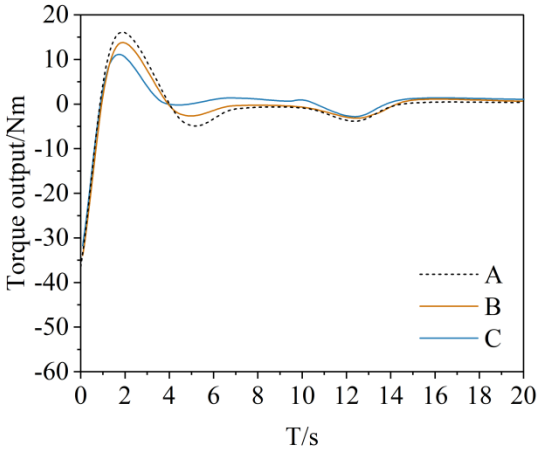



(f)Joint 6

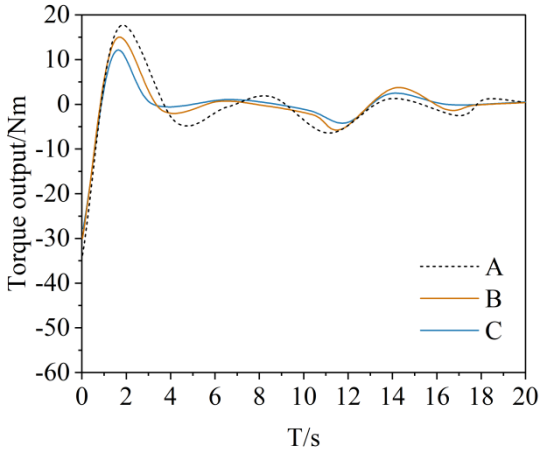
*Figure 10: The space trajectory of the joint space is tracked by deviation*



