





## Article

# Kinematics Model Optimization Algorithm for Six Degrees of Freedom Parallel Platform

Mingzhe Liu <sup>1</sup>, Qiuxiang Gu <sup>2</sup>, Bo Yang <sup>2,\*</sup>, Zhengtong Yin <sup>3</sup>, Shan Liu <sup>2</sup>, Lirong Yin <sup>4</sup>  
and Wenfeng Zheng <sup>2,\*</sup>

- <sup>1</sup> School of Data Science and Artificial Intelligence, Wenzhou University of Technology, Wenzhou 325000, China  
<sup>2</sup> School of Automation, University of Electronic Science and Technology of China, Chengdu 610054, China  
<sup>3</sup> College of Resource and Environment Engineering, Guizhou University, Guiyang 550025, China  
<sup>4</sup> Department of Geography and Anthropology, Louisiana State University, Baton Rouge, LA 70803, USA  
\* Correspondence: boyang@uestc.edu.cn (B.Y.); winfirms@uestc.edu.cn (W.Z.)

**Abstract:** The attitude closed-loop control of the parallel platform in the working space needs to determine the relationship between the pose of the top moving platform and the length of each mechanical arm, that is, the kinematics problem of the parallel platform. In this study, the kinematics model of the six-degree-of-freedom parallel platform was established. The kinematics forward solution algorithm based on Newton–Raphson iteration was studied. The kinematics forward solution method usually adopts a numerical solution, which often needs multiple iterations, and the algorithm has a poor real-time performance. In order to improve the real-time performance of the parallel platform control system, a multivariate polynomial regression kinematics forward solution algorithm is proposed in this paper. Moreover, by combining the multivariate polynomial regression with the Newton iterative method, we obtained an efficient solution algorithm with controllable solution accuracy. The effectiveness of the proposed method was verified by simulation tests and physical tests.

**Keywords:** parallel platform; multivariate polynomial regression; Newton iterative; kinematics



**Citation:** Liu, M.; Gu, Q.; Yang, B.; Yin, Z.; Liu, S.; Yin, L.; Zheng, W. Kinematics Model Optimization Algorithm for Six Degrees of Freedom Parallel Platform. *Appl. Sci.* **2023**, *13*, 3082. <https://doi.org/10.3390/app13053082>

Academic Editor: Dimitris Mourtzis

Received: 7 February 2023

Revised: 24 February 2023

Accepted: 25 February 2023

Published: 27 February 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Multi-degree-of-freedom parallel platforms are widely used in intelligent manufacturing and industrial production, and the research on parallel platforms has also been a research hotspot in recent years. Among them, the dynamics of the parallel platform studies the relationship between the force on the moving platform and the force on each manipulator [1]. Commonly used dynamic modeling methods for parallel platforms include Newton–Euler method [2], Lagrange method [3], spiral theory and virtual work principle [4–6], etc. The dynamic model can be used to design the inverse dynamics controller of the parallel platform, so as to improve the transient, dynamic, and steady-state performance of the parallel platform [7–9].

The movement of the parallel platform in space is realized by changing the length of the six groups of parallel manipulator arms, which is an indirect control [10]. In order to realize arbitrary adjustment of the pose of the parallel platform, it is necessary to establish a mapping relationship between the pose of the parallel platform and the length of each manipulator, that is, a kinematics problem. The kinematics problem of the parallel platform consists of two problems of forward kinematics and inverse kinematics [11,12]. At present, there are many solutions for the kinematics solution of the parallel platform. Guo et al. proposed a control scheme for a five-degree-of-freedom parallel platform based on neural network technology [13]. Abadi et al. present various solutions in Real-Time Solving of Kinematics Problems in Parallel Mechanisms [14–16]. Among them, the kinematics forward solution has a large amount of computation, which makes it difficult to carry out real-time control of the parallel platform in the workspace. In order to avoid this problem,

the expected trajectory of the parallel platform in the workspace is usually mapped to the joint space of the manipulator through the inverse kinematics solution. Then, in the joint space, the closed-loop control of the length of the manipulator is performed according to the mapping value, thereby indirectly realizing the trajectory tracking of the parallel platform. However, there are the following deficiencies in such a control method [17–19]:

- (1) The movement trajectory of each robot arm is calculated by interpolation before movement. For the situation that has deviated from the preset trajectory, no correction can be made, and the control accuracy is not high.
- (2) During the movement of the parallel platform, the convergence speed of each robot arm is different. The length closed-loop control of each manipulator independently according to the mapping value will lead to large process errors in the platform shake and working space.
- (3) It is difficult to realize the speed control of the parallel platform in the working space.

Therefore, this study proposes a kinematics forward solution algorithm based on multivariate polynomial regression to address the difficulty caused by the fact that the kinematics forward solution has a large amount of computation. The optimization goal of controllable solution accuracy and high efficiency is achieved. Through the comparison of simulation experiments, it is concluded that the multivariate polynomial regression method can guarantee the accuracy. The calculation time consumption and the number of iterations is far less than the conclusion of the traditional Newton iterative algorithm.

## 2. Materials and Methods

### 2.1. Kinematic Modeling

To study the kinematics of the parallel platform, it is necessary to establish a mathematical model of the parallel platform. Mathematically abstract the physical system of the parallel platform and use three-dimensional space vectors to describe each component. The six-degree-of-freedom platform studied in this paper is based on the Stewart–Gough platform, so it is referred to as Stewart–Gough in the following [20].

The Stewart–Gough parallel platform is mainly composed of three parts: the bottom static platform, six sets of parallel mechanical arms, and the top dynamic platform. The mechanical arm is connected to the dynamic and static platforms through the structure of universal joints. The six-degree-of-freedom attitude of the parallel platform in the working space refers to the  $xyz$  coordinates of the center of mass of the top moving platform in three-dimensional space. Moreover, the pitch angle ( $\alpha$ ) around the  $x$ -axis, the roll angle ( $\beta$ ) around the  $y$ -axis, and the yaw angle ( $\gamma$ ) around the  $z$ -axis are three rotation angles, expressed as  $[x, y, z, \alpha, \beta, \gamma]^T$  in vector.

In this study, the RPY angle method was used to represent the rotational motion of the moving platform around the axis. The rotation operators  $R_X(\alpha)$ ,  $R_Y(\beta)$ , and  $R_Z(\gamma)$  were used to represent the rotation of the moving platform by an angle of  $\alpha$  around the  $x$ -axis, the angle of  $\beta$  around the  $y$ -axis, and the angle of  $\gamma$  around the  $z$ -axis. The rotation operator of the parallel platform satisfies Equation (1) [21]:

$$\begin{cases} R_X(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \\ R_Y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \\ R_Z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{cases} \quad (1)$$

When the moving platform rotates around multiple axes simultaneously in the workspace, matrix multiplication is performed in the order of  $R_Z(\gamma)$ ,  $R_Y(\beta)$ , and  $R_X(\alpha)$ .

Constitute the rotation matrix  $R_{XYZ}(\alpha, \beta, \gamma)$ . If sin is abbreviated as  $s$  and cos is abbreviated as  $c$ , then we can obtain Equation (2):

$$R_{XYZ}(\alpha, \beta, \gamma) = \begin{bmatrix} c\gamma c\beta & c\gamma s\beta s\alpha - s\gamma c\alpha & c\gamma s\beta c\alpha + s\gamma s\alpha \\ s\gamma c\beta & s\gamma s\beta s\alpha + c\gamma c\alpha & s\gamma s\beta c\alpha - c\gamma s\alpha \\ -s\beta & c\beta s\alpha & c\beta c\alpha \end{bmatrix} \quad (2)$$

In order to study the kinematics relationship of the parallel platform, the mechanical structure of the parallel platform is abstracted mathematically, and the spatial geometric model is established. With the center of mass,  $O$ , of the static platform at the bottom of the parallel platform as the origin, a space Cartesian coordinate system,  $Oxyz$ , is established. With the center of mass,  $P$ , of the top moving platform as the origin, a space rectangular coordinate system,  $Px'y'z'$ , is established. Simplify the mechanical structure of each mechanical arm of the parallel platform and the upper and lower platforms. Establish the parallel platform space geometric model in MATLAB, as shown in Figure 1.

