Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **69**, *2*, pp. 243–248, Bucarest, 2024

PARALLEL PLATFORM CONTROLLER BASED ON ADAPTIVE DIFFERENCE ALGORITHM – PART 1

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 details the research background and the algorithm design process. Then, the MPC algorithm is implemented on the upper computer using C++, and the physical test is implemented. The test results show that the controller has a good control effect on the physical platform.

1. INTRODUCTION

Six-degree-of-freedom (6-DOF) parallel robots are widely used in motion simulation, high-precision pose adjustment, and CNC machining systems [1–6]. Motion control includes trajectory planning, trajectory tracking, and other related content. Aiming at trajectory, some scholars have proposed a multi-objective constrained evolutionary algorithm [7] and a trajectory planning method based on time and energy optimization [8]. In addition, there are relatively simple trajectory planning methods such as spline interpolation, velocity trapezoidal curve, and velocity S-curve [9]. Trajectory tracking [10] mainly includes two technical routes of joint space control and workspace control, and each technical route has a variety of control strategies that can be used. Y. Zuo et al. proposed a linear active disturbance rejection controller for the PMSM speed control system considering the speed filter [11]. In addition, various control methods, such as high-order differential feedback control, fuzzy neural network sliding mode control, and L1 adaptive control [12–16], have been applied to the 6-DOF parallel platform control.

This paper presented a control model for a 6-DOF parallel robot. We first established the state-space equation for the motion of the parallel platform in the workspace based on modern control theory. At the same time, to reduce the influence of external noise on the control system, which leads to inaccurate modeling, this paper also designed a model predictive controller (MPC) to control the parallel platform. Then, based on the idea of swarm intelligence optimization, the loss function of the MPC parameter optimization was constructed. Then, the MPC algorithm was implemented on the upper computer of the control center using C++ language, and the physical test was completed. The physical test results show that the parallel platform achieves good real-time control results. Finally, the Adaptive Differential Evolution algorithm optimized the model's predictive control parameters. In addition, this paper also carried out simulation experiments on MATLAB to compare the MPC algorithm with the PI controller.

2. STATE SPACE MODELING

This paper established a control model for the controlled parallel platform according to modern control theory [17] to complete the design of the motion controller of the parallel platform. In this study, the motion of the parallel platform in the workspace is decomposed into two parts: the translation of the moving platform's center of mass and the rotation around the center of mass.

The time under physical conditions changes continuously, while the digital control system can only complete the motion control under the discrete time state. Therefore, when establishing the control model of the controlled object, it is first necessary to discretize its motion state. Assuming that the discrete time step is t_s according to Newton's second law when the center of mass of the parallel platform moves in translation at a uniform speed in the workspace, the relationship between the displacement, velocity, and acceleration of the center of mass of the platform is:

$$\begin{cases} S(k+1)=s(k)+v(k)t_s+\frac{1}{2}a(k)t_s^2, \\ v(k+1)=v(k)+a(k)t_s. \end{cases} \tag{1}$$

Let the system control function $\mathbf{U}(k)=a(k)$ state variable $\mathbf{X}(k)=[\mathrm{s}(k)\mathrm{v}(k)]^{\mathrm{T}}$. Writing eq. (1) as a state space expression yield

$$\begin{cases} \mathbf{X}(k+1)=\mathbf{A}\mathbf{X}(k)+\mathbf{B}\mathbf{U}(k), \\ \mathbf{Y}(k)=\mathbf{C}\mathbf{X}(k), \\ \mathbf{A}=\begin{bmatrix}1 & t_s\\ 0 & 1\end{bmatrix}, \\ \mathbf{B}=\begin{bmatrix}\frac{1}{2}t_s^2\\ t_s\end{bmatrix}, \\ \mathbf{C}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}. \end{cases} \tag{2}$$

Next, we need to determine the controllability of the above

[1] School of Automation, University of Electronic Science and Technology of China, Chengdu 610054 China.
E-mail: Ruiyang.wang@std.uestc.edu.cn, guqiuxiang@alu.uestc.edu.cn, jravis.tian@std.uestc.edu.cn, siyu.lu@std.uestc.edu.cn, winfirms@uestc.edu.cn (Corresponding Author)
[2] College of Resource and Environment Engineering, Guizhou University, Guiyang 550025, China.
E-mail: ztyin@gzu.edu.cn (Corresponding Author)
[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: Lyin5@lsu.edu

DOI: 10.59277/RRST-EE.2024.69.2.21

model. The controllability matrix of the parallel platform system is:

$$[\mathbf{B} \quad \mathbf{AB}]=\begin{bmatrix}\frac{1}{2}t_s^2 & \frac{3}{2}t_s^2\\ t_s & t_s\end{bmatrix}. \tag{3}$$

Equation (3) shows that the matrix is full rank when $t_s \neq 0$. Therefore, the parallel platform system is fully controllable.

