Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **69**, *3*, pp. 335–340, Bucarest, 2024

Automatique et ordinateurs
Automatic Control and Computer Sciences

PARALLEL PLATFORM CONTROLLER BASED ON ADAPTIVE DIFFERENCE ALGORITHM – PART 2

RUIYANG WANG[1], QIUXIANG GU[1], SIYU LU[1], JIAWEI TIAN[1], ZHENGTONG YIN[2,*], XIAOLU LI[3], XIAOBING CHEN[4], LIRONG YIN[5], WENFENG ZHENG[1,*]

Keywords: Workspace control; Model predictive controller (MPC); Adaptive difference algorithm; Parallel platform control.

There are two main approaches to motion control on parallel platforms: joint space control and workspace control. Joint space control is an easy-to-implement semi-closed-loop strategy, but its control effect could be better. The workspace control is to obtain the real-time position of the parallel platform through the forward solution and close the speed and position loop of the parallel platform in the workspace. This paper uses a Model Predictive Controller (MPC) to control the parallel platform with workspace control as the research goal. The loss function is constructed based on the swarm intelligence optimization idea, and the adaptive difference algorithm is used to optimize the parameters of MPC. This part uses MATLAB to perform simulation experiments to complete the S-shaped velocity trajectory planning algorithm. In addition, the control effect of MPC and position-loop PI controller in a robust disturbance environment is compared. Experiments show that MPC has the advantages of low energy consumption and high control accuracy.

1. INTRODUCTION

In the first part, we deeply study the related methods of the existing parallel platform control and design a model predictive controller (MPC) to complete the parallel platform control. We start by modeling the state space of the controlled parallel platform from modern control theory. Then, the MPC of this six-degree of freedom (6-DOF) parallel platform is built based on this. After the model was established, we combined the swarm intelligence optimization idea to construct the loss function and used the adaptive difference (ADE) algorithm to optimize the parameters of MPC. Finally, we implemented the model on the upper computer in C++ and carried out a physical test. We obtained satisfactory test results, which verified the model's good performance in practical applications.

Based on Part 1, this part will expand our research and turn the focus to MATLAB simulation. By introducing a simulation environment, we will comprehensively evaluate and compare the proposed MPC model with a conventional PI controller. This comparison will help deepen our understanding of the performance benefits of MPC on parallel platforms and provide a solid foundation for further research. In this section, we discuss the design of the simulation experiment and the corresponding result analysis in detail to provide strong support for improving the control efficiency of the parallel platform.

2. ALGORITHM SIMULATION AND RESULTS

2.1 BENCHMARK FUNCTION SELECTION

MATLAB is used to complete the simulation experiment to verify the theory of the MPC control algorithm and complete the optimal selection of MPC parameters. The benchmark function expressions and their optimal values used in this experiment are shown in Table; the Sphere, Rosenbrock, Schwefel's 2.22, and Schwefel's 1.2 functions are unimodal test functions. This type of function has only a minimum value and is mainly used to verify the algorithm's development ability. Griewank and Rastrigin functions are multimodal functions with multiple minimum values, which can be used to test the algorithm's exploration ability. The test benchmark functions can fully verify the convergence, convergence speed, and global optimization capabilities of the differential evolution (DE) algorithm.

Table 1
Test benchmark functions and their optimal values

Benchmark function	Benchmark function equation	Best fitness
Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	0
Rosenbrock	$f(x)=\sum_{i=1}^{n-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	0
Griewank	$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	0
Rastrigin	$f(x)=10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	0
Schwefel's 2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	0
Schwefel's 1.2	$f(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_i\right)^2$	0

2.2 DIFFERENTIAL EVOLUTION ALGORITHM TEST

The population must first be initialized to test the benchmark function using the ADE. Set the individual dimension of the population to 6, the population iterative evolution is ten times, and the individuals in the population are initialized in a uniform distribution. First, the Schwefel 1.2 benchmark function in Table 1 is used to test the impact of population size, variation scaling factor, and crossover rate on algorithm performance. When the population size is 30, 100, and 300, respectively, and the test variation scaling

