Simulation and Analysis of an Adaptive Manipulator Based on Machine Learning

Zihan Wang *

School of Ocean Engineering, Harbin Institute of Technology, Weihai, 264209, China

* Corresponding Author Email: 2024210535@stu.hit.edu.cn

Abstract. This study takes the PUMA560 manipulator as the research object, selects its first three rotating joints to construct a 3-degree-of-freedom (3-DOF) manipulator system, and aims to improve the control accuracy of the manipulator and environmental adaptability. First, the Denavit-Hartenberg (DH) parameter method is used to establish the manipulator's kinematic model; the MATLAB Robotics Toolbox is employed to define the link coordinate system, joint angles, link lengths, and other DH parameters, and a velocity ellipsoid is constructed to analyze the motion characteristics of the end effector. In the trajectory planning stage, the Joint Trajectory Planning (JTRAJ) function (for smooth interpolation in joint space) and the Cartesian Trajectory Planning (CTRAJ) function (for linear path interpolation at the end effector) are compared, and finally, the JTRAJ function is selected to plan the target path. To achieve precise control of the manipulator, a PID (Proportional-Integral-Derivative) control simulation model based on Simulink is built. Simulation results show that compared with manual parameter tuning, the PID control optimized by Genetic Algorithm (GA) has a smaller error fluctuation range, and the error tends to be stable at the end of the simulation, significantly improving the manipulator's control robustness and trajectory accuracy. This study realizes the effective integration of machine learning and traditional control algorithms, providing a feasible solution for the optimization of manipulator adaptive control; future research can further explore the influence of parameter ranges and fitness function definitions on control effects, and improve data horizontal comparison to deepen the research.

Keywords: Machine Learning; PID; 3-DOF Manipulator; Genetic Algorithm.

1. Introduction

After entering the industry 4.0 era, with the continuous emergence of new technologies, manipulators have gradually transformed into intelligent control modes [1]. Relying on precise and diverse sensors, control algorithms can achieve precise control of the output of manipulators based on real-time data, such as acceleration. In the future, manipulators integrated with new technologies are expected to replace some repetitive tasks and become a more intelligent and efficient new productive force.

The pose of the manipulator's end effector is determined by three position variables and three attitude variables. Therefore, to achieve complete pose control, a manipulator generally needs to have six degrees of freedom. According to actual application scenarios and requirements, manipulators may also have under-actuated configurations (with fewer than six degrees of freedom) or redundant configurations (with more than six degrees of freedom). As the number of degrees of freedom increases, the flexibility of the manipulator improves, but its stiffness decreases accordingly, and the solution of inverse kinematics also becomes more complex [2]. Therefore, for different scenarios, it is necessary to comprehensively consider the advantages and disadvantages of various manipulator structures to determine the optimal manipulator structure.

When controlling the motion of a manipulator, the Denavit-Hartenberg (DH) parameter method is usually used to solve its kinematic equations to obtain the homogeneous transformation matrix from the base coordinate system to the Cartesian coordinate system, and the Lagrange method is used to solve the dynamic model. Then, based on these two models, methods such as adaptive control and sliding mode control are used to achieve precise control of the manipulator. Many researchers have also integrated machine learning methods with traditional parameter tuning methods to improve the effectiveness and accuracy of parameters. For example, researchers such as Azimirad applied spiking

neural networks to the parameter optimization of Fractional Order PID (FOPID), which enabled the 4-link manipulator to maintain integral absolute errors (IAE) of 0.007631 and 0.004124 for Link 1 and Link 2 respectively when facing viscous friction disturbances, significantly outperforming the traditional FOPID method [3]. Mohamed et al. integrated the Zebra Optimization Algorithm (ZOA) into the tuning process of various PID and FOPID structures; when the mass of the manipulator link increased by 5% or the manipulator was subjected to an external disturbance of sin(50t), the controller could still maintain excellent performance [4]. However, researchers often start directly from the algorithm perspective, lacking a systematic process from manipulator selection, manipulator construction, to motion performance analysis, resulting in deficiencies in the logic of the entire optimization process.

