Int J Adv Manuf Technol (2006) 29: 1284–1294
DOI 10.1007/s00170-005-0009-x

ORIGINAL ARTICLE

Jing-Shan Zhao · Xu Yang · Liang Zhu · Zhi-Jing Feng · Kai Zhou

On the forward and inverse displacement of spatial parallel manipulators

Received: 25 September 2004 / Accepted: 25 January 2005 / Published online: 28 September 2005

Abstract In engineering applications, the singularity, and the forward and inverse displacement of spatial parallel manipulators have received a great deal of attention. This paper presents a new simple but effective methodology to investigate these problems based on the analysis of the degree of freedom (DoF) of the spatial parallel manipulator. Through numerical method, the singularity and the forward displacement problems can be cracked simultaneously. With a numerical example, we point out that the Newton-GMRES algorithm is quite a good redeeming method to solve the forward displacement of spatial parallel manipulators, in which cases the differential coefficient matrix of the Newton−Raphson method is nearly singular.

Keywords Forward displacement · Inverse displacement · Numerical method · Singularity · Spatial parallel manipulator

1 Introduction

The generalized Stewart platform is a fully parallel mechanism with six DoFs [1]. Parallel mechanisms based on the Stewart platform have received a great deal of attention from many researchers. This popularity is a result of the fact that the parallel mechanisms have special attractive characteristics for many robotic applications, such as the flight simulator [2] and the force-torque sensor [3]. It is used in flight and automotive simulators, robotic end-effectors, and other applications requiring spatial mechanisms with high structural stiffness. The kinematic problem of the Stewart platform can be stated as follows: given the lengths of the six variable limbs, find all the possible positions of the movable manipulator [1]. The very basic kinematic problems of parallel mechanisms can be roughly divided into two categories: forward displacement problem (FDP) and inverse displacement problem (IDP).

Solution of the FDP requires resolving the location and orientation of the payload of a manipulator given displacements at sensed joints [4]. Han, Liao, and Liang [5] presented a method to study the forward displacement of a general 5−5 parallel manipulator with closed-form solutions, in which the spherical joints on the base and top platforms are not restricted to be in planes. All solutions have been verified by inverse position analysis. Dhingra, Almadi, Kohli [6] presented closed-form polynomial solutions to the displacement analysis problem of planar 10-link mechanisms with 1 DoF. Using the successive elimination procedure, the input-output polynomials as well as the number of assembly configurations for all mechanisms resulting from two 10-link kinematic chains are presented. Kamra, Kohli, and Dhingra [7] addressed the forward displacement analysis of a six-DoF-platform manipulator with numerical method, which is actuated by four different configurations involving six chains with six joints in each chain.

The direct displacement kinematic problem is to compute the position and orientation of the top platform relative to the base platform when the displacements of the actuating devices are given. On the other hand, the inverse kinematic problem can be imposed as: given the position and orientation of the top platform, calculate the displacements of the actuating devices (or link lengths) which can be used to attain this given position and orientation. The parallel manipulators (multiple branches acting on a common payload with some joint unsensed) solutions of the FDP can be more involved for the parallel device being dependent on the device layout and the joint sensing utilized.

In this paper, the forward and inverse displacement of a manipulator is defined as a particular identified point of the

J.-S. Zhao (✉) · Z.-J. Feng · K. Zhou
Department of Precision Instruments,
Tsinghua University,
Beijing, 100084, China
e-mail: zjs01@mails.tsinghua.edu.cn
Fax: +86-0106-2782351

X. Yang · L. Zhu
Department of Mathematics,
Tsinghua University,
Beijing, 100084, China

manipulator. The absolute coordinates of a certain identified point, say, the geometric center of the end-effector, are expressed as a set of parameter equations. The FDP and IDP are depicted in the nonlinear parameter equations. Through the numerical analysis, the FDP can be easily obtained with an error tolerance that is small enough and meets the engineering requirements. The widely used numerical methods are the Newton−Raphson method and the derivatives [8]. However, the Newton−Raphson method will not be effective when the differential coefficient matrix is nearly singular at the nearby of real solutions. Through examples in this paper, we point out that the Newton-GMRES algorithm is quite a good redeeming method to crack this problem because it can avoid calculating the differential coefficient matrix.

2 Methodology

In this article, we address a new methodology based on the analysis of the DoF of the manipulator [9] to study the forward and inverse displacement of spatial parallel manipulators. The benefit of this methodology is that the whole analysis process can be shortened greatly by analyzing the DoF of the manipulator and studying the kinematics of the limbs. Besides, the inverse displacement problems can be directly obtained; usually with the Newton−Raphson iterative method, the forward displacement problems can also be solved through iterating step by step.

In the following, we will introduce the new methodology to study the forward and inverse displacement of spatial parallel manipulators.