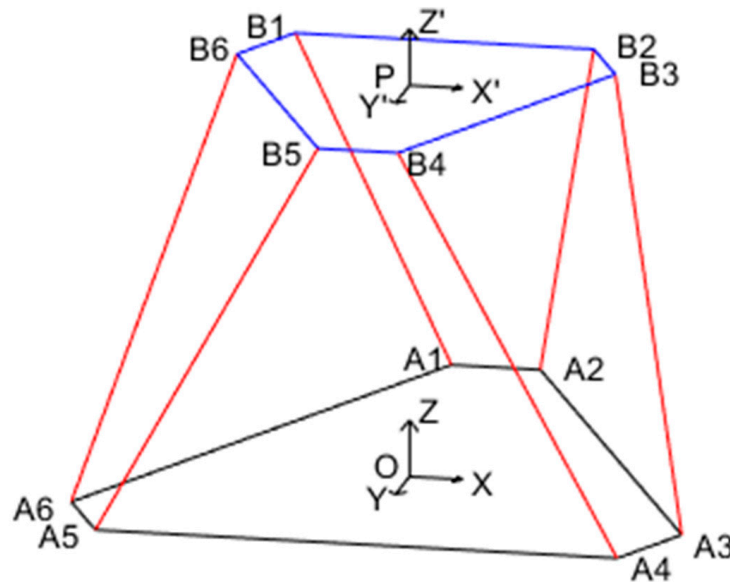
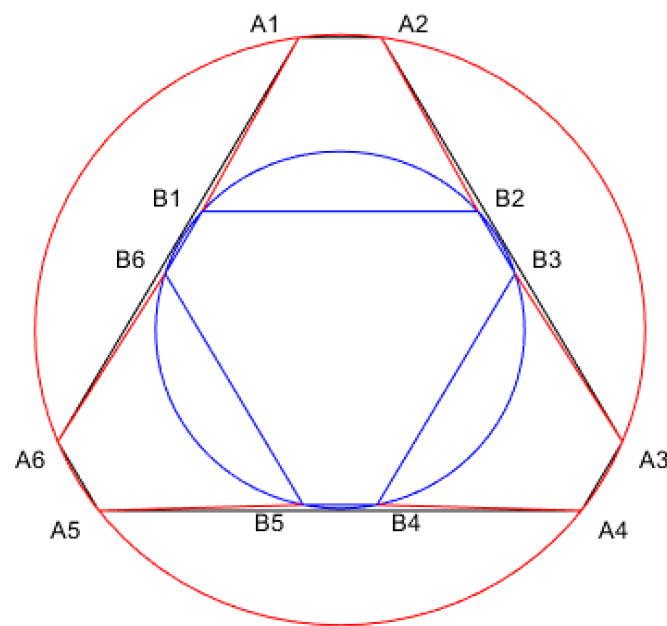


Figure 1. Space geometric model of parallel platform.

A1 to A6 in Figure 1 are the connection points between the bottom static platform and each parallel manipulator arm. B1 to B6 are the connection points between the top moving platform and each group of parallel mechanical arms. The six connection points B1 to B6 of the top moving platform and the mechanical arm are on the same circumscribed circle. The six connection points A1 to A6 of the bottom static platform are on the same circumscribed circle. In the initial zero state, the six robotic arms are of equal length, and the centers of the circumscribed circles of the moving platform and the static platform coincide. Figure 2 shows the top view of the parallel platform in the initial zero state.



**Figure 2.** Top view of the parallel platform in zero state.

## 2.2. Dataset

Table 1 shows the coordinate parameters of the connection points of each parallel manipulator arm, the moving platform, and the static platform at the initial position.

**Table 1.** Physical motor parameters.

Coordinate Points	$p_x$ (mm)	$p_y$ (mm)	$p_z$ (mm)
A1	−25	184	52.5
A2	25	184	52.5
A3	171.85	−70.35	52.5
A4	146.85	−113.65	52.5
A5	−146.85	−113.65	52.5
A6	−171.85	−70.35	52.5
B1	−84.01	74.49	325.4
B2	84.01	74.49	325.4
B3	106.51	35.51	325.4
B4	22.50	−110	325.4
B5	−22.50	−110	325.4
B6	−106.51	35.51	325.4

To reduce the impact of operating system multitask scheduling and task-switching time on the algorithm calculation time-consuming evaluation, this experiment used a large sample batch calculation method to evaluate the algorithm time-consuming evaluation. Each independent sample consists of two parts, input and output. The output is the six-degree-of-freedom attitude of the parallel platform workspace, which is randomly selected on a specific closed subinterval, using the Monte Carlo method. We then performed kinematic inverse solution on the output attitude data. The lengths of six groups of parallel manipulators were obtained as the input information of this sample. When the algorithm was executed, the length information of the mechanical arm of each sample was input, and the kinematics forward solution algorithm was executed. Moreover, we compared the output information obtained by the algorithm with the output information of the sample to find the absolute error  $|e(X_{tar} - X_k)|$ .

Before the execution of the MPR (Multivariate Polynomial Regression) algorithm, all polynomial coefficients need to be solved offline. Therefore, the selected sample set needs to be composed of two parts: the training sample set and the test sample set. Among

them, the training sample set needs to contain at least 1554 linearly independent sample points. Use the Monte Carlo method to randomly select 100,000 sample points of attitudes in the closed subspace  $[x \pm 10 \text{ mm}, y \pm 10 \text{ mm}, z \pm 10 \text{ mm}, \alpha \pm 10^\circ, \beta \pm 10^\circ, \gamma \pm 10^\circ]$  of the workspace. Moreover, through the inverse kinematics solution, the length of the manipulator corresponding to each sample point is obtained as the first training sample set (Test Set 1). In the same sample space, 10,000 sample points are selected as the first test sample set. In order to verify that the Newton iterative algorithm and the MPR algorithm have global convergence, using the same method, in the closed subspace  $[x \pm 40 \text{ mm}, y \pm 40 \text{ mm}, z \pm 40 \text{ mm}, \alpha \pm 40^\circ, \beta \pm 40^\circ, \gamma \pm 40^\circ]$  of the workspace, randomly select 100,000 sample points to form the second training sample set (Test Set 2). Then, in the same sample space, select 10,000 sample points to form the second test sample set.

### 2.3. Parallel Platform Jacobian Matrix

The Jacobian matrix is of great significance in the parallel platform system, it shows the relationship between the speed change between the moving platform and each group of manipulators in the parallel platform system [22,23]. Based on the space geometry model of the parallel platform, the Jacobian matrix of the parallel platform can be deduced. Note that the position vector of the center of mass,  $P$ , of the moving platform relative to the center of mass,  $O$ , of the static platform is  $p$ , and the velocity vector corresponding to the translation of the center of mass of the moving platform is  $v_p$ . Note that the angular velocity of the moving platform around the center of mass is  $w$ . Note that the position vector between the centroid  $O$  of the static platform and the connection point,  $A_i$ , at the lower end of each robot arm is  $p_{A_i}$ . The angular velocity of each mechanical arm around point  $A_i$  is  $w_i$ . Note that the position vector between the centroid  $O$  of the static platform and the connection point  $B_i$  at the upper end of the manipulator arm is  $p_{B_i}$ . When the mechanical arm moves, the velocity vector corresponding to  $B_i$  is  $v_{B_i}$ . According to the spatial geometric model of the parallel platform shown in Figure 1, the length of each manipulator arm can be expressed as Equation (3):

$$l_i n_i = p_{B_i} - p_{A_i} \tag{3}$$

By deriving both sides of Equation (3) with respect to time simultaneously, we can obtain Equation (4):

$$\dot{l}_i n_i + l_i (w_i \times n_i) = v_{B_i} \tag{4}$$

Since the angular velocity,  $w_i$ , of the manipulator around  $A_i$  is always orthogonal to the unit direction vector,  $n_i$ , of the manipulator, multiplying both sides of Equation (4) by  $n_i$  at the same time can eliminate the second half of the left side of the equation. We can obtain the relationship between the length-change speed of the mechanical arm and the speed ( $v_{B_i}$ ) at point  $B_i$ , as shown in Equation (5):

$$\dot{l}_i = v_{B_i}^T n_i \tag{5}$$

According to the spatial geometric model in Figure 1, the motion velocity of point  $B_i$  on the moving platform can be expressed by the translation of the center of mass of the moving platform and the rotation around the center of mass, as shown in Equation (6):

$$v_{B_i} = v_p + w \times p_{B_i} \tag{6}$$

Substitute Equation (6) into Equation (5). The relationship between the length change speed of each mechanical arm and the center-of-mass velocity and angular velocity of the moving platform can be obtained, as shown in Equation (7):

$$\dot{l}_i = v_p + w \times p_{B_i}^T n_i \tag{7}$$

According to Equation (7), the same equations are listed for the six groups of manipulators and written in matrix form. The relationship expressions between the length

change rate of each manipulator and the center-of-mass velocity and angular velocity of the moving platform can be obtained, as shown in Equation (8):

$$\begin{pmatrix} \dot{l}_1 \\ \vdots \\ \dot{l}_6 \end{pmatrix} = \begin{pmatrix} n_1^T & (p_{B_1} \times n_1)^T \\ \vdots & \vdots \\ n_6^T & (p_{B_6} \times n_6)^T \end{pmatrix} \begin{pmatrix} v_p \\ \omega \end{pmatrix} \tag{8}$$

Write Equation (8) as a simplified expression between the length change speed of each mechanical arm and the center-of-mass velocity and angular velocity of the moving platform, and Equation (9) can be obtained:

$$\dot{L} = J\dot{X} \tag{9}$$

In Equation (9),  $J$  is the Jacobian matrix between the moving platform and the manipulator.