Machine learning includes multiple branches such as Neural Networks (NN), Genetic Algorithms (GA), and Reinforcement Learning (RL). Among them, the GA algorithm draws on the theory of natural selection and achieves control objectives in the context of large-scale search through methods such as crossover and mutation [5-6]. The GA algorithm is widely used to find the optimal solution of parameters to achieve the purpose of adjusting and optimizing them [6]. Compared with the above-mentioned machine learning methods, the genetic algorithm does not rely on specific mathematical models, making it suitable for optimizing nonlinear models such as manipulators. In addition, GA inherently has excellent parallelism, so it is widely used in fields such as flight scheduling and route scheduling [7].

This study selects the first three links of the PUMA560 manipulator and focuses on studying the rationality of its structure and the scientificity of the selection of motion trajectory strategies. Then, a PID control model is built to perform dynamic control of the manipulator, and GA is combined for parameter selection and optimization to achieve adaptive parameter adjustment through machine learning.

2. Methods

2.1. Construction of 3-DOF Manipulator Kinematic Equations

2.1.1 Manipulator model construction using DH parameters

In this study, a 3-DOF manipulator with three rotating joints (RRR) is selected for research. Link coordinate systems {1}, {2}, and {3} are established for the three links, respectively. The Z-axis of the coordinate system coincides with the joint axis, the straight line where the X-axis lies is parallel to the common perpendicular of the previous joint axis and the current joint axis, the intersection point of the X-axis and the current joint axis is set as the origin, and the Y-axis direction is determined by the right-hand rule. Based on this, the DH parameter method can be used to describe its structure. The standard DH parameters of the manipulator are shown in Table 1.

Table 1. Specific DH parameter of the manipulator

j	theta	d	a	alpha	offset
1	q1	0	0	1.5708	0
2	q2	0	0.4318	0	0
3	q3	0.15005	0.0203	-1.5708	0

Using the Robotics Toolbox, the basic model of the manipulator, as shown in Fig. 1, can be established.

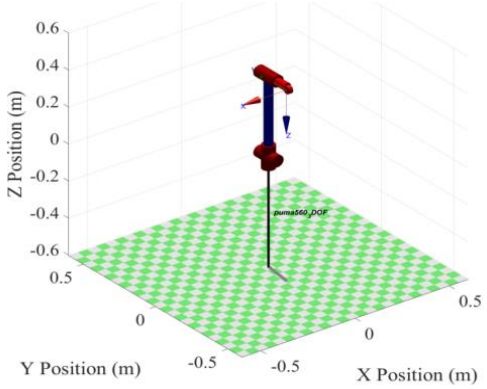

Fig. 1 Basic model of manipulator (Picture credit: Original)

2.1.2 Drawing and studying the velocity ellipsoid of the manipulator

By establishing a velocity ellipsoid, the operating speed of the manipulator's end effector in various directions can be clearly and intuitively displayed. First, the Jacobian matrix is used to establish the relationship between the joint velocity and the end velocity, which is expressed as

$$v=J(q)\cdot qd. \tag{1}$$

where J(q) represents the Jacobian matrix transformed from the Cartesian space to the joint angle space, and qd represents the joint angular velocity. Then, the modulus of the joint velocity vector is set to be no more than 1, that is

$$qd^{T}\cdot qd\leq 1. \tag{2}$$

Substituting Equation (1) into Equation (2), we can obtain

$$v^{T}\left[1/\left(J\cdot J^{T}\right)\right]\cdot v. \tag{3}$$

In Equation (3), v represents the end velocity, and the part of J·JT is called the Gram matrix. Substituting this matrix into the ellipsoid drawing function built in the Robotics Toolbox, the velocity ellipsoid of the manipulator's end effector can be obtained. The joint angle selected here is a relatively central non-singular configuration [0, π/2, -π/3].

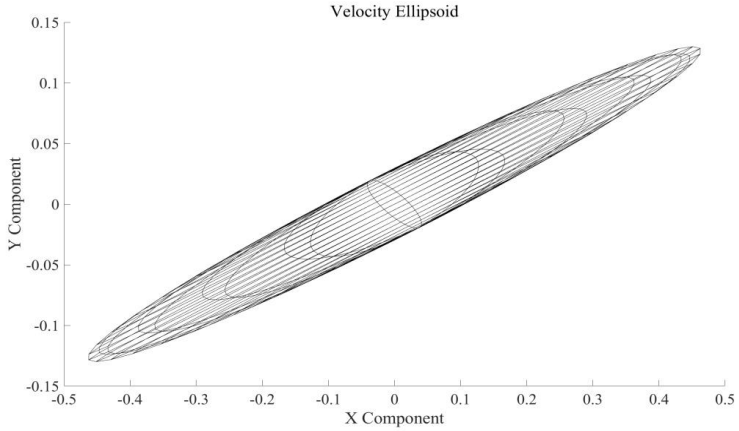

Fig. 2 Velocity ellipsoid of end effector (Picture credit:Original)