According to the relevant conclusions of rigid body mechanics, the rotation motion of the parallel platform around the center of mass belongs to the uniform rotation of a rigid body around a fixed axis. It has the same mathematical expression as the translational motion of the center of mass. Therefore, the state-space model developed in terms of the uniform variable linear motion of the centroid of the parallel platform is also applicable to the uniform variable rotation around the centroid.

3. DESIGN OF MOTION CONTROLLER FOR PARALLEL PLATFORM

The 6-DOF platform used in this study is shown in Fig. 1.

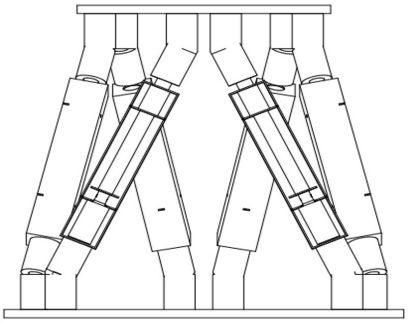

Fig. 1 – Schematic diagram of a six-degree-of-freedom parallel platform.

For the 6-DOF parallel platform, when modeling the state space of the parallel platform, factors such as the friction force of the joint joints of each manipulator are not modeled. Therefore, when designing the controller, it is necessary to adopt a more robust control method to make the system achieve a better control effect. MPC [18,19] is a special kind of control. Its current control action is obtained at each sampling instant by solving a finite-time domain open-loop optimal control problem. The current state of the process is used as the initial state of the optimal control problem, and only the first control action is implemented in the solved optimal control sequence. It adopts the model of non-minimization description, and the method is robust and stable. Therefore, the MPC method is selected to control the motion of the parallel platform in this system [19].

There are many methods for parameter optimization [20], among which Differential Evolution (DE) and its improved algorithms are often used for parameter optimization of MPC [21,22]. Adaptive Differential Evolution (ADE) [23,24] is an improved DE algorithm that uses an adaptive strategy to adjust the parameters of the differential evolution algorithm to improve its performance. Due to its good optimization effect, ADE is selected to complete the parameter optimization of MPC in this paper [25].

3.1 PARALLEL PLATFORM MODEL PREDICTIVE CONTROL

The main process of motion control using the MPC control method consists of three parts: prediction model, rolling optimization, and feedback correction. The block diagram of the MPC control structure is shown in Fig. 2.

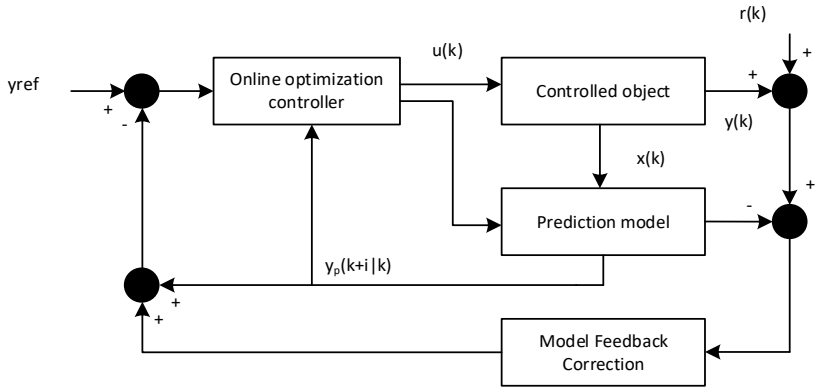

Fig. 2 – MPC control structure block diagram.