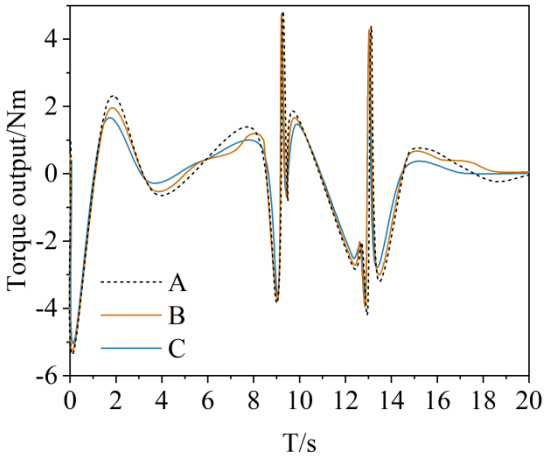

(a)Joint 1



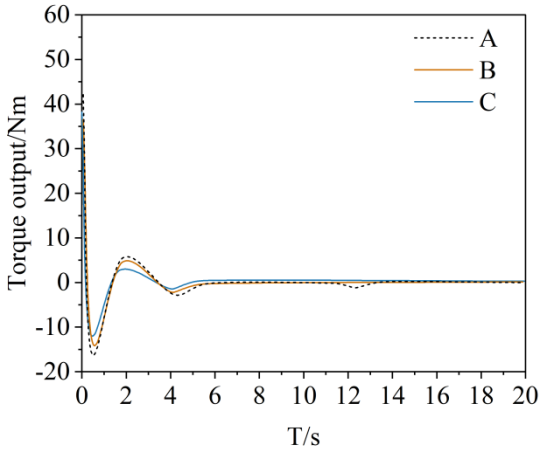

(b)Joint 2



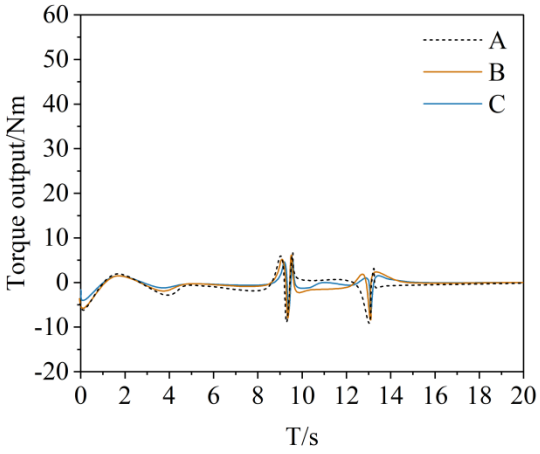

(c)Joint 3




(d)Joint 4




(e)Joint 5




(f)Joint 6

*Figure 11: Each joint controller torque output*

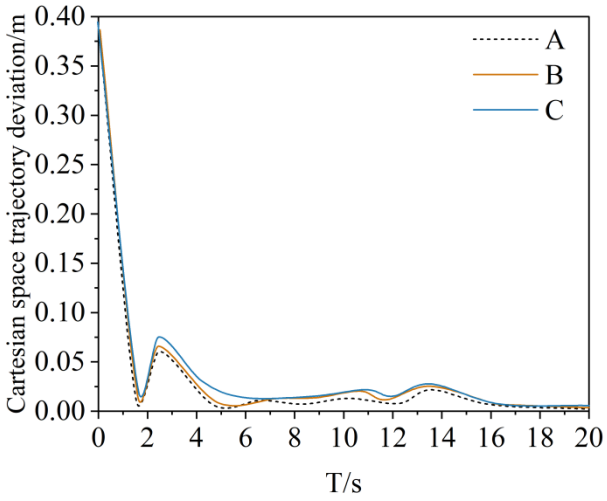



*Figure 12: The robot ends cartesian space deviation*

To sum up, the analysis of the optimization outcomes of PSO, DZIA-PSO and DP-DZIA-PSO has shown that the best fuzzy PID controller optimization outcomes are achieved using DP-DZIA-PSO, which confirms the usefulness of the modified particle swarm approach suggested in this paper to optimize the fuzzy PID controller. The improved particle swarm algorithm can give a better Pareto frontier than the standard particle swarm algorithm when used on the optimization of fuzzy PID controllers in articulatory-robot pathway tracking, and the right solution in the Pareto set may then be chosen based on particular engineering needs. Other simulation results also show that controllers with various optimization points have different control properties at various control goals.

# 4 Conclusion

In order to enhance the accuracy of the trajectory-tracking of a robotic arm, this paper suggests an application of a fuzzy PID controller to a six-degree-of-freedom robotic arm. In order to make fuzzy PID control even more efficient at robotic-arm trajectory tracking, a dynamic-population dead-zone-initiated adaptive particle swarm optimization algorithm is presented to optimize the fuzzy PID controller. The simulation and analysis of the motion of the six-degree-of-freedom robotic arm and the torques needed by the six joints are carried out using the ADAMS software $9\times10^5$, $8\times10^5$, $5\times10^5$, $4\times10^4$, $5\times10^4$, and $7\times10^4$N·mm, respectively. Besides, optimization experiments on trajectory trackers fuzzy PID controllers are conducted with the use of PSO, DZIA-PSO, and DP-DZIA-PSO algorithms whereas parallel controller optimization experiments are conducted with enhanced PSO algorithm. The Pareto frontier produced by the new approach is superior to that of the conventional PSO algorithm. Pareto frontier solutions of DZIA-PSO and DP-DZIA-PSO are chosen based on their respective Pareto frontiers and the corresponding fuzzy PID controllers constructed. The solution achieved by DZIA-PSO is that of a small-output fuzzy PID controller that has the lowest overall performance but the greatest combined performance in terms of both tracking error and output torque. Conversely, the solutions developed by DP-DZIA-PSO are more versatile, and they may be selected depending on various conditions of operation.

# About the Author

Yang Zhang (June 1983), male, of Han ethnicity, was born in Gaizhou City, Liaoning Province. He works in the Department of Vehicle Engineering, Shanxi Vocational and Technical College, with the academic title of associate professor. His main research directions include the development and application of programmable logic controllers (PLCs) and communication technologies.

# References

[1] Haleem, A., Javaid, M., Singh, R. P., Rab, S., & Suman, R. (2021). Hyperautomation for the enhancement of automation in industries. Sensors International, 2, 100124.

[2] Pramod, D. (2022). Robotic process automation for industry: adoption status, benefits, challenges and research agenda. Benchmarking: an international journal, 29(5), 1562-1586.