Fig. 2 is drawn on the XOY plane, and the values of the horizontal and vertical axes both represent proportions, indicating the distribution of the movement capability of the manipulator's end effector in various directions. It can be seen from Fig. 2 that the end effector has the best movement speed along the long-axis direction of the ellipsoid in Fig. 2, and the movement capability along the short-axis direction is poor. However, in general, this manipulator has good all-round working capabilities and is suitable as the analysis object of this study.

2.1.3 Trajectory planning

In this study, the manipulator trajectory is planned from two perspectives: the joint space and the end effector. The JTRAJ function plans the path through smooth interpolation between the initial joint vector and the target joint vector, and the trajectory of the end effector can be obtained by performing a forward kinematic transformation on the planned intermediate joint vector. The other function, CTRAJ, directly generates a linear path through interpolation between the initial position vector and the final position vector of the end effector.

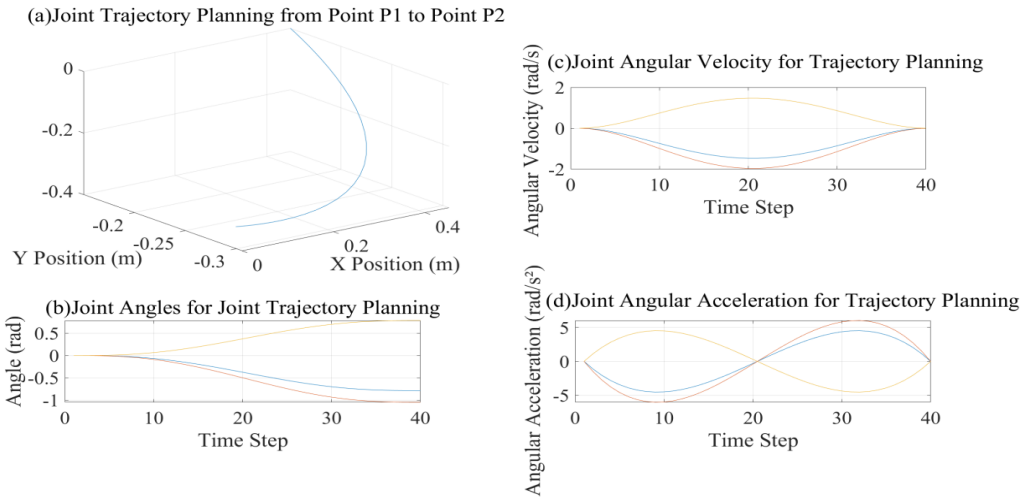

Fig. 3 Joint Trajectory planning (a)Joint planning from point P1 to point P2;(b) Joint angles for joint trajectory planning;(c) Joint angular velocity for trajectory planning;(d) Joint angular acceleration for trajectory planning (Picture credit: Original)