[1] School of Automation, University of Electronic Science and Technology of China, Chengdu 610054 China. E-mails: Ruiyang.wang@std.uestc.edu.cn, guqiuxiang@alu.uestc.edu.cn, siyu.lu@std.uestc.edu.cn, jravis.tian@std.uestc.edu.cn, (*) winfirms@uestc.edu.cn
[2] College of Resource and Environment Engineering, Guizhou University, Guiyang 550025, China. E-mail: ztyin@gzu.edu.cn
[3] School of Geographical Sciences, Southwest University, Chongqing, 400715, China. E-mail: xliswu@swu.edu.cn
[4] Division of Electrical and Computer Engineering, Louisiana State University, Baton Rouge 70803 LA, USA. E-mail: xchen87@lsu.edu
[5] Department of Geography and Anthropology, Louisiana State University, Baton Rouge 70803 LA, USA. E-mail: (*) yin.lyra@gmail.com

DOI: 10.59277/RRST-EE.2024.69.3.14

factor and crossover rate are 0.1, 0.5, and 0.9, this paper calculates the average fitness of the DE algorithm to the test benchmark function. The average fitness is obtained by taking 10 consecutive trials and arithmetically averaging the results. The impact of the parameters of the DE algorithm on the average fitness is shown in Table 2.

Table 2

Effect of differential evolution algorithm parameters on results

Population size	Variation scaling factor	Crossover rate	Average fitness
30	0.1	0.1	29.6212
	0.5	0.5	8.1958
	0.9	0.9	21.6096
100	0.1	0.1	14.3354
	0.5	0.5	1.3259
	0.9	0.9	10.3520
300	0.1	0.1	6.5354
	0.5	0.5	0.5868
	0.9	0.9	3.2705

It can be seen from Table 2 that under the same variation scaling factor and crossover rate, as the number of individuals in the population increases, the average fitness obtained by the DE algorithm has been significantly improved, which is closer to the theoretical value. However, as the number of populations continues to increase, the convergence speed of the average fitness of the algorithm shows a slowing down trend. In the case of the same population size, too small or too large variation scaling factor and crossover rate cannot make the individuals in the population achieve a better convergence effect. In the whole experiment, when the population size is set to 300, the variation scaling factor and crossover rate are set to 0.5. The minimum average fitness is 0.5868, closest to the theoretical value of 0. Therefore, if the platform's computing power allows the number of individuals in the population to be as large as possible, the variation scaling factor and crossover rate need to be chosen for better experimental values.

To verify the versatility of the DE algorithm, the DE algorithm is used to test all the test benchmark functions in Table 1. The number of individuals in the initialization population is 300, the individual dimension is 6, and the variation scaling factor and crossover rate are set to 0.5. For the calculation results, the method of carrying out ten experiments and taking the average is also adopted, and the obtained test results are shown in Table 3.

Table 3

Differential evolution algorithm benchmark function test results

Benchmark function	Best fitness	Average fitness
Sphere	0	0.0307
Rosenbrock	0	9.8143
Griewank	0	0.0614
Rastrigin	0	5.1943
Schwefel’s 2.22	0	0.1941
Schwefel’s 1.2	0	0.6562

It can be found from Table 3 that the optimization results of the DE algorithm for the two benchmark functions of Rosenbrock and Rastrigin are not ideal. The resulting average fitness differs considerably from the optimal fitness.

The convergence speed images of each benchmark function are shown in Fig. 1.