According to [9], the DoF of the manipulator can be gained:

$$F = 6 - d. \tag{1}$$

Where F—the DoF of the manipulator; d—the dimension of the constraints spaces spanned by all of the inverse screws.

According to the physical meaning of the inverse screws, we can decide the independent type of movements the manipulator should execute, including translational movements along the three orthogonal axes and rotational movements around the three orthogonal axes.

After the decision of the movements the manipulator should execute, we can select the least number of parameters to depict the stance relationships of the manipulator with the fixed base.

Considering most spatial parallel manipulators often consist of identical limbs, we can simplify the procedure of the analysis greatly [9]! What we should do, in most cases, is only analyze the kinematics of any one of the same limbs. Therefore, this method, in fact, is very concise and effective.

If we presume that the geometric center point on the manipulator connected by n limbs is the origin of the local coordinate system $o_c x_c y_c z_c$, the end vertexes of each kinematic chain connecting the manipulator are denoted by M_i,(i=1,2⋯,n), the end vertexes of each kinematic chain connecting the fixed base are denoted by B_i,(i=1,2,⋯,n), the pose angles are Ψ, θ, ϕ, where Ψ denotes precession angle, θ nutation angle, and ϕ spinning angle; in the local coordinate system, $o_c x_c y_c z_c$, the coordinates of M_i,(i=1, 2,⋯,n), denoted by $\mathbf{r}^L_{Mi}$, will be very simple.

In the absolute coordinate system $oxyz$, the vectors' coordinates of the end vertexes of each kinematic chain connecting the fixed base are $\mathbf{r}_{B_1}\ \mathbf{r}_{B_2}\ \cdots\ \mathbf{r}_{B_n}$, the coordinates of the geometric center of the manipulator, C, are (x_c, y_c, z_c) and the vectors of the vertexes M_i and point C are denoted by r_{M_i} and $\mathbf{r}_C$.

$$\mathbf{r}_{M_i} = \mathbf{r}_C + A\mathbf{r}^L_{M_i} \tag{2}$$

Therefore,

$$\mathbf{r}_{B_i} = \mathbf{r}_C + A\mathbf{r}^L_{M_i} + \mathbf{r}_{M_iB_i} \tag{3}$$

where A—the transform matrix from the local coordinate system to the absolute one.

Equation 3 expresses the forward and inverse displacements of the manipulator. The inverse displacement can be obtained when the position and posture of the manipulator are given with Eq. 3 directly; while the forward displacement can be gained through solving Eqs. 2 and 3 with numerical method.

Assume function

$$G(\mathrm{var}_1, \mathrm{var}_2, \cdots, \mathrm{var}_F) = \begin{bmatrix} \|\mathbf{r}_{B_1} - \mathbf{r}_{M_1}\|^2 - l_1^2 \\ \|\mathbf{r}_{B_2} - \mathbf{r}_{M_2}\|^2 - l_2^2 \\ \vdots \\ \|\mathbf{r}_{B_F} - \mathbf{r}_{M_F}\|^2 - l_F^2 \end{bmatrix} = \mathbf{0} \tag{4}$$

and

$$J_{F\times F} = G'(X) = \frac{\partial G(\mathrm{var}_1, \mathrm{var}_2, \cdots, \mathrm{var}_F)}{\partial(\mathrm{var}_1, \mathrm{var}_2, \cdots, \mathrm{var}_F)} \tag{5}$$

where var_i, i=1,2,⋯, F—the independent variables.

With numerical methods, such as the Newton−Raphson iterative method, we can solve the forward displacement problems corresponding to each actuation $[l_1\, l_2 \cdots l_F]$, provided that the differential coefficient matrix, $J_{F\times F}$, of $G(\mathrm{var}_1, \mathrm{var}_2, \cdots, \mathrm{var}_F)$ is not singular.

If we presume $X = \begin{bmatrix} \mathrm{var}_1 \\ \mathrm{var}_2 \\ \vdots \\ \mathrm{var}_F \end{bmatrix}$, the *ith* iterative value of X are denoted as $X_i = \begin{bmatrix} \mathrm{var}_1^i \\ \mathrm{var}_2^i \\ \vdots \\ \mathrm{var}_F^i \end{bmatrix}$, the Newton−Raphson

iterative function can be written as:

$$X_{i+1} = X_i - [G'(X_i)]^{-1} G(X_i). \tag{6}$$

That is,

$$X_{i+1} = X_i - J_{F\times F}^{-1} G(X_i). \tag{7}$$

In reality, the DoF of the manipulator are not more than six, therefore, the order of $J_{F\times F}$ is not more than six, either. So, Eq. 7 is an economical numerical method to solve the forward displacement problems of the manipulator.