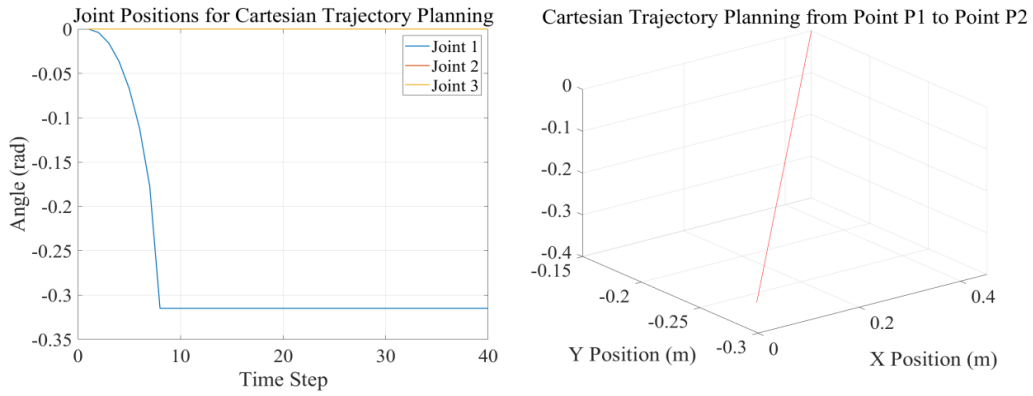

Fig. 4 Cartesian trajectory planning, (a)Joint positions for the Cartesian trajectory planning; (b) Cartesian trajectory planning from point P1 to point P2 (Picture credit: Original)

The unit of the horizontal axis in Figs. 3(b), 3(c), and 3(d) is seconds. It can be seen from Fig. 3 that joint space planning can make the joint angle, joint angular velocity, and joint angular acceleration change smoothly with time steps, but the end path is not a straight line. When no obstacles are set, its efficiency is lower than that of the linear path planned by the CTRAj function shown in Fig. 4. However, when the path planned by the CTRAj function is converted into a joint vector, a relatively complex inverse kinematics function must be used to solve its numerical solution, resulting in slow calculation speed and poor accuracy. It can be seen from Fig. 4(a) that trajectory planning based on the end effector is prone to sudden changes in joint angles, which makes it difficult to meet the requirement of smooth joint trajectories. Considering the above advantages and disadvantages and combining with the actual situation of this research topic, the JTRAJ function with better path planning performance is selected in this study for planning the target path.

2.2. PID

2.2.1 Overview of PID principle

PID is widely used in the adjustment process for feedback signals [5], and its specific formula is

$$u(k)=K_p\left[e(k)+\frac{T_s}{T_i}\sum_{i=0}^{k}e(i)+\frac{T_d}{T_s}(e(k)-e(k-1))\right]. \quad (4)$$

Where u(k) refers to the controller output at the k-th sampling moment; K_p refers to the Proportional Gain; e(k)refers to the error at the k-th sampling moment; T_s refers to the time interval between two samplings;T_i refers to the Integral Time Constant. T_d refers to the derivative time constant; e(i) refers to the error at the (i)-th sampling moment. This formula is derived based on rectangular integration and backward difference, and it is converted into an intuitive form with separated p, i, and d parameters, that is

$$u(k)=K_p\cdot e(k)+K_i\cdot\sum_{i=0}^{k}e(i)+K_d\cdot(e(k)-e(k-1)). \quad (5)$$

Here,K_p, K_i and K_d represent the integral gain and derivative gain respectively.By adjusting the three parameters K_p, K_i and K_d, specific control for different environments and different objects can be realized. In the field of control engineering, how to use appropriate K_p, K_i and K_d to improve system stability is still one of the main focuses [8]. In addition to the traditional trial-and-error method and Z-N tuning method, in recent years, methods for parameter tuning using machine learning technologies such as genetic algorithms and reinforcement learning have emerged [9], thereby reducing the labor cost in the optimization and adjustment process of the PID algorithm and making the system more adaptive and flexible [10].

2.2.2 PID simulation analysis based on MATLAB

In this study, the Simulink module built in MATLAB is used to simulate the PID control of the manipulator, and its simulation model is shown in Fig. 5

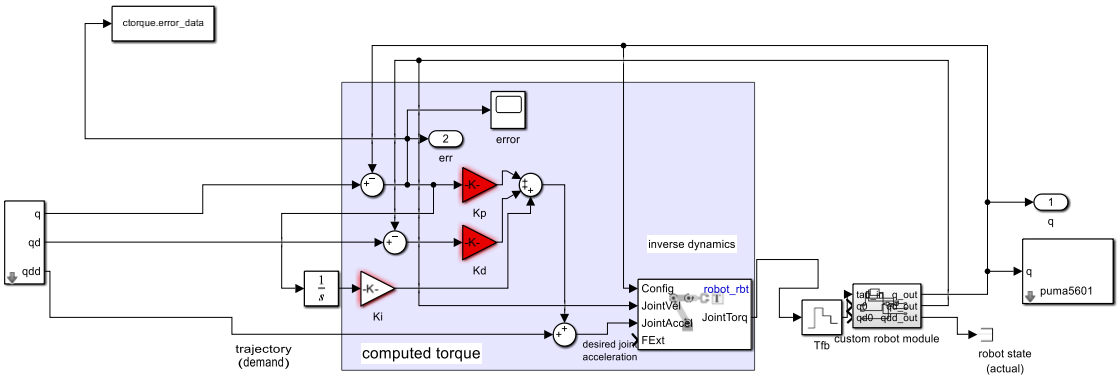

Fig. 5 Simulation Logic Diagram (Picture credit: Original)