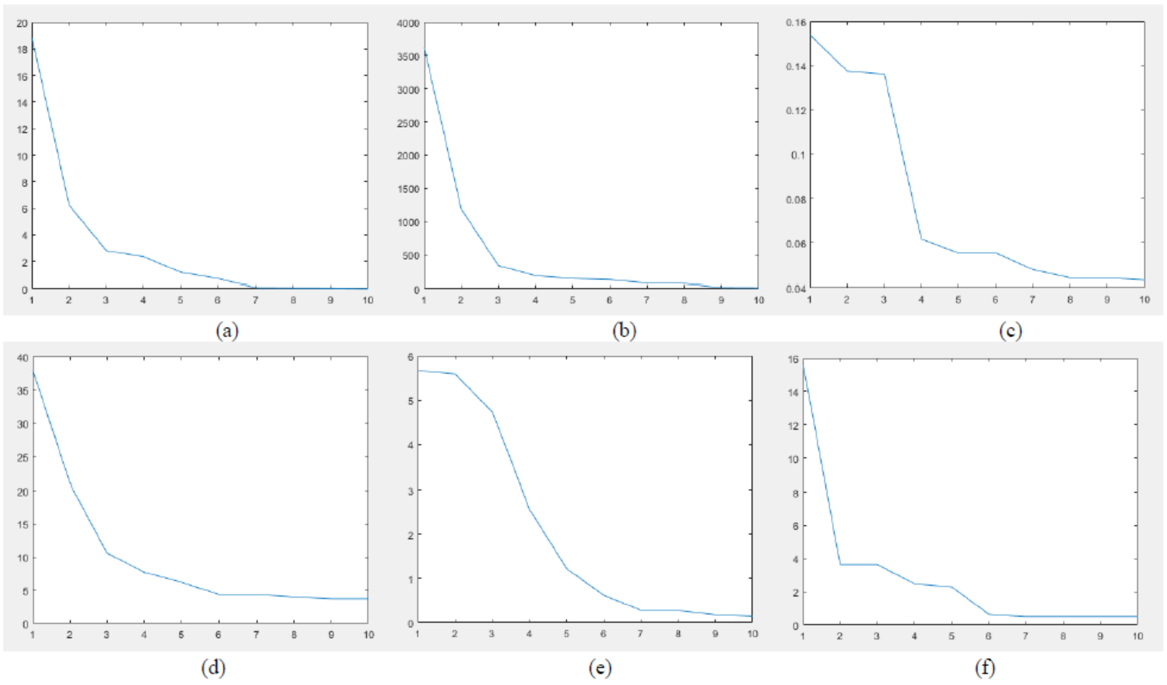

Fig. 1 – Convergence speed of differential evolution algorithm in benchmark function: a) sphere function; b) Rosenbrock function; c) Griewank function; d) Rastrigin function; e) Schwefel’s 2.22 function; f) Schwefel’s 1.2 function.

Comparing the convergence images of each function in Fig. 1, the fitness search range of the two functions in Figs. 1b and 1d is larger than that of other functions. After 10 iterations of DE, the fitness of the Rosenbrock function [1] dropped from 3 500 to 9.81.

The Rastrigin function [2] is also optimized from the initial 35 to 5.19, which has a good optimization effect. Figure 1 shows that the DE algorithm [3] is highly versatile and can be used to find the optimal solution for various nonlinear problems.

In addition, it can also be seen from Fig. 1 that the DE algorithm converges faster at the initial stage of iteration. Still, as the number of iterations increases, the improvement of the fitness of new individuals needs to be made apparent.

1.1. ADAPTIVE DIFFERENTIAL EVOLUTION (ADE) TESTING

The following conclusions can be drawn by analyzing the variation strategy of the ADE in

$$x_i' = x_{\text{best}} + F(x_{r1} - x_{r2}). \tag{1}$$

There are large differences between individuals in the initial stage of the iteration. At this time, the mutation operation will cause a large amount of individual movement, and finding the global optimal point near the individual is challenging. Therefore, this paper considers the linear correlation between the differential variation factor and the number of iterations to form an ADE algorithm [4–7]. The differential variation factor is given by:

$$F = \frac{I_{\text{cnt}}}{2 \times I_{\text{num}}}. \tag{2}$$

Among them, I_{cnt} is the current iteration number, and I_{num} is the total iteration number.

It can be seen from eq. (2) that the variation scaling factor at the initial stage of the algorithm is small, which can enable uniformly distributed individuals to detect the minimum points around the initial value fully. When the number of iterations increases gradually, the scaling factor is increased accordingly to accelerate the convergence of the population. ADE algorithm [8] and DE algorithm [9] are used to test the benchmark functions of Table 1, and the test results are shown in Table 4, and the convergence images of each test benchmark function corresponding to ADE are shown in Fig. 2.

Table 4

ADE and DE benchmark function test comparison results

Benchmark function	Best fitness	DE Average fitness	ADE Average fitness
Sphere	0	0.0307	0.0022
Rosenbrock	0	9.8143	3.2575
Griewank	0	0.0614	0.0375
Rastrigin	0	5.1943	4.7951
Schwefel's 2.22	0	0.1941	0.0575
Schwefel's 1.2	0	0.6562	0.0802

From the comparison of the results of the ADE and DE algorithms in Table 4, the search performance of the ADE algorithm is significantly better than that of the DE algorithm. It can be seen from Figure 2 that although the convergence speed of ADE is not fast at the initial stage of algorithm iteration, with the increase of the number of iterations, the differential variation factor is also increasing so that the population individuals in the late iteration can still have a certain development ability.