#### 2.4. Forward Solution Algorithm of Newton–Raphson Kinematics

The Newton–Raphson method is an approximate solution method proposed by Newton and Raphson in the 17th century to find the root of a nonlinear function [24]. Its core idea is to solve iteratively based on the first-order Taylor expansion of the function  $f(x)$  to be solved, as shown in Equation (10):

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{10}$$

In Equation (10),  $f$  is the function to be solved, and  $x_n$  is the solution at the  $n$ th step. After the Newton iterative method gives the initial solution  $X_0$ , the next approximate solution,  $x_1$ , can be obtained through  $f(x_0)$  and its first derivative. By analogy, gradually make  $f(x_n)$  converge to zero.

With the development of computer technology, Newton’s iterative method has become a general and effective method for solving nonlinear equations. At the same time, it is also a classic numerical solution for the forward kinematics solution of parallel platforms [25]. In order to apply the Newton iterative method to the solution of the forward kinematics solution of the parallel platform, it is necessary to obtain the iterative formula of the parallel platform. Suppose we have a set of robotic-arm-length data,  $L_{tar} = [l_1, l_2, l_3, l_4, l_5, l_6]^T$ . The corresponding pose data on the parallel platform workspace are  $X_{tar} = [x, y, z, \alpha, \beta, \gamma]^T$ . Denote the parallel platform kinematics inverse solution operator as  $G$ . At  $X_k$ , the error between the length of the mechanical arm obtained by the inverse kinematics solution and the given length,  $L_{tar}$ , is  $E(X_k)$ . Therefore, the error  $E(X_{tar})$  is zero when the parallel platform is at  $X_{tar}$ , as shown in Equation (11):

$$E(X_{tar}) = G(X_{tar}) - L_{tar} = 0 \tag{11}$$

In Equation (11), the first-order Taylor expansion of  $E(X_{tar})$  at  $X_k$  can help us obtain Equation (12):

$$E(X_{tar}) = E(X_k) + E'(X_k)(X_{tar} - X_k) + o(X_{tar} - X_k) \tag{12}$$

In Equation (12),  $E'(X_k)$  is the Jacobian matrix,  $J$ , when the moving platform and the manipulator are at  $X_k$ . Discard the high-order infinitesimal term for Equation (12). The combination Equation (11) can be arranged to obtain the relational expression (13) between the target pose,  $X_{tar}$ , and the current pose,  $X_k$ :

$$X_{tar} \approx X_k + J^{-1}(L_{tar} - G(X_k)) \tag{13}$$

Using the idea of Newton’s iterative method, the initial pose,  $X_0$ , of the moving platform is given. Continuously and iteratively approximate  $X_{tar}$ , and then the iterative formula of step  $k + 1$  of the moving platform is Equation (14):

$$X_{k+1} = X_k + J^{-1}(L_{tar} - G(X_k)) \tag{14}$$

It can be seen from Equation (14) that this algorithm takes advantage of the fact that the forward kinematics solution of the parallel platform is difficult, but the kinematics inverse solution is simple. Therefore, it is a classical numerical solution for the forward kinematics solution of the parallel platform.

2.5. Kinematics forward Solution Algorithm Based on Multivariate Polynomial Regression

Through the analysis of the solution process of the Newton iterative algorithm, it is not difficult to find that this method requires multiple iterations, and each iteration needs to complete the 6-order Jacobian matrix inversion operation. Therefore, the Newton iterative algorithm has a huge amount of calculation when solving the forward kinematics solution of the parallel platform. It has a great influence on the real-time performance of the parallel platform control system. In addition, the convergence of Newton’s iterative method is also affected by the selection of initial values. A poor initial value will further increase the number of iterations of the kinematics positive solution of the Newton iterative algorithm and even fail to converge in extreme cases. In order to improve the real-time performance of the control system, this paper proposes a kinematics forward solution algorithm based on MPR [26].

Polynomial regression is a type of regression analysis in statistics. The expression form of a polynomial of degree,  $n$ , is shown in Equation (15):

$$y = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n \tag{15}$$

In Equation (15),  $a_0, a_1, \dots, a_n$  are all constant values.

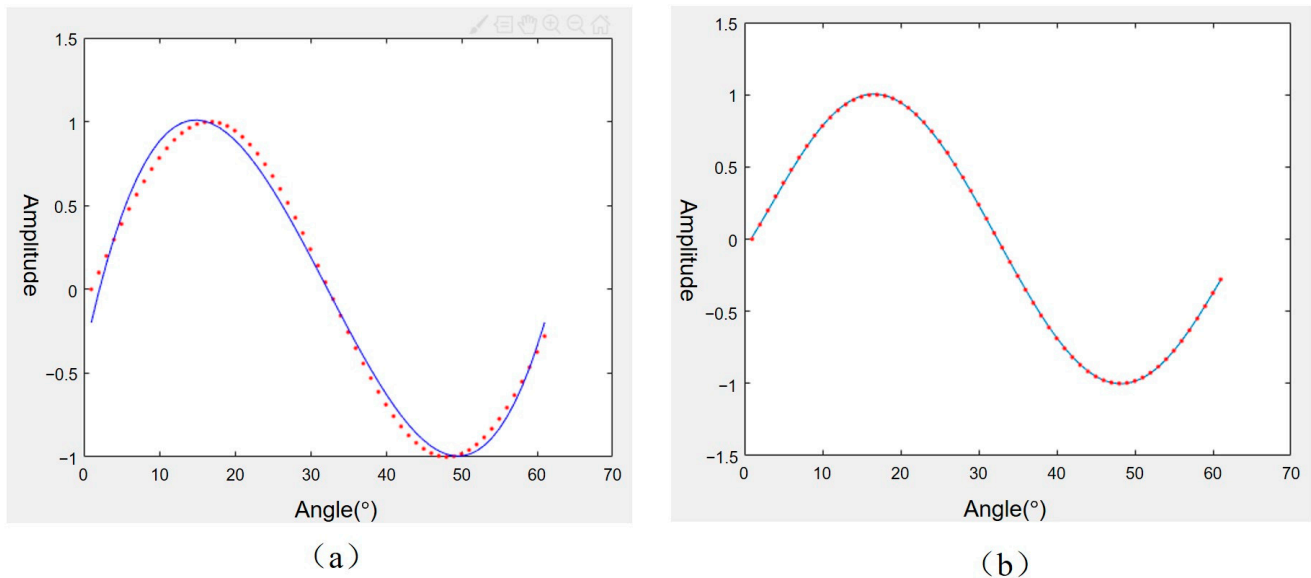
According to the Weierstrass polynomial approximation theorem, any continuous function,  $f(x)$ , on the closed interval  $[a, b]$  can be expressed as a uniformly convergent limit of a polynomial sequence  $\{f_n(x)\}$  [27]. To have an intuitive understanding of this theoretical thought, in MATLAB, use the cubic polynomial regression curve and the quintic polynomial regression curve to approximate the sin function curve; the effect is shown in Figure 3.

It can be seen from Figure 3 that the fitting error of the fitted curve to the discrete sin data points decreases significantly as the polynomial degree increases. In line with the theoretical basis of polynomial regression, based on the theoretical basis of polynomial regression, this paper proposes to use the MPR method to complete the approximation of the forward solution operator of parallel platform kinematics. Then we propose using the obtained approximation function to complete the numerical solution of the kinematics forward solution.

From the spatial geometric model of the parallel platform, it can be seen that the relationship between the moving platform and each group of manipulators is a continuous change. It meets the requirements of the continuous function in the theorem. After the working range is limited to the closed sub-range reachable by the parallel platform, the establishment condition of the polynomial approximation theorem can be fully satisfied. Take the six-variable cubic polynomial regression algorithm as an example [28]. According to the input–output relationship of the kinematics positive solution of the parallel platform, the six-dimensional cubic polynomial can be expressed as Equation (16):

$$X = \sum_{i=1}^6 \sum_{j=1}^6 \sum_{k=1}^6 A_{ijk}l_i l_j l_k + \sum_{m=1}^6 \sum_{n=1}^6 B_{mn}l_m l_n + \sum_{p=1}^6 C_p l_p + D \tag{16}$$

In Equation (16),  $X$  is the posture to be solved in a certain dimension;  $l_i$  represents the length of the input  $i$ -th group of robotic arms;  $A_{ijk}$ ,  $B_{mn}$ ,  $C_p$ , and  $D$  are the polynomial coefficients to be solved;  $A_{ijk}$  is the cubic coefficient of  $l_i$ ;  $B_{mn}$  is the quadratic term coefficient of  $l_i$ ;  $C_p$  is the coefficient of the linear term of  $l_i$ ; and  $D$  is the constant coefficient.



**Figure 3.** The polynomial regression function (red) approximates the sine function (Blue): (a) cubic polynomial approximation and (b) quintic polynomial approximation.