The joint trajectory planning module in Fig. 5 plans the trajectory based on the initial joint angle [0, 0, 0] and the target joint angle [-π/4, -π/3, π/4], ensuring that the manipulator completes the task within 5 seconds, and then outputs the joint angle (JA), joint angular velocity (JAV), and joint angular acceleration (JAA) corresponding to each step. These three matrices are subtracted from the actual JA, JAV, and JAA fed back from the backend, and the resulting errors are subject to the gains of the three parametersK_p, K_i and K_d.The sum of the three parameter gains is used as the actual joint angular acceleration required for inverse kinematics to participate in the calculation. Combined with the target JA and JAV obtained by the joint trajectory planning method, the joint

torque to be output in this step is calculated. The formula for calculating the joint torque is

$$\tau=M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q). \quad (6)$$

Where τ represents the joint torque, M(q) represents the inertia matrix, C(q,q̇) represents the Coriolis-centrifugal force term,and G(q) represents the gravit term.The obtained torque is further substituted into the forward dynamics module to solve the actual trajectory. This module is modified based on the forward kinematics module built in the software, with input ports for the initial joint angle q_0 and initial joint angular velocity q_{d0} added, and the output structure modified to make it adapt to the overall structure of the entire simulation model. It should be particularly noted that since both the built-in toolbox and the robot toolbox developed by Peter Corke are used in this model, it is necessary to define both the DH parameters and the Rigid Body Tree parameters of the manipulator. The values of these two sets of parameters are the same, and only the parameter writing format needs to be changed.

2.3. PID auto-tuning based on GA

The key to realizing PID control lies in the selection of the three parameters K_p, K_i and K_d. Usually, the parameter tuning process is relatively complex and cumbersome, but the use of machine learning methods can realize automatic parameter tuning, greatly enhancing the environmental adaptability of the manipulator. In this study, the genetic algorithm is used for PID parameter optimization. The built-in manual initial PID parameters are set as K_p= 800, K_i= 50, K_d = 100, along with the upper and lower limits of K_p, K_i and K_d. The population size, maximum number of iterations, and display mode are set to draw the optimal fitness curve. The genetic algorithm function is called, and the fitness function is configured to optimize the three parameters within the established parameter range, and finally, output the optimized parameter values. Fitness is an important indicator of the genetic algorithm. A reasonable definition of fitness can effectively improve the efficiency of the algorithm and the quality of the results. The input of the fitness function used in this study is the PID parameter vector to be optimized, which includes K_p, K_i, and K_d, and the output is the fitness value. A smaller fitness value indicates better parameter performance. First, the PID parameters are parsed, and then the workspace parameters are updated, assigning K_p, K_i and K_d to the workspace variables. Next, the simulation program is run, and an attempt is made to call the model for simulation. If the simulation fails (e.g., model error or parameter out of bounds), the fitness is set to 1010as a penalty value, and this value is returned to end the function. Then, the error data is verified to check whether the error data variable exists in the workspace. If it does not exist, the fitness is set to 1010 as a penalty value, and this value is returned to end the function. After that, the fitness is calculated: the error value calculated by the model is read from the workspace, and one thousand times the sum of its squares is taken as the fitness. The purpose of multiplying by one thousand is to amplify the impact of fitness changes, making the algorithm sensitive to small changes in fitness.