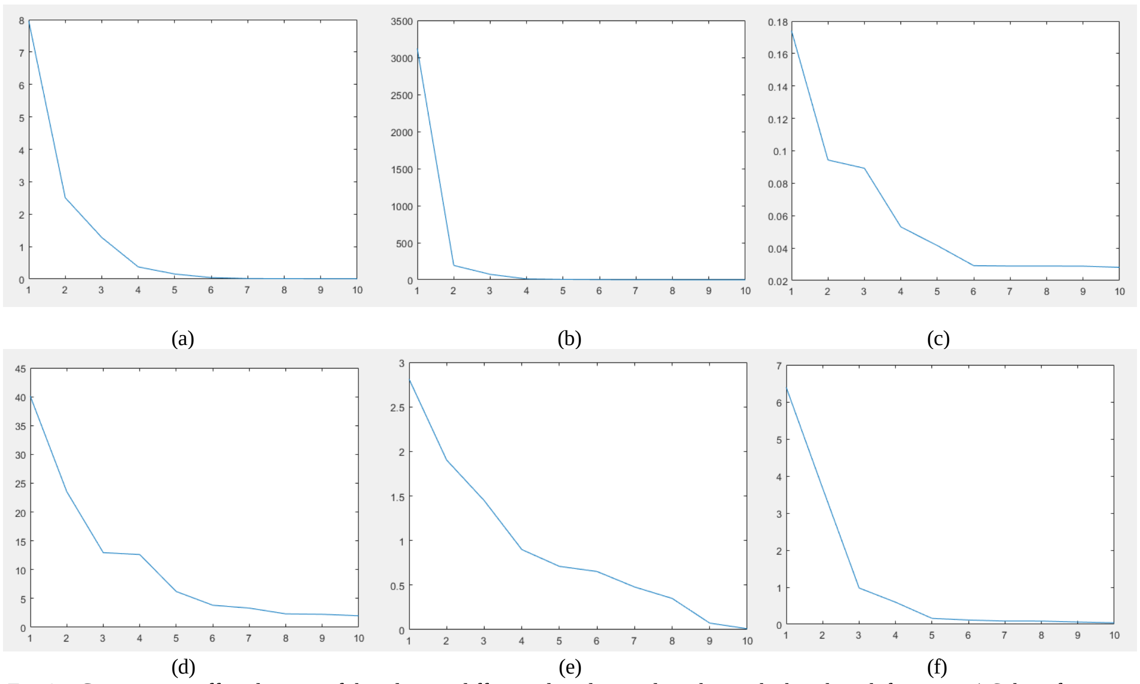

Fig. 2 – Convergence effect diagram of the adaptive differential evolution algorithm in the benchmark function: a) Sphere function; b) Rosenbrock function; c) Griewank function; d) Rastrigin function; e) Schwefel's 2.22 function; f) Schwefel's 1.2 function.

1.2. PERFORMANCE TEST OF MPC CONTROLLER

Functional modules in the MATLAB simulation platform are required to test the performance and improvement of the MPC controller's parameters. The simulation system's three main modules are the reference trajectory setting, motion controller, and controlled object.

The S-type velocity trajectory planning algorithm [10,11] sets the reference trajectory. This algorithm produces a trajectory image with continuous acceleration. The control function $\mathbf{U}(k)$ of the system is trapezoidal, and the reference acceleration is obtained through the S-type velocity trajectory planning algorithm. The jerk is set to 4 mm/s^3, and after 1 second of jerk, the reference control action reaches a maximum value of 4 mm/s^2. The system goes through a sequence of acceleration, uniform acceleration, deceleration, uniform velocity, acceleration and deceleration, uniform deceleration, and deceleration, in that order, within 1-second intervals, to complete the S-shaped velocity trajectory planning.