Predictive models can be based on system state space models, and the state $\mathbf{X}(k)$ of the system at time k, the current input $\mathbf{U}(k)$ and the calculated expected future input $\mathbf{U}(k+i-1|k)$ to predict the future output $y(k+i-1|k)$. $\mathbf{U}(k+i-1|k)$ is the control time domain, and $y(k+1|k)$ is the prediction time domain. Each time the control quantity is calculated, the lengths of the control and prediction time domains remain unchanged, and the time axis moves by one unit, forming a rolling calculation effect.

The online optimization controller sets the penalty function and optimization conditions. The penalty function must comprehensively consider the system output's process error, steady-state error, and energy consumption. By minimizing the penalty function, the optimal control effect can be obtained. Because external noise and other factors inevitably affect the controlled object, there will be a certain deviation between the actual and expected output. Therefore, the actual output of the controlled object needs to be measured and fed back. The feedback result is used as the parameter for calculating the optimal control quantity at the next moment to complete the control closed loop.

For the model predictive controller, it is assumed that both the length of the prediction time domain and the length of the control time domain are N. Let the output $\mathbf{Y}(k)$ be equal to the state $\mathbf{X}(k)$ and the reference output be $\mathbf{Y}_r(k)$. The output of step i is predicted to be $\mathbf{Y}(k+i/k)$ at time k. Then, the error between the predicted output and the reference output at the i-th moment is calculated by

$$\mathbf{E}(k+i|k)=\mathbf{Y}(k+i|k)-\mathbf{Y}_r(k+i)$$

$$=\mathbf{X}(k+i|k)-\mathbf{Y}_r(k+i). \quad (4)$$

The set penalty function in the online optimization controller includes process error, process energy consumption, and steady-state error. The expression is

$$J(\mathbf{U})=\sum_{i=0}^{N-1}[\mathbf{e}^{\mathrm{T}}(k+i|k)]\mathbf{Q}\mathbf{e}(k+i|k) + \mathbf{U}^{\mathrm{T}}(k+i|k)\mathbf{R}\mathbf{U}(k+i|k)+ \mathbf{e}^{\mathrm{T}}(k+N|k)\mathbf{P}\mathbf{e}(k+N|k). \quad (5)$$

Among them, **Q** is a positive semi-definite matrix with the same dimension as the state **X**, and it is a penalty coefficient for the cumulative error of the process. **R** is a symmetric positive definite matrix with the same dimension as **U**, and it is a penalty coefficient for the energy consumption of the process. **P** is a positive semi-definite matrix with the same dimension as the state ***X***, and it is a penalty coefficient for the steady-state error.

The design goal of the parallel platform motion controller is to solve the optimal control sequence $\mathbf{U}(k+i/k)$ so that the penalty function reaches the minimum value. According to the state, space eq. (2), the predicted state at time $k+i$ is expressed by

$$\mathbf{X}(k+i|k)=\mathbf{A}^{i}\mathbf{X}(k|k)+\sum_{j=1}^{i}\mathbf{A}^{i-j}\mathbf{B}\mathbf{U}(k+j-1|k). \quad (6)$$

Denote the predicted state sequence at time k as $\hat{\mathbf{X}}(k)=[\mathbf{X}(k|k)\ \ \boldsymbol{X}(k+1|k)\ \cdots\ \mathbf{X}(k+N|k)]^{\mathrm{T}}$, the reference output sequence as $\mathbf{Y}_r=[\mathbf{Y}_r(k)\ \ \mathbf{Y}_r(k+1)\ \cdots\ \mathbf{Y_r}(k+N)]^{\mathrm{T}}$, and the current and future control action sequence as $\hat{\mathbf{U}}(k)=[\mathbf{U}(k|k)\ \ \mathbf{U}(k+1|k)\ \cdots\ \mathbf{U}(k+N-1|k)]^{\mathrm{T}}$. The state-space equations for $\hat{\mathbf{X}}(k)$ and $\hat{\mathbf{U}}(k)$ are then expressed by,

$$\hat{\mathbf{X}}(k)=\begin{bmatrix}\mathbf{I}\\ \mathbf{A}\\ \vdots\\ \mathbf{A}^{\mathbf{N}}\end{bmatrix}\mathbf{X}(k)+\begin{bmatrix}\mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{B} & & & \\ \mathbf{AB} & \mathbf{B} & \ddots & \\ \vdots & & & \\ \mathbf{A}^{N-1}\mathbf{B} & \cdots & & \mathbf{B}\end{bmatrix}\hat{\mathbf{U}}(k). \quad (7)$$