In order to improve the efficiency of the MPR algorithm, the MPR algorithm is completed by offline training + online calculation. The offline training process is the process of solving the polynomial regression coefficients. It can be seen from Equation (16) that, for each degree of freedom of the parallel platform, there are  $6^3 + 6^2 + 6 + 1 = 259$  polynomial coefficients to be solved in the hexagram cubic polynomial. Therefore, for all six degrees of freedom in the parallel platform, there are a total of  $259 \times 6 = 1554$  polynomial coefficients to be solved. In order to realize the complete solution of all polynomial coefficients, at least 1554 sets of linearly independent input and output data need to be given. If and only if 1554 sets of linearly independent input and output data are given, the polynomial regression equations are positive definite, and the unique solution of polynomial coefficients can be obtained.

More than 1554 sets of linearly independent input and output data will make the equation overdetermined. At this time, the polynomial regression equation has no solution, but there is a minimum norm least-squares solution. Due to the serious overfitting phenomenon of the positive definite equation, the error for the training sample is 0, but the error for the random test sample is often large. In order to make the polynomial function have a better generalization performance, the MPR parameters are solved by using the method of overdetermined equations to find the least norm least-squares solution. There are  $n$  groups of training sample points that are uniformly distributed in the selected parameter space. The equation for polynomial parameter solving using least squares is Equation (17):

$$p = (X^T X)^+ X^T Y \tag{17}$$

In Equation (17),  $p$  is a polynomial regression parameter with one degree of freedom, and the dimension is 259.  $X$  is the vectorized robotic arm length data in  $n$  groups of training samples, and its dimension is  $n \times 259$ . Due to the high probability of  $X^T X$  being singular in the case of large samples, the matrix is irreversible; thus, use the  $+$  operator to find its



pseudo-inverse.  $Y$  is the actual value of the degree of freedom to be solved in  $n$  groups of training samples, and the dimension is  $n \times 1$ .

When using MPR to solve it, the underfitting phenomenon will occur if the polynomial degree is selected to be too low. This makes the polynomial function unable to approach the kinematics positive solution operator very well, and the solution error is relatively large. If the degree of polynomial is too high, the time complexity of the algorithm will increase exponentially. Therefore, the balance between calculation amount and calculation accuracy should be considered in the selection of the polynomial degree.

### 2.6. MPR-NR Algorithm Design

The MPR-NR algorithm uses the MPR algorithm to initially locate the kinematics positive solution. Reduce the distance between the initial attitude and the target attitude in the NR algorithm. Then use the NR algorithm to iteratively solve the kinematics positive solution that meets the requirements according to the accuracy requirements. The algorithm flow of using the MPR-NR algorithm for the kinematics forward solution is shown in Figure 4.

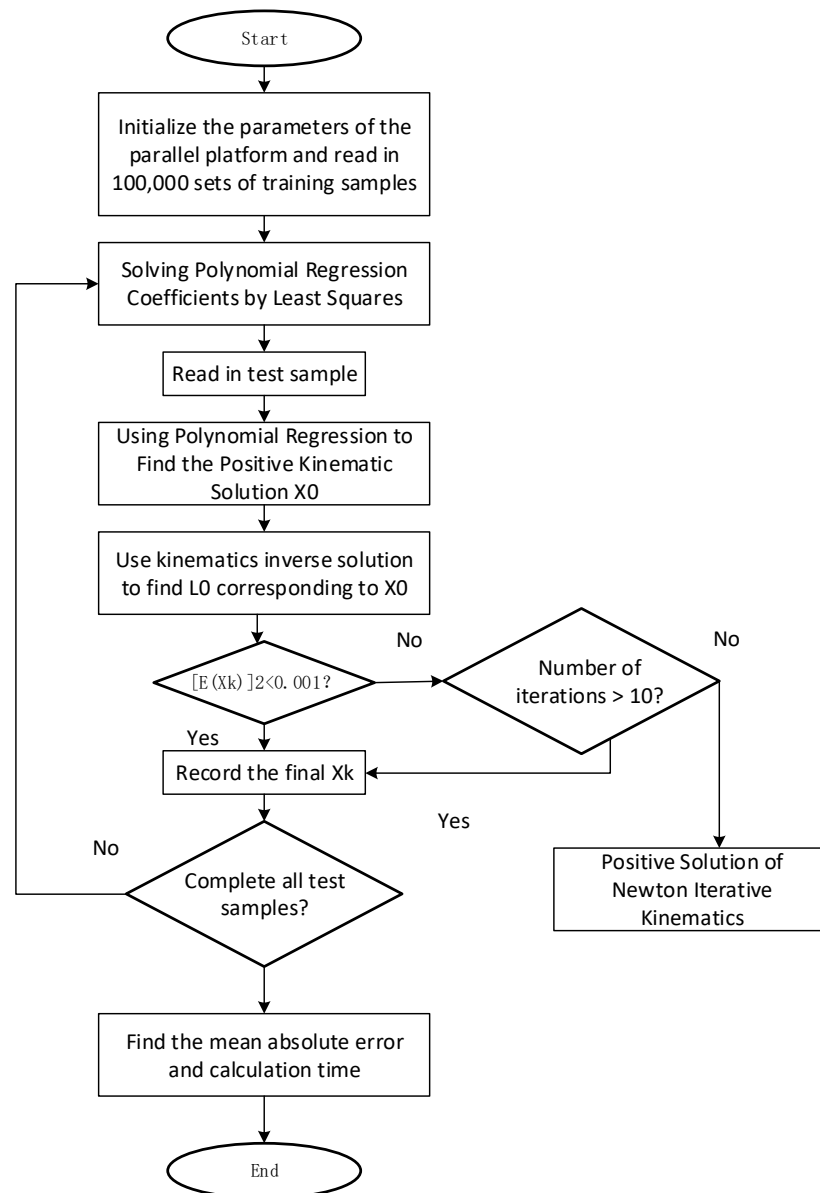


Figure 4. MPR-NR algorithm flowchart.

It can be seen from Figure 4 that when the result calculated by the MPR algorithm for the first time meets the set accuracy, the Newton iteration process will not be entered. If the solution accuracy of the MPR algorithm does not meet the requirements, iterative optimization will be carried out through the Newton iterative algorithm, and finally the accuracy will be improved to the accuracy required by the system. Therefore, the goal of controllable precision can be achieved in this way. On the other hand, the number of iterations for Newton iterative solution is related to the distance between the initial attitude and the target attitude. Given a better initial solution through the MPR algorithm, the number of iterations can be reduced, and the time-consuming solution can be shortened.

### 3. Results

#### 3.1. Single Algorithm Simulation

In order to verify the correctness of the Newton iteration and polynomial regression algorithms, and to conduct a comparative test on the performance of the algorithms, the m language script code is written on the MATLAB platform for simulation experiments. The experimental platform is Windows 10 system, the CPU is i7-8750H 2.2 GHz, and the RAM is 16 GB single-channel DDR4 memory at 2666 MHz.

The initial value of the Newton iterative algorithm is set to the initial attitude of the parallel platform  $[0, 0, 0, 0, 0, 0]^T$ . Set the stop condition of the Newton iterative algorithm as the number of iterations exceeds 10, or the Euclidean distance,  $\sqrt{[E(X_k)]^2}$ , between the length of the manipulator obtained from the inverse solution of the attitude after the kth iteration and the length of the given manipulator is less than 0.001. The parameters of the MPR algorithm are set to two cases of the six-element quadratic polynomial and the six-element cubic polynomial. Before using the MPR algorithm for kinematics positive solution, the polynomial coefficients of MPR are solved using the training sample set in the same closed subspace as the test sample set. Write the least-squares method code according to Equation (17). The minimum norm least-squares solution of the MPR coefficients is obtained by using the length attitude data of the training sample set, and the MPR kinematics positive solution is completed using the obtained parameters. The kinematics positive solutions of the first test sample set and the second test sample set were performed using Newton iterative algorithm, six-variable quadratic polynomial regression (MPR2), and six-variate cubic polynomial regression (MPR3) algorithms respectively. Record the total time-consuming and average absolute error of 10,000 solutions. The test results are shown in Table 2.

**Table 2.** Comparison of Newton iteration method and polynomial regression algorithm.

Algorithm	Dataset	Kinematics Positive Solution Mean Absolute Error $E[ e(X_{tar}-X_k) ]$	Time
Newton iteration	Test Set 1	$[0.3 \ 0.2 \ 1.7 \ 0.4 \ 0.4 \ 0.2] \times 10^{-5}$	1.144
	Test Set 2	$[0.3 \ 0.1 \ 0.6 \ 0.05 \ 0.05 \ 0.03] \times 10^{-3}$	1.815
MPR2	Test Set 1	$[3.4 \ 1.7 \ 2.8 \ 0.4 \ 0.5 \ 0.2] \times 10^{-4}$	0.169
	Test Set 2	$[2.2 \ 1.0 \ 1.8 \ 0.2 \ 0.3 \ 0.1] \times 10^{-2}$	0.164
MPR3	Test Set 1	$[1.8 \ 1.9 \ 2.6 \ 0.3 \ 0.2 \ 0.2] \times 10^{-5}$	0.287
	Test Set 2	$[5.1 \ 3.5 \ 6.4 \ 0.6 \ 0.5 \ 0.3] \times 10^{-3}$	0.272

The first three columns of the average absolute error in Table 2 are the solution errors of the rotation angles of the *x*-axis, *y*-axis, and *z*-axis, respectively, and the unit is radian. The last three columns are the displacement solution errors of the *x*-axis, *y*-axis, and *z*-axis, respectively, in meters. The solution time unit is seconds.