Factors such as friction and unmodeled dynamics are not considered in the state space modeling of parallel platforms [12]. Therefore, bases reference control action $\mathbf{U}(k)$, Gaussian noise with a standard deviation of 2 mm/s^2 and fixed noise with a fixed value of 0.2 mm/s^2 are superimposed to simulate external disturbances and model mismatches. For the reference control effect after superimposed noise, the effect of using the open-loop control method to control the controlled object is shown in Fig. 3:

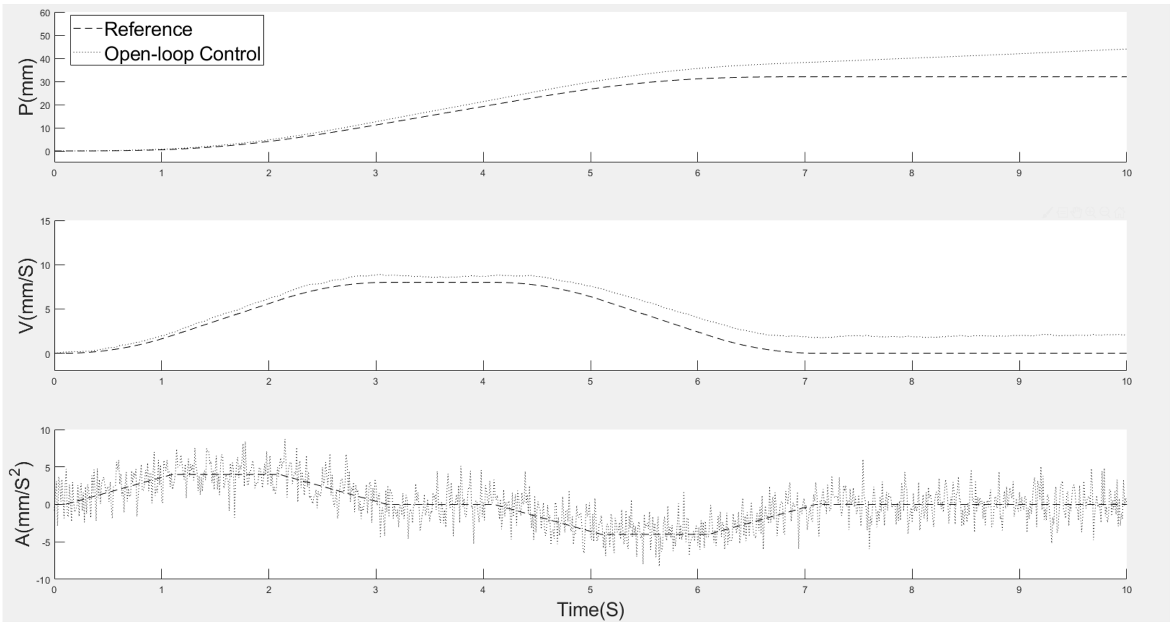

Fig. 3 – S-shaped speed trajectory open-loop control.

Figure 3 shows that after the acceleration of the controlled object is superimposed on the interference noise signal, the actual velocity and displacement have seriously deviated from the expected trajectory. The effect of open-loop control could be better, and a motion controller must be designed to control the controlled object. To apply the ADE algorithm in the process of MPC parameter optimization, the MPC is designed according to the control law,

$$\widehat{\mathbf{U}}(k) = -(\bar{\mathbf{B}}^{\mathrm{T}}\bar{\mathbf{Q}}\bar{\mathbf{B}} + \bar{\mathbf{R}})^{-1}\bar{\mathbf{B}}^{\mathrm{T}}\bar{\mathbf{Q}}(\bar{\mathbf{A}}\mathbf{X}(k) - \mathbf{Y}_r) \qquad (3)$$

The control effect was evaluated using the evaluation function of

$$F_{cost} =$$

$$\sum_{k=1}^{n}\{[\mathbf{X}(k) - \mathbf{Y}_r(\boldsymbol{k})]^{\mathrm{T}}[\mathbf{X}(k) - \mathbf{Y}_r(k)] + \mathbf{U}^{\mathrm{T}}(\boldsymbol{k})\mathbf{U}(k)\} + N^2. \ (4)$$

Each individual in the ADE algorithm corresponds to a set of parameters $[\mathbf{P} \quad \mathbf{Q} \quad \mathbf{R} \quad \mathbf{N}]^{\mathbf{T}}$ of the controller. Set the number of individuals in the population to 100, the crossover factor to 0.5, and the adaptive variation factor to be dynamically calculated by eq. (2).