Equation (7) is abbreviated as,

$$\hat{\mathbf{X}}(k)=\bar{\mathbf{A}}\mathbf{X}(k)+\bar{\mathbf{B}}\hat{\mathbf{U}}(k). \quad (8)$$

Substitute eq. (8) into eq. (5). Get the system penalty function expression,

$$J=\left[\bar{\mathbf{A}}\mathbf{X}(k)+\bar{\mathbf{B}}\hat{\mathbf{U}}(k)-\mathbf{Y}_r\right]^{\mathrm{T}}\bar{\mathbf{Q}}\left[\bar{\mathbf{A}}\mathbf{X}(k)+\bar{\mathbf{B}}\hat{\mathbf{U}}(k)-\mathbf{Y}_r\right]+ +\hat{\mathbf{U}}^{\mathrm{T}}(k)\bar{\mathbf{R}}\hat{\mathbf{U}}(k). \quad (9)$$

There are N **Q**s along the diagonal of $\bar{\mathbf{Q}}$ and the last one is **P**. There are N **R**s along the diagonal $\bar{\mathbf{R}}$. They are,

$$\begin{cases}\bar{\mathbf{Q}}=\begin{bmatrix}\mathbf{Q} & \cdots & \\ \vdots & \ddots & \vdots\\ & \cdots & \mathbf{P}\end{bmatrix},\\ \bar{\mathbf{R}}=\begin{bmatrix}\mathbf{R} & & \\ & \ddots & \\ & & \mathbf{R}\end{bmatrix}.\end{cases} \quad (10)$$

By solving the quadratic programming problem of eq. (9), the model control sequence $\hat{\mathbf{U}}(k)$ can be obtained. Under the unconstrained condition of model predictive control sequence $\hat{\mathbf{U}}(k)$. Equation (11) can be obtained by taking the derivative concerning $\hat{\mathbf{U}}(k)$:

$$\frac{\partial J}{\partial\hat{\mathbf{U}}(k)}=2\bar{\mathbf{B}}^{\mathrm{T}}\bar{\mathbf{Q}}\left[\bar{\mathbf{A}}\mathbf{X}(k)+\bar{\mathbf{B}}\hat{\mathbf{U}}(k)-\mathbf{Y}_r\right]+2\bar{\mathbf{R}}\hat{\mathbf{U}}(k). \quad (11)$$

Setting eq. (11) to zero, the minimum value of J under the action of the control sequence $\hat{\mathbf{U}}(k)$ can be obtained. In the model predictive controller, **A**, **B**, **P**, **Q** are semi-positive definite *matrices* and **R** is positive definite. So, $\bar{\mathbf{B}}^{\mathrm{T}}\bar{\mathbf{Q}}\bar{\mathbf{B}}+\bar{\mathbf{R}}$ is positive definite so that the control sequence can be expressed as

$$\hat{\mathbf{U}}(k)=-\left(\bar{\mathbf{B}}^{\mathrm{T}}\bar{\mathbf{Q}}\bar{\mathbf{B}}+\bar{\mathbf{R}}\right)^{-1}\bar{\mathbf{B}}^{\mathrm{T}}\bar{\mathbf{Q}}(\bar{\mathbf{A}}\mathbf{X}(k)-\mathbf{Y}_r). \quad (12)$$

3.2 OPTIMIZATION OF MPC CONTROL PARAMETERS BASED ON DIFFERENTIAL EVOLUTION ALGORITHM

In MPC, parameters such as penalty coefficients **P**, **Q**, **R** the forecast time domain N length needs to be set. To solve the problem that the manual method is tedious and difficult to achieve a better control effect, this paper established an evaluation function for MPC control parameter optimization. This function uses a real-coded differential evolution algorithm to perform a metaheuristic search in the parameter space to complete parameter optimization, which makes the MPC achieve a better control effect.

The DE algorithm uses the difference vectors of different individuals in the parameter space to make them move. Finally, a solution vector that better matches the evaluation function is found. The algorithm flow is shown in Fig. 3.