It can be seen from Table 2 that both the Newton iterative algorithm and the MPR algorithm can successfully complete the solution of the positive kinematics solution of the parallel platform. The average single solution time of an algorithm is defined as the total

time consumed by the same algorithm in two sample sets divided by the total number of samples, as shown in Equation (18):

$$\text{Average time for a single solution} = \frac{\text{Computation time of sample set 1} + \text{calculation time of sample set 2}}{\text{Number of samples in sample set 1} + \text{number of samples in sample set 2}} \quad (18)$$

According to Equation (18) and the data in Table 2, the average single solution time of the Newton iterative algorithm is calculated to be about 148 microseconds. The average single solution time of MPR2 is about 17 microseconds. The average single solution time of MPR3 is about 28 microseconds.

Comparing the solution time of MPR2 and MPR3, it can be seen that, with the increase of the polynomial degree, the calculation time consumption of a single solution is significantly improved. Compared with the Newton iterative algorithm, it can be seen that MPR has a significant advantage in solution efficiency. The solution time of MPR2 in the small space Test Set 1 is about 14.7% of Newton iteration, and the solution time of MPR3 with higher solution accuracy is only 25.1% of Newton iteration.

Comparing the time consumption of MPR to solve Test Set 1 and Test Set 2, it can be seen that another characteristic of the MPR algorithm is that it will not significantly increase the calculation time consumption as the solution range increases. This is because the forward solution algorithm of MPR kinematics is composed of a fixed number of multiplications and additions. Therefore, the calculation time of the same algorithm for Test Set 1 and Test Set 2 does not change much. When comparing the calculation time consumption of Newton iterative algorithm in Test Set 1 and Test Set 2, the following conclusions can be drawn. The calculation time of the Newton iterative algorithm to solve Test Set 2 is significantly higher than that of Test Set 1, which increases from 1.144 s to 1.815 s. This is because Test Set 2 has a larger spatial range, so the deviation between the initial value and the target attitude is also greater. The Newton iterative algorithm requires a corresponding increase in the number of iterations, which eventually leads to an increase in calculation time.

In terms of solution accuracy, when comparing the Newton iterative algorithm and the MPR algorithm, it can be seen that the solution accuracy of the Newton iterative algorithm is better than that of the two MPR algorithms. Moreover, the Newton iterative algorithm can set more iterations and a smaller  $\sqrt{[E(X_k)]^2}$  by changing the stop condition to meet the higher precision requirements. The calculation result of the MPR algorithm is only determined by the algorithm itself and the polynomial coefficients. The polynomial coefficients are calculated offline, so the solution accuracy cannot be improved by changing the parameters of the algorithm itself. When comparing the performance of the MPR algorithm in Test Set 1 and Test Set 2, it can be seen that the accuracy of the MPR algorithm decreases by two orders of magnitude as the sample space increases. MPR2 achieved an average angular error of  $0.95^\circ$  and an average displacement error of 2 mm in Test Set 2. In contrast, MPR3 in Test Set 2 has an average angular error of  $0.1^\circ$  and an average displacement error of 0.5 mm. By comparing the data of MPR2 and MPR3, it can be seen that, with the increase of the polynomial degree, the accuracy of the solution was significantly improved. In theory, increasing the polynomial degree of the MPR algorithm can obtain higher solution accuracy. However, with the increase of polynomial coefficients, the amount of calculation for kinematics solution shows a geometric progression. As a result, the advantage of MPR in real-time performance over Newton iterative algorithm is reduced. Therefore, the strategy of increasing the solution accuracy by increasing polynomial coefficients is not sustainable.

Through the simulation experiment, it is found that the Newton iteration method has high solution accuracy, and the polynomial regression solution time is short. In order to achieve a balance between the solution time and solution accuracy, the advantages of controllable solution accuracy of the Newton iterative algorithm and controllable calculation time of the MPR algorithm are utilized at the same time. This paper proposes a kinematics positive solution optimization algorithm (multivariate polynomial regression/Newton–

Raphson iteration, MPR-NR) that uses a combination of polynomial regression and Newton iteration. By combining the advantages of the two algorithms, the forward kinematics solution of the system can reduce the time consumed by the solution under the premise of meeting the accuracy requirements.

### 3.2. MPR-NR Algorithm Simulation

The following experiments were set up to verify that the new method can operate normally according to theoretical expectations. The six-variate quadratic polynomial regression + Newton iterative method (MPR2-NR) and the six-variate cubic polynomial regression + Newton iterative method (MPR3-NR) were used to solve the kinematics positive solution for the same training set and test set. Moreover, compared with the results obtained by the Newton iterative algorithm, the solution accuracy, the calculation time consumption, and the total number of iterations were compared. The results are shown in Table 3.

**Table 3.** Comparison between Newton iteration method and MPR-NR algorithm.

Algorithm	Dataset	Kinematics Positive Solution Mean Absolute Error $E[ e(X_{tar}-X_k) ]$	Time	Iterations
Newton iteration	Test Set 1	$[3.2 \ 2.7 \ 12.5 \ 3.2 \ 3.0 \ 1.4] \times 10^{-5}$	1.144	26,710
	Test Set 2	$[0.3 \ 0.1 \ 0.6 \ 0.05 \ 0.05 \ 0.03] \times 10^{-3}$	1.815	41,664
MPR2-NR	Test Set 1	$[3.4 \ 1.7 \ 2.8 \ 0.4 \ 0.5 \ 0.2] \times 10^{-4}$	0.296	0
	Test Set 2	$[0.3 \ 0.1 \ 0.6 \ 0.05 \ 0.05 \ 0.03] \times 10^{-3}$	1.284	26,019
MPR3-NR	Test Set 1	$[1.8 \ 1.9 \ 2.7 \ 0.3 \ 0.2 \ 0.2] \times 10^{-5}$	0.407	0
	Test Set 2	$[0.4 \ 0.3 \ 0.7 \ 0.07 \ 0.07 \ 0.04] \times 10^{-3}$	1.162	16,998

It can be seen from Table 3 that, in the two sets of test sets, the calculation accuracy of MPR3-NR is at the same level as that of Newton iteration. The MPR2-NR algorithm is an order of magnitude weaker than the other two algorithms in Test Set 1. Using the calculation method of Equation (18), we can obtain an average single solution time of about 148 microseconds for the Newton iterative algorithm. The average single solution time of the MPR2-NR algorithm is about 79 microseconds. The average single solution time of the MPR3-NR algorithm is about 78 microseconds.

For Test Set 1, it can be seen that the number of Newton iterations for both MPR2-NR and MPR3-NR is 0. This phenomenon shows that, in Test Set 1, the accuracy of the MPR algorithm met the system requirements, and there is no Newton iteration process. At this time, compared with the MPR algorithm, the MPR-NR algorithm has one more inverse kinematics solution and the judgment of the accuracy of the solution. Therefore, compared with the corresponding MPR algorithm in Table 2, the average single calculation time of the MPR-NR algorithm is increased by 12 microseconds. For Test Set 2, MPR3-NR takes more time to solve than MPR2-NR when performing a polynomial regression procedure. However, MPR3-NR gives a better initial solution  $X_0$  than MPR2-NR, making the average number of Newton iterations of MPR3-NR 0.78 times less than MPR2-NR. Therefore, the total solution time of the algorithm is that MPR3-NR is slightly better than MPR2-NR.

The following conclusions can be drawn by comparing the number of Newton iterations of the Newton iteration algorithm and the MPR-NR algorithm. After adding the initial solution calculation of polynomial regression, the method of using the Newton iterative algorithm for kinematics forward solution has a better efficiency than directly using the Newton iterative algorithm, and the accuracy can be maintained at the same level. Conclusions can be drawn from the experimental results. The MPR-NR algorithm can guarantee the same accuracy as the Newton iterative algorithm and greatly reduce the solution time, as is consistent with the theoretical analysis conclusion.