[3] Mihola, M., ZEMAN, Z., BOLESLAVSKY, A., Bém, J., Pastor, R., & Fojtík, D. (2022). Automation of design of robotic arm. MM Science Journal, 2022, 5876-5882.

[4] Mathavan Jeyabalan, P. K., Nehrujee, A., Elias, S., Magesh Kumar, M., Sujatha, S., & Balasubramanian, S. (2023). Design and characterization of a self-aligning end-effector robot for single-joint arm movement rehabilitation. Robotics, 12(6), 149.

[5] Aboelhassan, A., Abdelgeliel, M., Zakzouk, E. E., & Galea, M. (2020). Design and implementation of model predictive control based PID controller for industrial applications. Energies, 13(24), 6594.

[6] Batiha, I. M., Njadat, S. A., Batyha, R. M., Zraiqat, A., Dababneh, A., & Momani, S. (2022). Design fractional-order PID controllers for single-joint robot arm model. Int. J. Adv. Soft Comput. Appl, 14(2), 96-114.

[7] Nouman, K., Asim, Z., & Qasim, K. (2018, May). Comprehensive study on performance of PID controller and its applications. In 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC) (pp. 1574-1579). IEEE.

[8] Borase, R. P., Maghade, D. K., Sondkar, S. Y., & Pawar, S. N. (2021). A review of PID control, tuning methods and applications. International Journal of Dynamics and Control, 9(2), 818-827.

[9] Chotikunnan, P., & Chotikunnan, R. (2023). Dual design PID controller for robotic manipulator application. Journal of Robotics and Control (JRC), 4(1), 23-34.

[10] Chia, K. S. (2018). Ziegler-nichols based proportional-integral-derivative controller for a line tracking robot. Indonesian journal of electrical engineering and computer science, 9(1), 221-226.

[11] Larbah, I. (2025). Optimization of a PID Controller for a Single-Link Robotic Arm Using a Genetic Algorithm. The International Journal of Engineering & Information

Technology (IJEIT), 14(1), 45-50.

[12] Feng, H., Yin, C. B., Weng, W. W., Ma, W., Zhou, J. J., Jia, W. H., & Zhang, Z. L. (2018). Robotic excavator trajectory control using an improved GA based PID controller. Mechanical Systems and Signal Processing, 105, 153-168.

[13] Yuan, T., Guo, G., Du, B., Zhao, Z., & Xu, W. (2021). The adaptive sliding mode control using improved genetic algorithm tuning PID controller for the planetary rover. Aircraft Engineering and Aerospace Technology, 93(1), 218-226.

[14] Yu, Y., Song, M., Chen, Z., Wu, H., & Ouyang, H. (2024, June). Design of PID controller for robot arm based on genetic algorithm and fuzzy theory. In Fourth International Conference on Mechanical, Electronics, and Electrical and Automation Control (METMS 2024) (Vol. 13163, pp. 307-312). SPIE.

[15] Yang, Y., Gao, Y., Wu, J., Ding, Z., & Zhao, S. (2024). Improving PID controller performance in nonlinear oscillatory automatic generation control systems using a multi-objective marine predator algorithm with enhanced diversity. Journal of Bionic Engineering, 21(5), 2497-2514.

[16] Jawad, A. T., Ali, N. S., Abdullah, A. N., & Alwash, N. H. (2021, July). Design of adaptive controller for robot arm manipulator based on ANN with optimized PID by IWO algorithm. In 2021 International Conference on Advanced Computer Applications (ACA) (pp. 107-111). IEEE.

[17] Elkhateeb, N. A., & Badr, R. I. (2017). Novel PID tracking controller for 2DOF robotic manipulator system based on artificial bee colony algorithm. Electrical, Control and Communication Engineering, 13(1), 55-62.

[18] Attya, S. M. (2022). Optimized PID Controller based on Artificial Bee Colony Algorithm for Robot Arm. International Journal of Software & Hardware Research in Engineering (IJSHRE), 10(1), 88-97.

[19] Huang, H. C., & Chuang, C. C. (2020). Artificial bee colony optimization algorithm incorporated with fuzzy theory for real-time machine learning control of articulated robotic manipulators. IEEE Access, 8, 192481-192492.