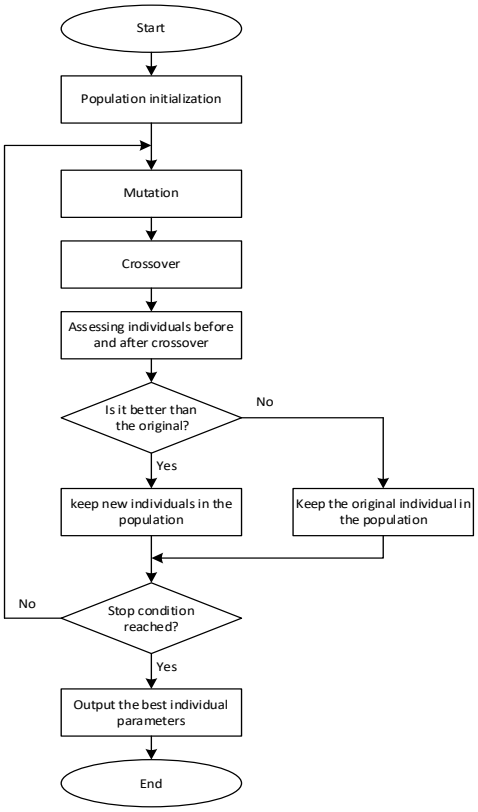

Fig. 3 – Flow chart of differential evolution algorithm.

In Fig. 3, population initialization needs to set the number and dimension of individuals. In metaheuristic search, selecting the number of individuals must consider the parameter optimization effect and computing time. This system is mainly used to optimize the selection of MPC parameters through the DE algorithm. The parameters the

MPC needs to determine are **P**, **Q**, **R**, and **N**. Among them, the dimension of penalty coefficient **P** and **Q** is 2, **R** is 1, and the dimension of time domain length N is 1. So, the dimension of a single individual in the population is 6. Initialization also needs to set the individual search range to meet the semi-positive definite or positive definite conditions of each parameter of MPC. N can only take positive integers. The initialized population individuals are randomly set in the parameter space in a uniform distribution manner. This distribution method can prevent the initial individuals from gathering in a specific area, enhance the exploration ability of the algorithm, and improve the optimization effect.

The mutation operation of the DE algorithm is fundamentally different from that of the genetic algorithm. In the DE algorithm, using different mutation algorithms will produce different optimization strategies. Record the *i-th* individual in the population as X_I, and use the DE/best/1 mutation strategy to mutate. The algorithm for generating new individuals is,

$$X_I^{'}=x_{\text{best}}+F(x_{\text{r1}}-x_{\text{r2}}). \quad (13)$$

Among them, x_{best} is the best individual in the population. X_{r1} and x_{r2} are two individuals randomly selected in the population. F is the variation scaling factor, and its value is a decimal between 0 and 1.

The crossover operation of the DE algorithm is realized by using the roulette wheel method. Generate a random number for each dimension of the individual. If the random number is greater than the crossover factor, the value of the corresponding dimension variation individual $x_I^{'}$ is retained, otherwise the value of the original individual x_I is retained.

The purpose of tuning the MPC control parameters is to achieve the optimal control effect with the minimum calculation cost and energy. For this, it is necessary to establish a loss function to objectively evaluate the performance of the original individuals in the population and the new individuals generated by the crossover operation. The model uses 4 parameters to be optimized to construct the parameter space, and the parameter vector is $\boldsymbol{p_i}=[\mathbf{P}\quad \mathbf{Q}\quad \mathbf{R}\quad \mathbf{N}]^{\mathrm{T}}$. The calculation amount of MPC is mainly affected by the time domain length N. The constructed loss function should include the time domain length N to improve the system's real-time performance. The loss function of constructing population individuals is

$$F_{\text{cost}}=\sum_{k=1}^{n}\{[\mathbf{X}(k)-\mathbf{Y}_r(k)]^{\mathrm{T}}[\mathbf{X}(k)-\mathbf{Y}_r(k)]+\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k)\}+N^2. \quad (14)$$

Among them, $\mathbf{X}(k)$ is the system state. $\mathbf{Y}_r(k)$ is the reference trajectory obtained by using the trajectory planning method. Both $\mathbf{X}(k)$ and $\mathbf{Y}_{\mathbf{r}}(k)$ have dimensions 2 and consist of displacement and velocity. Equation (14) consists of three parts: the cumulative error square value, the cumulative energy consumption square value, and the time domain length square value during the entire movement process from the start point to the endpoint. Therefore, the loss function in eq. (14) can comprehensively evaluate the individuals in the population from the three aspects of control performance, energy consumption, and calculation cost. In the optimal selection process of the DE algorithm, the original individuals of the population and the new individuals generated by the crossover are evaluated using eq. (14) respectively. The individual with a smaller F_{cost} value is kept in the population, and the other one is eliminated. After reaching the stopping condition, output the corresponding parameter value of the best individual. The output values are used as the optimal control parameters of the MPC.