Due to the randomness of the population generation process, some parameters may cause abnormal parameters, such as singular values, during model calculation, leading to simulation errors. If such abnormal functions continue to participate in the subsequent crossover process and are passed on to the next generation, it will cause a chain reaction and lead to the abnormal termination of the entire simulation. To this end, a special penalty term is defined in the fitness function. If abnormal parameters appear, the algorithm will assign a very large penalty value to this chromosome, so that it is eliminated in a timely manner, avoiding the occurrence of errors and improving the stability of the model. In this study, the upper and lower limits of the parameters selected for the genetic algorithm function are [1500, 150, 350] and [100, 10, 50], respectively. This range is derived based on a good value $[K_p, K_i, K_d]$ = [800, 50, 100] obtained from the initial manual parameter tuning, with approximately the same floating values taken above and below this value. To prevent the genetic algorithm from falling into a local optimum, the parameter range set for the algorithm is very large. This provides a large parameter exploration space for the GA function and improves the diversity of parameters in the early stage of iteration.

3. Results

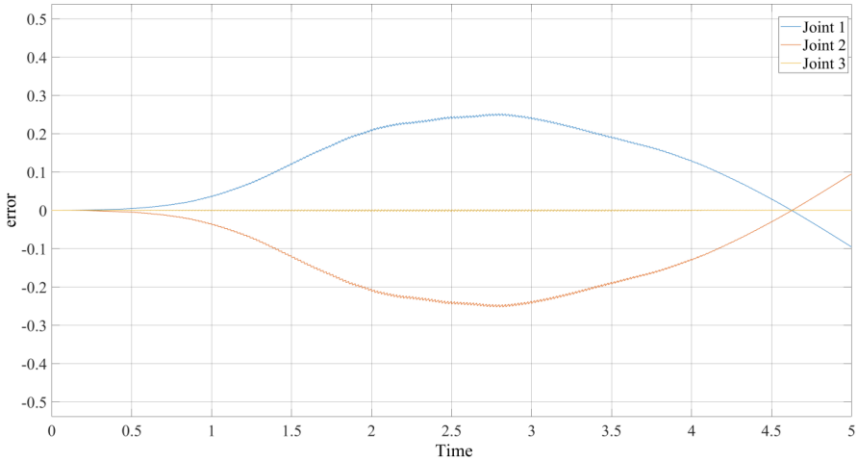

Fig. 6 Error curve obtained by manual parameter tuning (Picture credit: Original)

Fig. 6 shows the results of manual parameter tuning. It can be seen that although the error converges to a small value when the task is about to be completed, the error peak in the middle of the task is relatively large. In actual work, such a large error may exceed the allowable error of the actual design of the manipulator, reduce the service life of the manipulator, and even damage the drive motor or mechanical structure. Therefore, it is necessary to use the genetic algorithm to find a better solution. By trying different combinations of genetic algorithm parameters and comparing and analyzing the characteristics of the parameter change curve each time, the parameter options with a Gaussian mutation probability of 0.45, a crossover probability of 0.7, and an elite retention number of 0 are finally determined, and the PID parameter change curve as shown in Fig. 7 is formed.

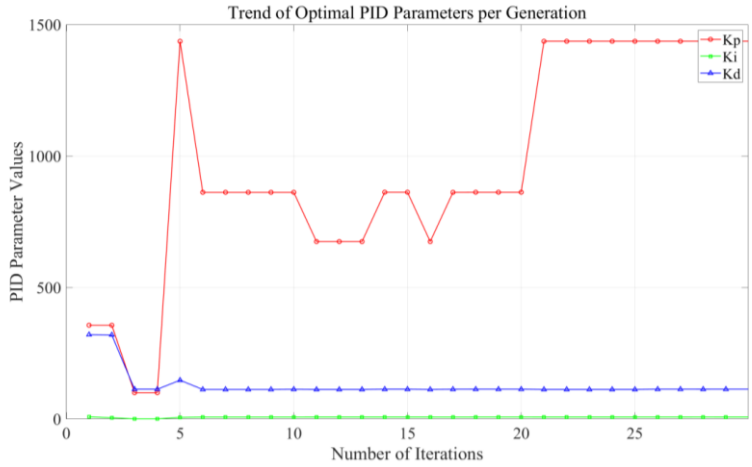

Fig. 7 Trend of optimal PID parameters per generation (Picture credit: Original)

It can be seen from Fig. 7 that the PID parameters fluctuate significantly in the early stage of iteration, indicating that the algorithm is actively trying different parameter combinations. In the middle stage of iteration, due to the continuous inheritance of excellent parameters, the amplitude of the curve becomes smaller. However, such stability can easily reduce the algorithm's enthusiasm for exploring excellent parameters, thereby causing it to fall into a local optimum. Therefore, an additional algorithm is added to intervene actively when the optimal parameters of multiple generations are the same, randomly disturbing some parameters and forcing the GA function to continue exploring, resulting in the parameter fluctuation in the middle section of Fig. 7. In the latter half of the simulation, the parameters tend to be stable, indicating that the algorithm that jumps out of the local optimum has locked the optimal solution of this simulation, which is K_p = 1436.5201, K_i = 7.6215, K_d= 113.08. The change curve of the optimal fitness per generation corresponding to the iteration process is shown in Fig. 7.