The number of iterations is 10, and the discrete time step t_s is 0.01 seconds. After each mutation and crossover process, all individuals in the population are brought into Equation (3). This step tracks the reference trajectory and detects the system state of the controlled object according to the feedback loop. Use eq. (4) to evaluate the performance of MPC control parameters and keep the best ones. The optimal control parameters obtained after the algorithm is completed are shown in Table 5: the N^2 , which introduces the term eq. (4) has successfully reduced the prediction time domain and improved the real-time performance of the control system.

In addition to directly punishing the N^2 term in eq. (4), the $\mathbf{P}, \mathbf{Q}$, and $\mathbf{R}$ parameters are all indirectly punished through the control effect generated by the control effect of eq. (3). Equation (3) is obtained, assuming that the control action $\boldsymbol{U}$ is unconstrained. Therefore, the penalty R for energy consumption is small, while the penalty for steady-state error $\boldsymbol{P}$ and process error $\boldsymbol{Q}$ is large. In the physical system, the power supply mode is DC [13–15], which is not sensitive to energy consumption. Therefore, it is better to use a larger control amount to ensure better control performance without exceeding the parallel platform's power load and physical limit.

Table 5

MPC control parameters obtained by ADE algorithm

Control parameter	N	$\mathbf{P}$	$\mathbf{Q}$	$\mathbf{R}$
Parameter value	3	$\begin{bmatrix} 9.9736 & \\ & 7.1022 \end{bmatrix}$	$\begin{bmatrix} 5.0224 & \\ & 9.4630 \end{bmatrix}$	0.01

The parameter group $[\mathbf{P} \quad \mathbf{Q} \quad \mathbf{R} \quad \mathbf{N}]^{\mathrm{T}}$ obtained by ADE solution meets the requirements of the control system. In physical system deployment, limiting the control effect within a safe range is necessary.

The control law of eq. (3) is used to verify the MPC control performance and robustness. Combined with the control parameters in Table 5, the motion control of the model after adding noise in Fig. 3 is performed. A position loop PI controller was added as a control group to verify the superiority of MPC control performance. The effect of using MPC and PI controllers to control the noise trajectory is shown in Fig. 4.

The acceleration in Fig. 4 is the control action $\mathbf{U}(k)$. It can be seen from Fig. 4 that both the position loop PI controller and the MPC can complete the position servo tracking task very well. Further, observe the velocity tracking curve and the control action $\mathbf{U}(k)$. However, the PI controller does not consider the motion speed and energy consumption accordingly. Therefore, the speed curve and acceleration curve tracking effect could be better. In the case of the same interference noise, after model predictive control, the motion speed curve of the controlled object almost completely coincides with the reference speed curve. It shows that the MPC controller can complete the displacement and velocity tracking of the reference trajectory at the same time. In addition, the MPC controller $\mathbf{U}(k)$ deviation from the reference value is also smaller than that of the PI controller, indicating that the MPC has stronger energy control performance.