4. PHYSICAL TEST

To verify the real-time control performance of the controller designed in this paper, we first use C++ language to implement the MPC algorithm in the upper computer and do a physical test to observe the control effect of the controller.

The MPC algorithm, the reference trajectory vector $\boldsymbol{Y}_{\mathbf{r}}$ involved in the calculation is composed of two parts: the reference displacement and the reference velocity. Therefore, the closed-loop control of the position cannot be independently performed like the PI controller. To implement the MPC algorithm in the upper computer of the parallel platform control center, two functional modules of trajectory planning, and motion control need to be completed. Among them, the trajectory planning is completed immediately when the UI receives the user target location information. Store the corresponding reference displacement and reference speed in the memory through an array. Motion control is done in a 100 mS timer. The running process of the upper computer is shown in Fig. 4:

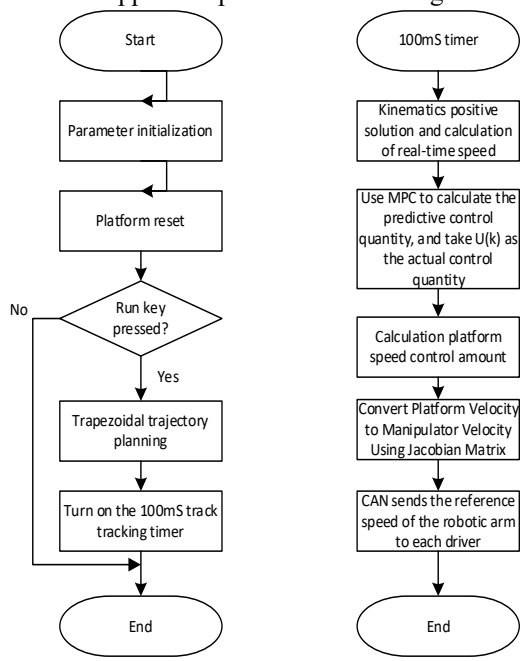

Fig. 4 – Flowchart of upper computer MPC algorithm.

As seen from Fig. 4, this progress must perform parameter initialization and reset the platform. Then, if the run key is pressed, the trapezoidal trajectory planning starts, and finally the 100 ms trajectory tracking timer is turned on. The position closed-loop period of the upper computer is 100 mS. In the 100 ms timer, the real-time pose of the parallel platform is first calculated using the MPR-NR kinematics forward solution algorithm. Then use the MPC control law to calculate the output $\mathbf{U}(k)$ of the MPC. Since the time domain length of the control system is N = 3, there are 3 calculated control quantities. However, only the control

quantity $\mathbf{U}(k)$ at this moment is taken out as the actual control quantity, and the forecast control quantity $\mathbf{U}(k+i)$ is not output. The calculated output ***U(k)*** is the acceleration control function of the parallel platform. Each servo driver can only complete the speed closed loop, so the conversion from acceleration to speed is completed by

$$\mathbf{V}r(k) = \mathbf{V}(k\text{-}1) + \mathbf{U}(k)*t/2 \tag{15}$$

where $t = 0.1$ seconds. The trapezoidal trajectory planning algorithm is a uniformly variable speed motion and eq. (15) calculates the average speed of the parallel platform between k-1 time and k-time. Therefore, acceleration control can be equivalent to speed control. Then, the acceleration control of the platform can be equivalent to speed control. It is then mapped to the reference velocity of each manipulator through the Jacobian matrix. Finally, the reference velocity is sent to each robot arm through the CAN bus to complete the control of the workspace of the parallel platform. The image of using the MPC algorithm to control the parallel platform is shown in Fig. 5.