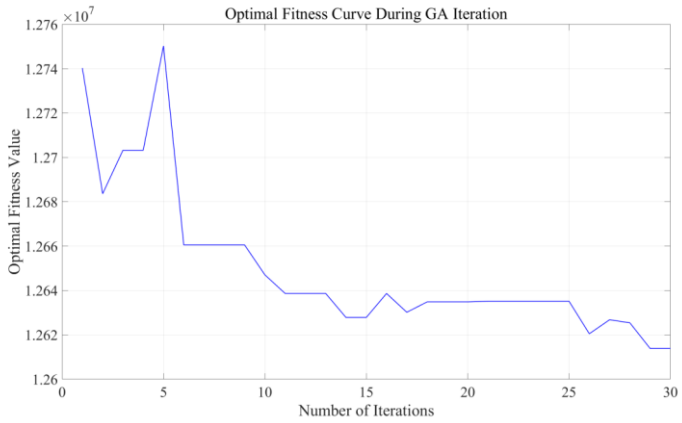

Fig. 8 Optimal fitness curve during GA Iteration (Picture credit: Original)

It can be seen from Fig. 8 that the fluctuation law of the fitness curve is basically consistent with the change law of the PID parameters in Fig. 7. It changes significantly at the beginning, fluctuates slightly, slows down in the middle, and shows a stable trend at the end of the 30-generation simulation. This further confirms that the GA algorithm has indeed jumped out of the local optimum and effectively selected scientifically reasonable, high-quality parameters during the entire simulation cycle. The error curve obtained by running these parameters in the model is shown in Fig. 9.

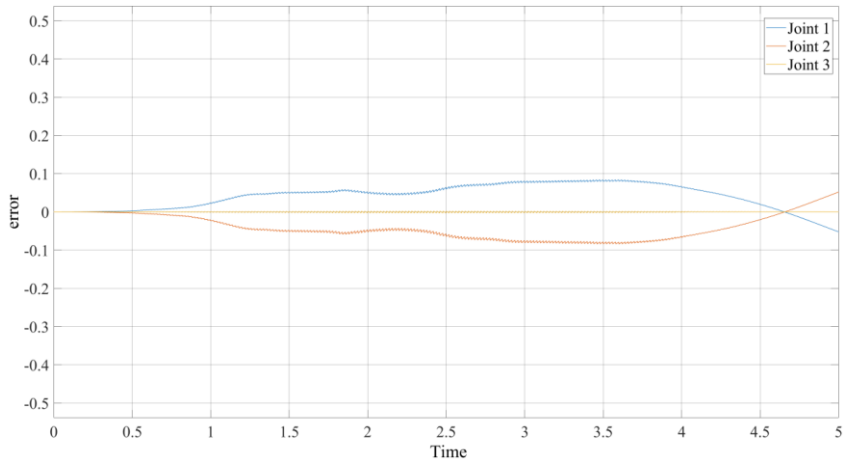

Fig. 9 Error variation curve optimized by GA (Picture credit: Original)

It can be seen from Fig. 9 that the optimized parameters enable the manipulator to reach the target position accurately and quickly within 5 seconds with an error peak maintained within ±0.1. Among them, Joint 3, which directly corresponds to the end effector, maintains an error close to 0, and the other two joints can also move smoothly to the corresponding positions, basically according to the planned path, ensuring the rationality and safety in actual work and realizing the research goal of solving traditional problems using new algorithms.