In order to further explore the accuracy controllability of the Newton iteration algorithm and the MPR-NR algorithm, the relationship between the algorithm solution accuracy

and the Newton iteration stop condition is obtained. We designed algorithmic accuracy experiments. Assume that the Euclidean distance between the numerical solution of the algorithm and the target attitude is  $D(X_k) = \sqrt{(X_k - X_{tar})^T (X_k - X_{tar})}$ . Let the stop condition be the Euclidean distance,  $D(L_k) = \sqrt{[E(X_k)]^2} = \sqrt{[G(X_k) - L_{tar}]^T [G(X_k) - L_{tar}]}$ , between the length of each group of manipulators after the kinematics forward solution  $X_k$  of the  $k$ -th iteration and the given length of each group of manipulators. Experiment on Test Set 2. The relationship between the iteration stop condition,  $\sqrt{[E(X_k)]^2}$ , of the MPR-NR and Newton iterative algorithm and the solution accuracy and number of iterations of the kinematics forward solution is shown in Table 4.

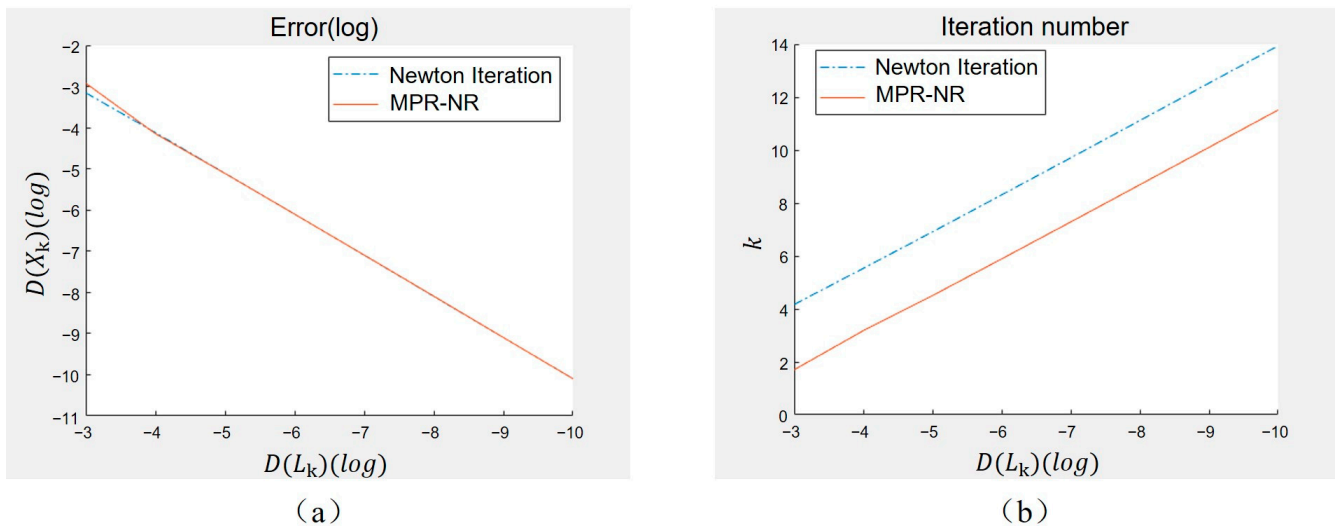
**Table 4.** Relationship between iteration stop conditions and solution accuracy and number of iterations.

$D(L_k)$	Newton Algorithm $D(X_k)$	Average Number of Iterations	MPR-NR $D(X_k)$	Average Number of Iterations
$1 \times 10^{-3}$	$7.00 \times 10^{-4}$	4.16	$1.20 \times 10^{-3}$	1.70
$1 \times 10^{-4}$	$7.59 \times 10^{-5}$	5.53	$7.19 \times 10^{-5}$	3.18
$1 \times 10^{-5}$	$7.81 \times 10^{-6}$	6.91	$7.74 \times 10^{-6}$	4.50
$1 \times 10^{-6}$	$7.96 \times 10^{-7}$	8.31	$7.93 \times 10^{-7}$	5.89
$1 \times 10^{-7}$	$8.01 \times 10^{-8}$	9.71	$8.07 \times 10^{-8}$	7.28
$1 \times 10^{-8}$	$7.99 \times 10^{-9}$	11.12	$8.09 \times 10^{-9}$	8.69
$1 \times 10^{-9}$	$7.99 \times 10^{-10}$	12.53	$8.01 \times 10^{-10}$	10.10
$1 \times 10^{-10}$	$8.03 \times 10^{-11}$	13.94	$7.98 \times 10^{-11}$	11.51

It can be seen from Table 4 that, with the improvement of the accuracy of the stop condition,  $D(L_k)$ , the accuracy of the kinematics forward solution of the Newton iterative algorithm and the MPR-NR algorithm also increases accordingly. The precision controllability of the Newton iteration algorithm and MPR-NR is illustrated. When comparing the kinematics forward solution accuracy  $D(X_k)$  of the MPR-NR algorithm and the Newton iterative algorithm, we can see that the solution accuracy of the two algorithms is at the same level.

By comparing the number of iterations of the MPR-NR algorithm and the Newton iterative algorithm, we can obtain the following conclusions. Since the initial solution  $X_0$  of MPR-NR is better than the Newton iteration algorithm, the average number of iterations is 2.41 times less than that of the Newton iteration algorithm. The relationship between  $D(L_k)$  and  $D(X_k)$  and the number of iterations is shown in Figure 5.

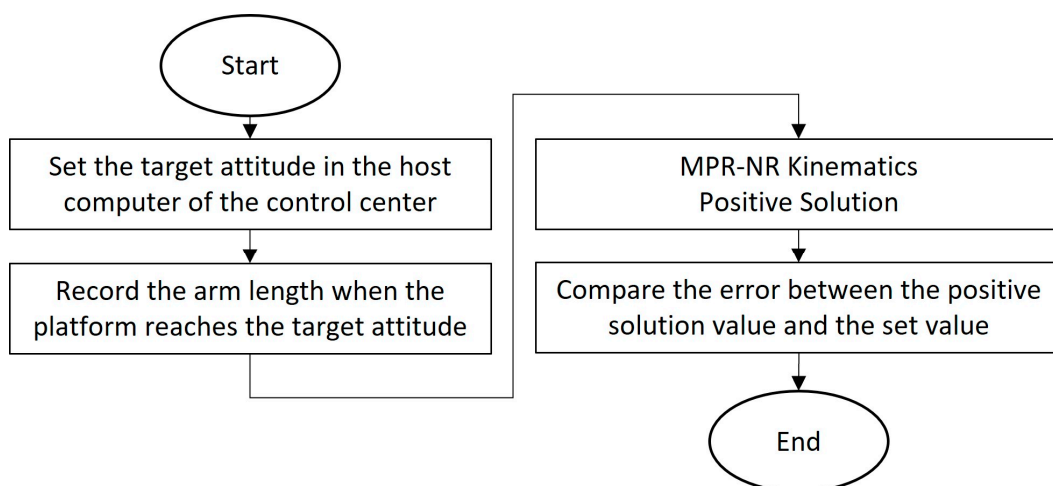
Both  $D(L_k)$  and  $D(X_k)$  in Figure 5a are logarithmic images with base 10. It can be seen from Figure 5a that  $\log_{10}D(L_k)$  and  $\log_{10}D(X_k)$  of the two algorithms present a positive linear relationship. Therefore, every time the stop condition,  $D(L_k)$ , increases by an order of magnitude, the accuracy of the kinematics positive solution,  $D(X_k)$ , also increases by an order of magnitude. It can be seen from Figure 5b that  $\log_{10}D(L_k)$  has a negative linear relationship with the number of iterations of the two algorithms, and the slopes of the two algorithms are similar. Therefore, it can be considered that in the second test sample set, the MPR process in the MPR-NR algorithm is equivalent to completing 2.41 Newton iterative algorithms.



**Figure 5.** The relationship between stopping conditions and solution accuracy: (a)  $D(L_k)$  and  $D(X_k)$  relationship diagram; (b)  $D(L_k)$  and iteration number relationship diagram.

3.3. Physical Test

In order to further verify the correctness and stability of the MPR-NR algorithm running in the parallel platform system, a kinematics positive solution experiment based on the physical platform was designed in this research. For parallel platforms, the kinematics inverse solution is relatively simple. The parallel platform can be moved to a specified attitude by adjusting the length of each group of robotic arms to achieve a closed-loop attitude. The experimental process is designed as follows: The parallel platform is adjusted to the specified attitude in the workspace through the attitude closed-loop control method. Record the arm length information fed back by each servo driver when reaching the target attitude. The length information of the manipulator is imported into the MPR-NR algorithm of the MATLAB platform to calculate the kinematics positive solution. Compare the calculation results with the attitude of the parallel platform physical setting. Through this method, the closed loop from the attitude of the parallel platform to the length of each group of manipulators and then to the attitude of the algorithm simulation can be completed. Then compare the attitude of the parallel platform calculated by the MPR-NR kinematics forward solution algorithm with the attitude of the real object. It can be confirmed whether the algorithm can run as theoretically expected. The experimental process is shown in Figure 6.