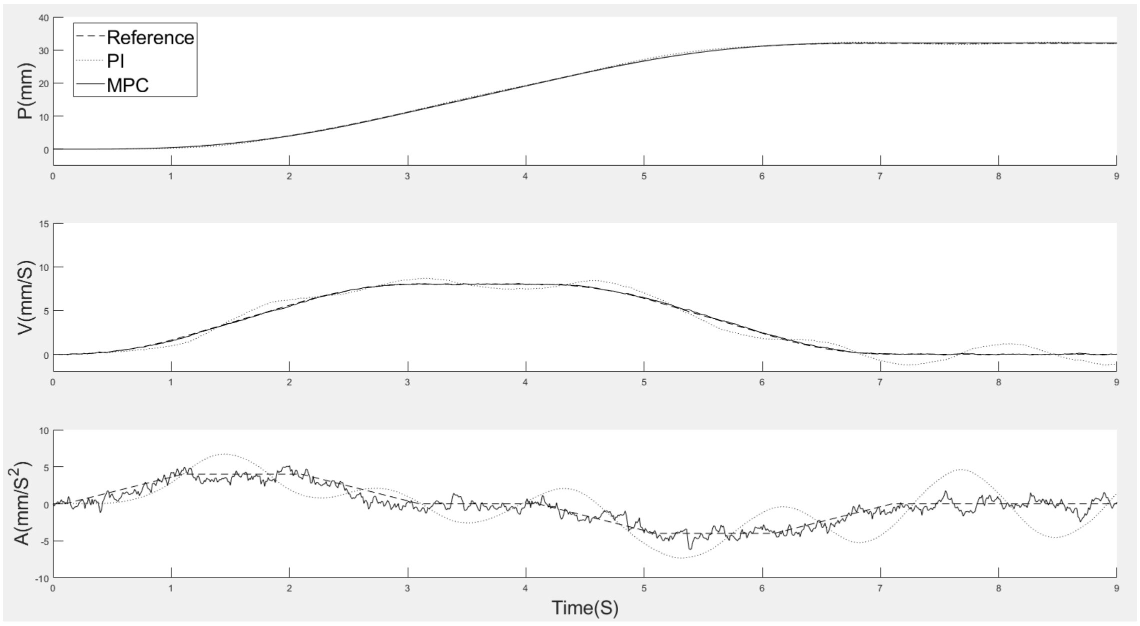

Fig. 4 – Comparison of MPC and PI control effects.

Then, the position servo tracking error of the PI controller and MPC will be compared. Draw the displacement error of the two algorithms during the entire movement process, as shown in Fig. 5.

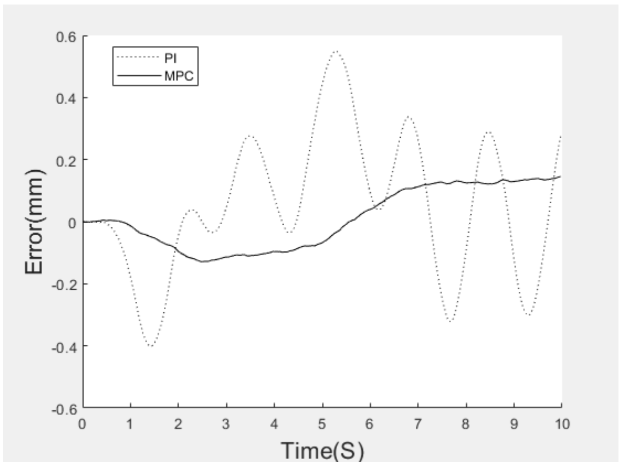

Fig. 5 – Error of PI controller and MPC with reference displacement.

It can be seen from Fig. 5 that the servo tracking accuracy of MPC is higher than that of the PI controller under the same noise environment. It can be concluded that the MPC algorithm has better control performance and stronger robustness than the PI controller.

3. **DISCUSSION AND CONCLUSION**

The controller constructed by the dynamic method has a better dynamic response, which is the key to further improving the control performance of the parallel platform [16-19]. In this paper, an MPC is first designed to control the 6-DOF parallel platform. Then, based on the swarm intelligence optimization idea, the loss function for optimizing the MPC parameters is constructed, and the ADE algorithm is used to optimize the MPC parameters. Then, the MPC algorithm is implemented on the upper computer of the control center by using C++ language. The physical objects of the parallel platform are controlled, and good control results are obtained. Finally, the S-shaped velocity trajectory planning algorithm was completed by simulation experiments on MATLAB, and the control effects of MPC and position loop PI controller in a robust disturbance environment were compared, verifying the MPC algorithm's superiority. In future research, we will test our platform in more diverse environments to explore control schemes for specific industrial application scenarios.

ACKNOWLEDGEMENTS

Supported by Sichuan Science and Technology Program (2021YFQ0003, 2023YFSY0026, 2023YFH0004).

AUTHOR CONTRIBUTIONS

Conceptualization, Qiuxiang Gu, XiaoBing Chen and Wenfeng Zheng; methodology, Jiawei Tian and Lirong Yin; software, Xiaolu Li, Qiuxiang Gu and Siyu Lu; formal analysis, Jiawei Tian, Xiaolu Li and Zhengtong Yin; writing-original draft preparation, Ruiyang Wang, Siyu Lu, Zhengtong Yin and Lirong Yin; writing-review and editing, Ruiyang Wang, Wenfeng Zheng and Lirong Yin; funding acquisition, Wenfeng Zheng. All authors have read and agreed to the published version of the manuscript.