4. Conclusion

In this study, first, the first three joints of the PUMA560 manipulator are selected, and the rationality of the manipulator selection is verified by analyzing its velocity ellipsoid. Then, a PID control model is constructed in the simulation environment, and a series of manipulator parameter definitions and genetic algorithm codes are built to effectively optimize the trajectory accuracy of the manipulator using machine learning methods. Compared with the manual parameter tuning results before optimization, the PID parameters optimized by the GA algorithm have a smaller error fluctuation range, and the absolute value of the error is always stably lower than 0.1. Moreover, the error tends to be stable at the end of the 5-second simulation, achieving a relatively ideal optimization effect.

From the results, machine learning methods have indeed improved the efficiency of PID parameter tuning for manipulators and effectively enhanced the rotation accuracy of manipulators. They are efficient and scientific technical means, which are conducive to manipulators adapting to different work requirements in a timely manner. Although this study considers the rationality of the motor output and mechanical structure changes during the actual operation of the manipulator, it does not conduct specific quantitative analysis on this factor. In future research, more discussions on motor control and mechanisms should be added, and the quantitative standards for the quality of relevant data should be improved.

In general, this study focuses on the latest machine learning technologies, realizes the effective combination of emerging technologies and traditional algorithms, and provides inspiring ideas and reference value for the optimization of the environmental adaptability of manipulators.

References

[1] Gallala A, Kumar A A, Hichri B, Plapper P. Digital twin for human-robot interactions by means of Industry 4.0 enabling technologies. Sensors, 2022, 22(13): 4950.

[2] Shi X, Guo Y, Chen X, Chen Z, Yang Z. Kinematics and singularity analysis of a 7-DOF redundant manipulator. Sensors, 2021, 21(21): 7257.

[3] Azimirad V, Khodkam S Y, Bolouri A. A new hybrid learning control system for robots based on spiking neural networks. Neural Networks, 2024, 180: 106656.

[4] Jasim M M, Oleiwi B K, Azar A T, Mahlous A R. Hybrid controller with neural network PID/FOPID operations for a two-link rigid robot manipulator based on the zebra optimization algorithm. Frontiers in Robotics and AI, 2024, 11: 1386968.

[5] Mawloud T, Abderrezak A, Zinelabidine D M, Madjid K. Optimized GA-based PID control design for stick-slip suppression in drill string of rotary drilling system. Lecture Notes in Networks and Systems, 2025, 1238: 453–463.

[6] Zhu X, Guan J, Liu Z, Li Y, Yang R. Fuzzy PID control of single chamber semi-active hydro-pneumatic suspension based on GA optimization. Chinese Control Conference, 2024: 792–797.

[7] Katoch S, Chauhan S S, Kumar V. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 2021, 80(5): 8091–8126.

[8] Joseph S B, Dada E G, Abidemi A, Oyewola D O, Khammas B M. Metaheuristic algorithms for PID controller parameters tuning: review, approaches and open problems. Heliyon, 2022, 8(5): e09399.

[9] Fan Y, Shao J, Sun G. Optimized PID controller based on beetle antennae search algorithm for electro-hydraulic position servo control system. Sensors, 2019, 19(12): 2727.

[10] Günther J, Reichensdörfer E, Pilarski P M, Diepold K. Interpretable PID parameter tuning for control engineering using general dynamic neural networks: an extensive comparison. PLOS ONE, 2020, 15(12): e0243320.

Conference Review Form

Should this paper be considered for publication in the conference proceedings?

Yes no changes	√	**Yes with minor revisions**		**Yes with major revisions**		**No**	

Please expand on any weak areas in the checklist and offer specific advice as to how the author(s) may improve the paper.

This paper takes the PUMA560 robotic arm as the research object, constructs a three-degree-of-freedom system, and conducts a relatively systematic study from kinematic modeling, trajectory planning to control strategy optimization. The authors establish a kinematic model based on the DH method and analyze the end-effector motion characteristics using a velocity ellipsoid. They then compare two types of trajectory planning methods: joint space and Cartesian space, and finally use JTRAJ to achieve smoother path generation. In the control stage, the paper uses Simulink to construct a PID control model and optimizes parameters using a genetic algorithm, resulting in smaller error fluctuations and more stable convergence, significantly improving control robustness and trajectory accuracy. The overall research approach is clear, and the simulation verification is sufficient, demonstrating the effectiveness of combining traditional control methods with intelligent optimization algorithms. It has certain reference value for the research of adaptive control of robotic arms. The paper is of high quality with no obvious flaws and is recommended for direct acceptance.