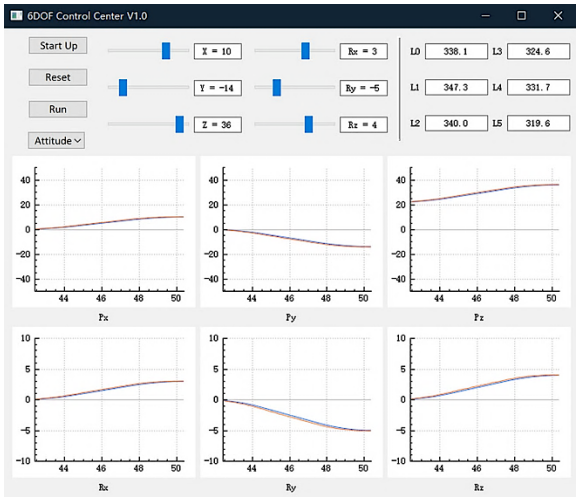

Fig. 5 –The target attitude parameters and servo tracking image of the host computer in the control center.

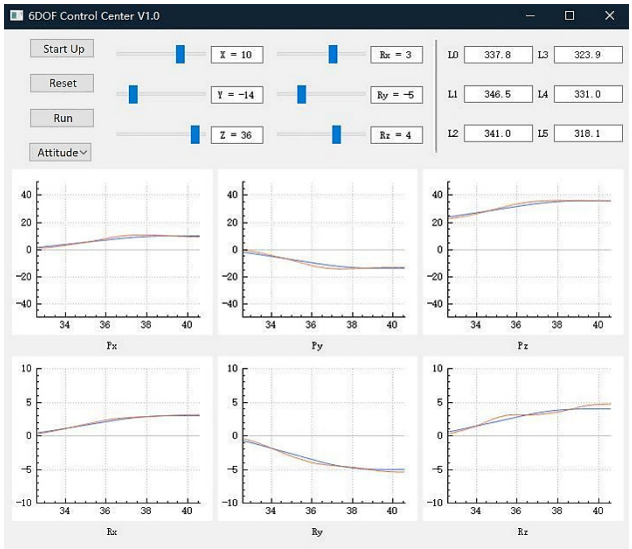

Fig. 6 – PI controller physical test image.

Figure 5 shows that the actual running attitude and the reference trajectory attitude completely coincide. MPC can complete the trajectory tracking task as expected. For the same attitude target, the attitude image using the PI controller for motion control is shown in Fig. 6.

It can be seen from Fig. 6 that the PI controller can also control the parallel platform to move in the direction of the expected trajectory. However, there is a certain error between the real-time trajectory image and the reference trajectory. During the entire control cycle of trajectory tracking, the two control methods are recorded separately, the absolute cumulative error between the actual pose of the parallel platform and the reference pose.

The obtained cumulative absolute error data of the two control methods are shown in Table 1.

Table 1

Cumulative absolute error in trajectory tracking period.

Control Method	$\sum\lvert e(\alpha)\rvert$	$\sum\lvert e(\beta)\rvert$	$\sum\lvert e(\gamma)\rvert$
PI	9.36	19.20	24.74
MPC	6.64	11.03	9.06
Control Method	$\sum\lvert e(x)\rvert$	$\sum\lvert e(y)\rvert$	$\sum\lvert e(z)\rvert$
PI	66.99	95.54	81.95
MPC	21.95	31.23	31.54

The first three columns in Table 1 are the cumulative absolute errors of the angles in the three rotation directions around the axis, and the unit is °. The last three columns are the cumulative absolute errors of the displacements of the three spatial coordinates in mm. It can be seen from Table 1 that the cumulative absolute error of the MPC controller in the entire trajectory tracking period is much smaller than that of the PI controller. It is consistent with theoretical expectations and simulation results.

5. DISCUSSION AND CONCLUSION

In this part, the state space equation of the parallel platform is established from modern control theory. Aiming at the trajectory tracking of the parallel platform workspace, an MPC is designed to complete the solution of the control law. This paper constructs a loss function for optimizing MPC parameters based on the swarm intelligence optimization idea to obtain better control effects. It uses the ADE algorithm to optimize the MPC parameters. Then, we successfully deployed the MPC to the control center of the upper computer using C++. Finally, the physical objects of the parallel platform are controlled, and a good control effect is obtained, which verifies the practicability of the model.