Received on 4 March 2024

REFERENCES

1. J. Ma, H. Li, *Research on Rosenbrock function optimization problem based on improved differential evolution algorithm*, Journal of Computer and Communications, **7**, 107–120 (2019).
2. A. Omeradzic, H.-G. Beyer, in *Parallel Problem Solving from Nature*, 17th International Conference (PPSN), Springer, Dortmund, Germany, September 10–14, 2022, Proceedings, Part II. pp. 499–511 (2022).
3. M. Pant, H. Zaheer, L. Garcia-Hernandez, A. Abraham, *Differential*

Evolution: A review of more than two decades of research, Engineering Applications of Artificial Intelligence, **90**, 103479 (2020).
4. T.-C. Chiang, C.-N. Chen, Y.-C. Lin, *Parameter control mech in differential evolution: A tutorial review and taxonomy*, IEEE Symposium on Differential Evolution (SDE), pp. 1–8 (2013).
5. B. Zhang, X. Sun, S. Liu, X. Deng, *Tracking control of multiple unmanned aerial vehicles incorporating disturbance observer and model predictive approach*, Transactions of the Institute of Measurement and Control, **42**, *5*, pp. 951–964 (2020).
6. D. Adhikari, E. Kim, H. Reza, *A fuzzy adaptive differential evolution for multi-objective 3D UAV path optimization*, IEEE Congress on Evolutionary Computation *(CEC)*, Donostia, Spain, pp. 2258–2265 (2017).
7. X. Zhang, *Shift based adaptive differential evolution for PID controller designs using swarm intelligence algorithm*, Cluster Comput, **20**, pp. 291–299 (2017).
8. I. Farda, A. Thammano, *A self-adaptive differential evolution algorithm for solving optimization problems*, International Conference on Computing and Information Technolo*gy*, Cham: Springer International Publishing, pp. 68–76 (2022).
9. M.F. Ahmad, N.A.M. Isa, W.H. Lim, K.M. Ang, *Differential evolution: A recent review based on state-of-the-art works*, Alexandria Engineering Journal, **61**, 3831–3872 (2022).
10. M. Zhang et al., *An S-type ascent trajectory control method based on scramjet engine working boundary of RBCC*, *33rd* Chinese Control and Decision Conference (CCDC), pp. 5070–5073, IEEE, (2021).
11. M. Wang, J. Xiao, F. Zeng, G. Wang, *Research on optimized time-synchronous online trajectory generation method for a robot arm*, Robotics, and Autonomous Systems **126**, 103453 (2020).
12. X. Yang, H. Wu, B. Chen, S. Kang, S. Cheng, *Dynamic modeling and decoupled control of a flexible Stewart platform for vibration isolation*, Journal of Sound and Vibration, **439**, 398–412 (2019).
13. F. Amrane, A. Chaiba, B. Francois, *Improved adaptive nonlinear control for variable speed wind-turbine fed by direct matrix converter*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **68**, *1*, pp. 58–64 (2023).
14. O.D. Laudatu, M. Iordache, *Comparison of inductive and capacitive couplings used to close the feedback loop used in switch mode power supplies*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **68**, *4*, pp. 363–368 (2023).
15. A.C. Ghoerghe, H. Andrei, E. Diaconu, G. Seritan, B. Enache, *Système intelligent pour la réduction de la consommation électrique en veille des équipements ménagers*, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., **68**, *4*, pp. 413–418 (2023).
16. C. Chen, H. Pham, *Trajectory planning in parallel kinematic manipulators using a constrained multi-objective evolutionary algorithm*, Nonlinear Dynamics, **67**, *2*, pp. 1669–1681 (2011).
17. C.-T. Chen, T.-T. Liao, *A hybrid strategy for the time-and energy-efficient trajectory planning of parallel platform manipulators*, Robotics and Computer-Integrated Manufacturing, **27**, *1*, pp. 72–81 (2011).
18. J.R.G. Martínez, J.R. Reséndiz, M.Á.M. Prado, E.E.C. Miguel, *Assessment of jerk performance s-curve and trapezoidal velocity profiles*, XIII International Engineering Congress (CONIIN*)*, IEEE, pp. 1–7 (2017).
19. Y. Zuo, J. Mei, C. Jiang, X. Yuan, S. Xie, C.H. Lee, *Linear active disturbance rejection controllers for PMSM speed regulation system considering the speed filter*, IEEE Transactions on Power Electronics, **36**, *12*, pp. 14579–14592 (2021).