**Figure 6.** Flowchart of physical test experiment.

In the host computer of the parallel platform control center, set the target attitude of the parallel platform in the workspace as  $[0^\circ \ 0^\circ \ 5^\circ \ 12 \text{ mm} \ -4 \text{ mm} \ 40 \text{ mm}]$ . The unit of displacement in the MATLAB algorithm is meter, and the unit of angle is radian. Therefore, unit conversion is required when comparing parameters. The converted target attitude value is  $[0 \ 0 \ 0.0870 \ 0.012 \ -0.004 \ 0.040]$ . Click the run button in the host computer and wait for each driver to complete the length adjustment of the mechanical arm. Make the parallel platform run to the set target attitude. The length information of each robot arm when the upper computer of the control center performs in the closed-loop attitude is shown in Figure 7.

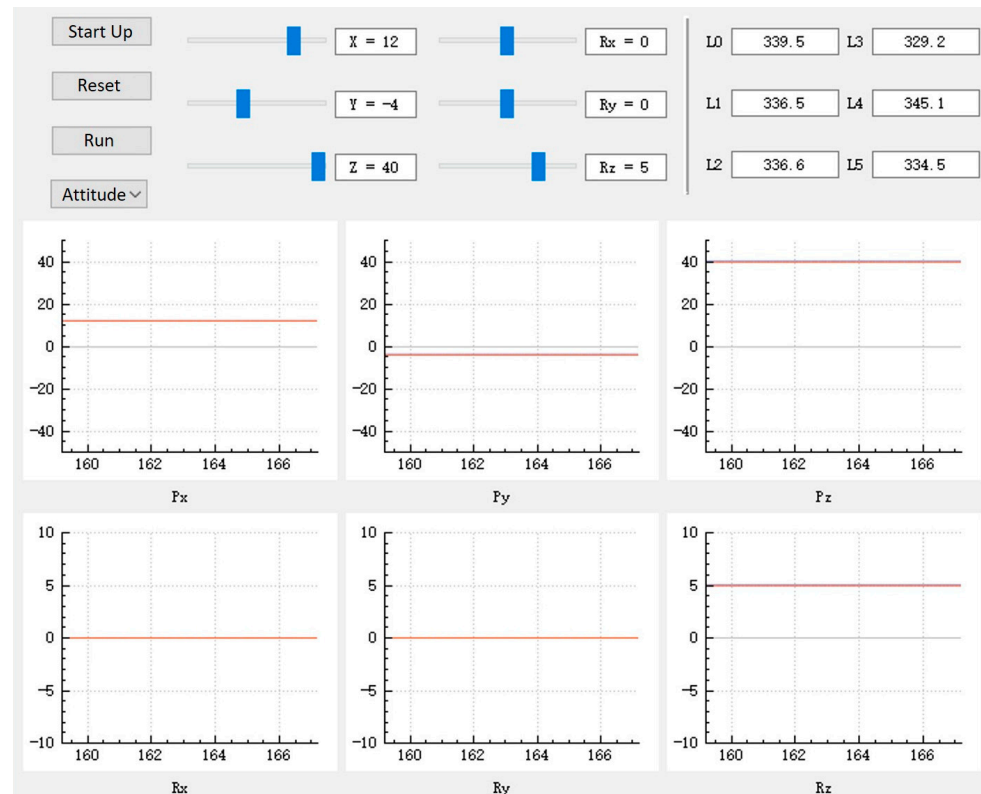


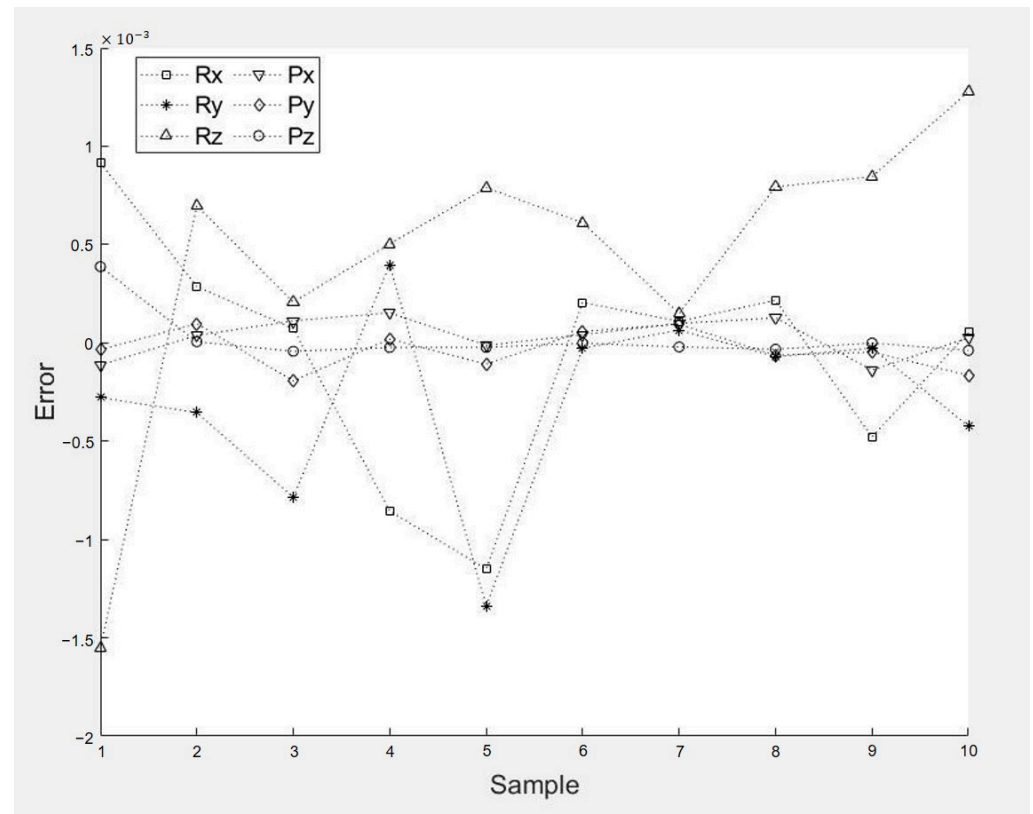
Figure 7. Set the target attitude and read the arm length.

It can be seen from Figure 7 that when the parallel platform performs joint closed-loop control and moves to the end point, the lengths of the six sets of robotic arms are  $[339.5 \ 336.5 \ 336.6 \ 329.2 \ 345.1 \ 334.5]$ . Input the length information of each group of manipulators into MATLAB. Using the MPR-NR algorithm to complete the kinematics forward solution, the attitude of the parallel platform corresponding to the length value of the group of manipulators can be obtained. The reference values of the pose parameters and the calculated values obtained by the MPR-NR algorithm are shown in Table 5.

Table 5. Attitude parameter reference value and MPR-NR algorithm calculation value.

Attitude Parameter	Reference Value	Calculation Value
$\alpha$	0	$-1.68 \times 10^{-4}$
$\beta$	0	$5.83 \times 10^{-5}$
$\gamma$	0.0870	0.0896
$x$	0.0120	0.0120
$y$	-0.0040	-0.0040
$z$	0.0400	0.0398

Comparing the attitude reference value in Table 5 with the attitude calculation value data obtained by the MPR-NR algorithm, it can be seen that the calculation result of the algorithm has a small gap with the set reference attitude. It can be shown that the operation of the MPR-NR algorithm is in line with expectations. The algorithm can successfully complete the calculation of the forward kinematics solution of the parallel platform. According to the experimental process in Figure 6, select 10 attitude data points in the parallel platform control center to test the error of the MPR-NR forward solution algorithm. The result is shown in Figure 8.



**Figure 8.** MPR-NR positive solution error.

It can be seen from Figure 8 that the angle error is in the order of  $10^{-3}$ , and the displacement error is in the order of  $10^{-4}$ . Therefore, the correctness of the operation of the MPR-NR algorithm can be proved. The main reason for the error is that the length value of the robotic arm shown in Figure 7 retains only one decimal place after the decimal point, resulting in a truncation error.

Based on the Eigen open-source matrix operation library, the MPR-NR algorithm is implemented in C++ code under the Qt platform. The code was successfully deployed to the host computer of the parallel platform control center. Use the MPR-NR algorithm to calculate the attitude of the parallel platform. The attitude closed-loop task in the workspace of the parallel platform runs normally, which further proves the stability and reliability of the algorithm.

#### 4. Discussion and Conclusions

In this study, a spatial geometric model was established for a six-degree-of-freedom parallel platform. The Jacobian matrix of the parallel platform was further studied, and the forward solution algorithm of Newton iterative kinematics was deduced. In order to improve the real-time performance of the forward motion solution of the parallel platform,



a kinematics forward solution algorithm based on multivariate polynomial regression (MPR) was proposed.