[20] Wazzan, A. N., Basil, N., Raad, M., & Mohammed, H. K. (2022). PID controller with robotic arm using optimization algorithm. Int. J. Mech. Eng, 7(2), 3746-3751.

[21] Dahmane, S. A., Azzedine, A., & Megueni, A. (2020). Ant colony optimization algorithm based on optimal PID parameters for a robotic arm. International Journal of Control Systems and Robotics, 5.

[22] Mirrashid, N., Alibeiki, E., & Rakhtala, S. M. (2022). Development and control of an upper limb rehabilitation robot via ant colony optimization-PID and fuzzy-PID controllers. International Journal of engineering, 35(8), 1488-1493.

[23] Karami, M., Tavakolpour-Saleh, A. R., & Norouzi, A. (2020). Optimal nonlinear PID control of a micro-robot equipped with vibratory actuator using ant colony algorithm: Simulation and experiment. Journal of Intelligent & Robotic Systems, 99(3), 773-796.

[24] Joyo, M. K., Raza, Y., Kadir, K., Naidu, K., Ahmed, S. F., & Khan, S. (2019, August). Firefly optimised pid control for upper extremity rehabilitation robot. In 2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA) (pp. 1-5). IEEE.

[25] Ab Talib, M. H., Ismail, M. I. S., Mohd Yatim, H., Hadi, M. S., Mohd Saufi, M. S. R., Abdul Saad, W. A., ... & Isham, M. F. (2023, August). Vibration Suppression of the Flexible Beam Structure Using PID Controller Tuned by Advanced Firefly Algorithm. In Innovative Manufacturing, Mechatronics & Materials Forum (pp. 169-179). Singapore: Springer Nature Singapore.

[26] Mohamed, M. J., Al-Araji, A. S., & Oleiwi, B. K. (2025). Hybrid Controllers Design Combining FOPID Operations and Neural Networks for a Three-Link Rigid Robot Manipulator Using Firefly Optimization Algorithm. International Journal of Intelligent Engineering & Systems, 18(10).

[27] Lee, Y. S., & Jang, D. W. (2021). Optimization of neural network-based self-tuning PID controllers for second order mechanical systems. Applied Sciences, 11(17), 8002.

[28] Xiao, W., Chen, K., Fan, J., Hou, Y., Kong, W., & Dan, G. (2023). AI-driven rehabilitation and assistive robotic system with intelligent PID controller based on RBF neural networks. Neural Computing and Applications, 35(22), 16021-16035.

[29] Lin, R. (2023, July). Research on manipulator control based on improved PID algorithm. In 3rd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC 2023) (Vol. 12717, pp. 196-200). SPIE.

[30] Ibraheem, G. A. R., Azar, A. T., Ibraheem, I. K., & Humaidi, A. J. (2020). A novel design of a neural network-based fractional PID controller for mobile robots using hybridized fruit fly and particle swarm optimization. Complexity, 2020(1), 3067024.

[31] Moshayedi, A. J., Li, J., Sina, N., Chen, X., Liao, L., Gheisari, M., & Xie, X. (2022). Simulation and validation of optimized pid controller in agv (automated guided vehicles) model using pso and bas algorithms. Computational Intelligence and Neuroscience, 2022(1), 7799654.

[32] Yadav, S., Kumar, S., & Goyal, M. (2024). Trajectory control and optimization of PID controller parameters for dual-arms with a single-link underwater robot manipulator. Journal of the Chinese Institute of Engineers, 47(7), 830-840.

[33] Zhou, Y., He, X., Shao, F., & Zhang, X. (2024, October). Research on the optimization of the PID control method for an EOD robotic manipulator using the PSO algorithm for BP neural networks. In Actuators (Vol. 13, No. 10, p. 386). MDPI.

[34] Akkar, H. A., & Haddad, S. Q. G. (2020). Design Stable Controller for PUMA 560 Robot with PID and Sliding Mode Controller Based on PSO Algorithm. International Journal of Intelligent Engineering & Systems, 13(6).

[35] Wu, K. J., & Chen, M. Y. (2023, October). Controller with the PID Parameters Optimization by PSO for a 6-DOF Robotic Arm. In 2023 International Conference on Fuzzy Theory and Its Applications (iFUZZY) (pp. 1-6). IEEE.