ACKNOWLEDG(E)MENT(S)

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 3 April 2023

REFERENCES

1. B. Monsarrat, C. M. Gosselin, *Singularity analysis of a three-leg six-degree-of-freedom parallel platform mechanism based on Grassmann line geometry*, The International Journal of Robotics Research, **20**, *4*, pp. 312–328 (2001).
2. K.E. Zanganeh, R. Sinatra, J. Angeles, *Kinematics and dynamics of a six-degree-of-freedom parallel manipulator with revolute legs*, Robotica, **15**, *4*, pp. 385–394 (1997).
3. J. Fu, F. Gao, Y. Pan, H. Du, *Forward kinematics solutions of a special six-degree-of-freedom parallel manipulator with three limbs*, Advances in Mechanical Engineering, **7**, *5*, p. 1687814015582118 (2015).
4. D. Zlatanov, M. Dai, R. Fenton, B. Benhabib, *Mechanical design and kinematic analysis of a three-legged six degree-of-freedom parallel manipulator*, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, **9396**, pp. 529–536 (1992).
5. M. Liu, Q. Gu, B. Yang, Z. Yin, S. Liu, L. Yin, W. Zheng, W, *Kinematics model optimization algorithm for six degrees of freedom parallel platform*, Applied Sciences, **13**, *5*, pp. 3082 (2023).
6. Gu, Q. et l., *A novel architecture of a six degrees of freedom parallel platform*, Electronics, **12**, *8*, pp. 1774 (2023).
7. 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).
8. 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).
9. 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).
10. M. Boussoffara, I.B.C. Ahmed, Z. Hajaiej, *Sliding mode controller design: stability analysis and tracking control for flexible joint manipulator*, Revue Roumaine Des Sciences Techniques—Série Électrotechnique et Énergétique, **66**, *3*, pp.161–167 (2021).
11. 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).
12. M.S. Ayas, E. Sahin, I.H. Altas, *High order differential feedback controller design and implementation for a Stewart platform*, Journal of Vibration and Control, **26**, *11-12*, pp. 976–988 (2020).
13. J. Xu, Q. Wang, Q. Lin, *Parallel robot with fuzzy neural network sliding mode control*, Advances in Mechanical Engineering, **10**, *10*, p. 1687814018801261 (2018).
14. J. Zhao, D. Wu, H. Gu, *Performance Evaluation of Stewart-Gough Flight Simulator Based on L 1 Adaptive Control*, Applied Sciences, **11**, *7*, p. 3288 (2021).
15. Z. Bingul, O. Karahan, *Real-time trajectory tracking control of Stewart platform using fractional order fuzzy PID controller optimized by particle swarm algorithm*, Industrial Robot: The International Journal of Robotics Research and Application (2021).
16. H. Tourajizadeh, S. Manteghi, *Design and optimal control of dual-stage Stewart platform using feedback-linearized quadratic regulator*, Advanced Robotics, **30**, *20*, pp. 1305–1321 (2016).
17. Z. Bubnicki, *Modern control theory*. Springer, Berlin, pp. 17–37 (2005).
18. N. Kacimi, A. Idir, S. Grouni, M.S. Boucherit, *A new combined method for tracking the global maximum power point of photovoltaic systems*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **67**, *3*, pp. 349–354 (2022).
19. B. Kouvaritakis, M. Cannon, *Model Predictive Control*, Springer, Switzerland, pp. 122–164 (2016).
20. M. Abdelwanis, R.A.G.B. El-Sehiemy, *Efficient parameter estimation procedure using sunflower optimization algorithm for six-phase induction motor*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **67**, *3*, pp. 259–264 (2020).
21. K.V. Price, *Differential Evolution*, in *Handbook of optimization*: Springer, Berlin, pp. 187–214 (2013).
22. 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**, p. 103479 (2020).
23. C.A. Chen, T.C. Chiang, *Adaptive differential evolution: A visual comparison*, 2015 IEEE Congress on Evolutionary Computation (CEC), 2017, IEEE, pp. 401–408 (2017).
24. X. Yao, Y. Liu, G. Lin, *Evolutionary programming made faster*, IEEE Transactions on Evolutionary Computation, **3**, *2*, pp. 82–102 (1999).
25. I. Farda, A. Thammano, *A self-adaptive differential evolution algorithm for solving optimization problems*, International Conference on Computing and Information Technology, Cham: Springer International Publishing, pp. 68–76 (2022).