Through simulation experiments, the following conclusions can be drawn: The MPR algorithm has the characteristics of fast solution speed and controllable calculation time but uncontrollable solution accuracy. The MPR algorithm has a certain accuracy, but with the expansion of the sample space, the accuracy drops significantly, and so it cannot meet the high-precision control requirements. The MPR algorithm can improve the solution accuracy by increasing the degree of polynomials, but the calculation amount will increase geometrically. On the basis of the above, this paper combines the advantages of the two algorithms and further combines the two algorithms into a polynomial regression-Newton iterative algorithm (MPR-NR). The MPR-NR algorithm uses the MPR algorithm to obtain a preliminary solution and then uses the Newton iteration method to improve the solution accuracy. The simulation proves that the MPR-NR algorithm can greatly reduce the number of iterations and improve the real-time performance of the algorithm while maintaining the same accuracy as the Newton iterative algorithm. Finally, the MPR-NR is deployed to the host computer of the control center to verify the correctness and stability of the new algorithm.

In this research, the kinematics of the parallel platform was deeply studied, and the space geometry model of the parallel platform was established. On this basis, the inverse kinematics solution of the parallel platform was analyzed and solved. The forward solution algorithm of Jacobian matrix and Newton iterative kinematics of the parallel platform was realized. In the study, we proposed a multivariate polynomial regression (MPR) kinematics forward solution algorithm. Aiming at the shortcomings of the above algorithms, an algorithm combining multivariate polynomial regression and Newton iteration method (MPR-NR) was proposed. The proposed algorithm has controllable solution accuracy and high efficiency. Through the comparison of simulation experiments, it is concluded that the calculation time and iteration times of the MPR-NR method are much smaller than that of the Newton iterative algorithm under the premise of ensuring accuracy. The feasibility and correctness of the algorithm in the host computer of the control center are verified by physical tests. Finally, the MPR-NR algorithm was successfully transplanted to the host computer of the parallel platform control center.

**Author Contributions:** Conceptualization, W.Z. and B.Y.; methodology, Z.Y. and Q.G.; software, M.L.; validation, Q.G.; formal analysis, M.L. and Q.G.; investigation, B.Y. and S.L.; resources, M.L. and Z.Y.; data curation, Q.G.; writing—original draft preparation, M.L., Z.Y. and L.Y.; writing—review and editing, Z.Y., M.L., W.Z. and L.Y.; visualization, Z.Y., S.L. and M.L.; supervision, B.Y.; project administration, W.Z.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** Support by the Sichuan Science and Technology Program, 2021YFQ0003.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available upon request from the corresponding author. The data are not publicly available due to the fact that the experiment, at current stage, is not at the level to be published.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Zahedi, A.; Shafei, A.; Shamsi, M. Kinetics of planar constrained robotic mechanisms with multiple closed loops: An experimental study. *Mech. Mach. Theory* **2023**, *183*, 105250. [[CrossRef](#)]
2. Bhaskar, D.; Mruthyunjaya, T.S. A Newton-Euler formulation for the inverse dynamics of the Stewart platform manipulator. *Mech. Mach. Theory* **1998**, *33*, 1135–1152.
3. Leroy, N.; Kokosy, A.M.; Perruquetti, W. Dynamic modeling of a parallel robot, Application to a surgical simulator. In Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 14–19 September 2003; pp. 4330–4335.

4. Gallardo, J.; Rico, J.M.; Frisoli, A.; Checcacci, D.; Bergamasco, M. Dynamics of parallel manipulators by means of screw theory. *Mech. Mach. Theory* **2003**, *38*, 1113–1131. [[CrossRef](#)]
5. Tsai, L. Solving the Inverse Dynamics of a Stewart-Gough Manipulator by the Principle of Virtual Work. *J. Mech. Des.* **2000**, *122*, 3–9. [[CrossRef](#)]
6. Nouri Rahmat Abadi, B.; Mahzoon, M.; Farid, M. Singularity-Free Trajectory Planning of a 3-RP RR Planar Kinematically Redundant Parallel Mechanism for Minimum Actuating Effort. *Iran. J. Sci. Technol. Trans. Mech. Eng.* **2019**, *43*, 739–751. [[CrossRef](#)]
7. Sumnu, A.; Güzelbey, I.H.; Cakır, M.V. Simulation and PID control of a Stewart platform with linear motor. *J. Mech. Sci. Technol.* **2017**, *31*, 345–356. [[CrossRef](#)]
8. Kazezkhan, G.; Xiang, B.; Wang, N.; Yusup, A. Dynamic modeling of the Stewart platform for the NanShan Radio Telescope. *Adv. Mech. Eng.* **2020**, *12*, 1687814020940072. [[CrossRef](#)]
9. Tang, J.; Cao, D.; Yu, T. Decentralized vibration control of a voice coil motor-based Stewart parallel mechanism: Simulation and experiments. *J. Mech. Eng. Sci.* **2019**, *233*, 132–145. [[CrossRef](#)]
10. Behi, F. Kinematic analysis for a six-degree-of-freedom 3-PRPS parallel mechanism. *IEEE J. Robot. Autom.* **1988**, *4*, 561–565. [[CrossRef](#)]
11. Zanganeh, K.E.; Sinatra, R.; Angeles, J. Kinematics and dynamics of a six-degree-of-freedom parallel manipulator with revolute legs. *Robotica* **1997**, *15*, 385–394. [[CrossRef](#)]
12. Mueller, A. An overview of formulae for the higher-order kinematics of lower-pair chains with applications in robotics and mechanism theory. *Mech. Mach. Theory* **2019**, *142*, 103594. [[CrossRef](#)]
13. Guo, D.; Xie, Z.; Sun, X.; Zhang, S. Dynamic Modeling and Model-Based Control with Neural Network-Based Compensation of a Five Degrees-of-Freedom Parallel Mechanism. *Machines* **2023**, *11*, 195. [[CrossRef](#)]
14. Nouri Rahmat Abadi, B.; Carretero, J.A. Modeling and Real-Time Motion Planning of a Class of Kinematically Redundant Parallel Mechanisms with Reconfigurable Platform. *J. Mech. Robot.* **2023**, *15*, 021004. [[CrossRef](#)]
15. Abadi, B.N.R.; Farid, M.; Mahzoon, M. Redundancy resolution and control of a novel spatial parallel mechanism with kinematic redundancy. *Mech. Mach. Theory* **2019**, *133*, 112–126. [[CrossRef](#)]
16. Abadi, B.N.R.; Taghvaei, S.; Vatankhah, R. Optimal motion planning of a planar parallel manipulator with kinematically redundant degrees of freedom. *Trans. Can. Soc. Mech. Eng.* **2016**, *40*, 383–397. [[CrossRef](#)]
17. Shimizu, M.; Kakuya, H.; Yoon, W.-K.; Kitagaki, K.; Kosuge, K. Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Trans. Robot.* **2008**, *24*, 1131–1142. [[CrossRef](#)]
18. Nakamura, Y.; Hanafusa, H. Inverse kinematic solutions with singularity robustness for robot manipulator control. *J. Dyn. Syst. Meas. Control* **1986**, *108*, 163–171. [[CrossRef](#)]
19. Sciavicco, L.; Siciliano, B. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE J. Robot. Autom.* **1988**, *4*, 403–410. [[CrossRef](#)]
20. Dasgupta, B.; Mruthyunjaya, T. The Stewart platform manipulator: A review. *Mech. Mach. Theory* **2000**, *35*, 15–40. [[CrossRef](#)]
21. Nguyen, C.C.; Antrazi, S.C.; Zhou, Z.-L.; Campbell, C.E., Jr. Analysis and implementation of a 6 DOF Stewart Platform-based robotic wrist. *Comput. Electr. Eng.* **1991**, *17*, 191–203. [[CrossRef](#)]
22. Chen, D.; Zhang, Y.; Li, S. Tracking control of robot manipulators with unknown models: A jacobian-matrix-adaption method. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3044–3053. [[CrossRef](#)]
23. Umetani, Y.; Yoshida, K. Resolved motion rate control of space manipulators with generalized Jacobian matrix. *IEEE Trans. Robot. Autom.* **1989**, *5*, 303–314. [[CrossRef](#)]
24. Akram, S.; Ann, Q.U. Newtonaphsonn method. *Int. J. Sci. Eng. Res.* **2015**, *6*, 1748–1752.
25. Yang, C.F.; Zheng, S.T.; Jin, J.; Zhu, S.B.; Han, J.W. Forward kinematics analysis of parallel manipulator using modified global Newton-Raphson method. *J. Cent. South Univ. Technol.* **2010**, *17*, 1264–1270. [[CrossRef](#)]
26. De Castro, Y.; Gamboa, F.; Henrion, D.; Hess, R.; Lasserre, J.-B. Approximate optimal designs for multivariate polynomial regression. *Ann. Stat.* **2019**, *47*, 127–155. [[CrossRef](#)]
27. Pinkus, A. Weierstrass and approximation theory. *J. Approx. Theory* **2000**, *107*, 1–66. [[CrossRef](#)]
28. Sinha, P. Multivariate polynomial regression in data mining: Methodology, problems and solutions. *Int. J. Sci. Eng. Res.* **2013**, *4*, 962–965.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.