Besides, the singularity criteria of the manipulator are easily expressed as:

$$|J_{F\times F}| = 0. \tag{8}$$

However in reality, the Newton−Raphson method exhibits the following shortcomings [10]: (i) it is essential that a reasonable estimate to the solution be used to initiate the iterative process, (ii) even with a reasonable initial estimate, the iteration process may be a time-consuming task at certain cases. In fact, any approximate step is acceptable provided that the relative residual of the iterating equation is small enough. The Newton-GMRES algorithm [11] is such an inexact numerical method that simplifies the process through approximating the solution of each Newton step. Especially when the differential coefficient matrix of the Newton−Raphson method is nearly singular at the nearby of real solutions, the merits of the Newton-GMRES algorithm is much more obviously demonstrated. Next, we will introduce the inexact Newton-GMRES algorithm briefly.

Review the Newton iteration of $G(X)$=0,

$$X_{i+1}=X_i+S_i, \text{ where } S_i \text{ is the solution of } G'(X_i)S_i = -G(X_i) \tag{9}$$

When the residual $R_i = G'(X_i)S_i + G(X_i)$ satisfies: $\frac{\|R_i\|}{\|G(X_i)\|} \le \eta_i$ where $\eta_i < 1$ or $\lim_{i\to\infty} \eta_i = 0$, then through iterating step by step, we will obtain a numerical result.

Now if we substitute S_i in Eq. 9 with the solution of $G'(X_i)S_i = -G(X_i) + R_i$, we will get the inexact Newton−Raphson method. Especially, we will list one of the best methods called Newton-GMRES as follows, whose virtue is that it can avoid the calculations of $G'(X_i)$.

To begin with, we call back the process of the GMRES method to solve AX=b.

Step1:

Choose the initial point X_0, set:

$R_0 = b - AX_0,\ v_1 = \frac{R_0}{\|R_0\|},\ \beta = \|R_0\|$.

Step2:

Make use of the Arnoldi process to get the orthonormal basis V_k=$\{v_1,\cdots,v_k\}$ and the corresponding matrix $\overline{H}_l$ of Krylov subspace K_l, where $K_l = Span\left\{R_0, AR_0, \cdots, A^{l-1}R_0\right\}$.

Step3:

Solve the linear least squares as Eq. 10, and note the result as y_k.

$$\min_y{}_k \left\|\beta e_1 - \overline{H}_l y\right\| \tag{10}$$

Step4:

X_k=X_0+$V_k y_k$ is the numerical solution.

Next, set $A = G'(X_k)$, $b = -G(X_k)$, and use the GMRES method to solve Eq. 9, then we get the inexact Newton-GMRES algorithm. Notice that

$$G'(X)\omega = \frac{G(X+h\omega) - G(X)}{h} + O(h)$$

denoted by

$$D_h G(x;\omega) = \begin{cases} 0 & \omega = 0 \\ \frac{G\left(X+h\omega\|X\|/\|\omega\|\right)-G(x)}{h\|X\|}\|\omega\| & \omega \neq 0, X \neq 0 \\ \frac{G\left(X+h\omega/\|\omega\|\right)-G(x)}{h}\|\omega\| & X = 0, \omega \neq 0 \end{cases}$$

and set X_0=0, then the Krylov subspace K_l in step 2 of GMRES can be recast as follows which can avoid the calculating of $G'(X)$:

$$R_0 = -G(X_0)$$

$$R_1 = AR_0 = G'(X)R_0 \approx D_h F(X; R_0)$$

$$R_i = A^i R_0 = G'(X)R_{i-1} \approx D_h F(X; R_{i-1})$$

where i=2,3,$\cdots l$.

The convergence of these two methods is shown in the two theorems below:

Theorem 1 Assume G:$D\subset R^n \to R^n$ is Lipschitz continuous and there exists $G(X^*)$=0, and $G'(X^*)$=0 is nonsingular. Then there is $\delta>0$ such that if $\forall x \in S(X^*,\delta)$, the Newton−Raphson iterate from X by Eq. 6 will converges quadratically to X^*.

Theorem 2 Assume G:$D\subset R^n \to R^n$ is Lipschitz continuous and there exists $G(X^*)$=0, and $G'(X)$ is Lipschitz continuous and nonsingular near X^*. Then the iterate from X by Newton-GMRES will converges linearly to X^*.

First, great importance is attached to the initial point and convergence speed. If the Jacobian matrix is not singular or nearly singular at the nearby of the real solution points, both of the two methods will generate a converging result. However, whether the Newton-GMRES algorithm can give a good approximation depends on the choice of initial point, while the Newton−Raphson method will converge in a wider range of initial point compared to Newton-GMRES algorithm. Moreover, Newton−Raphson will offer a faster

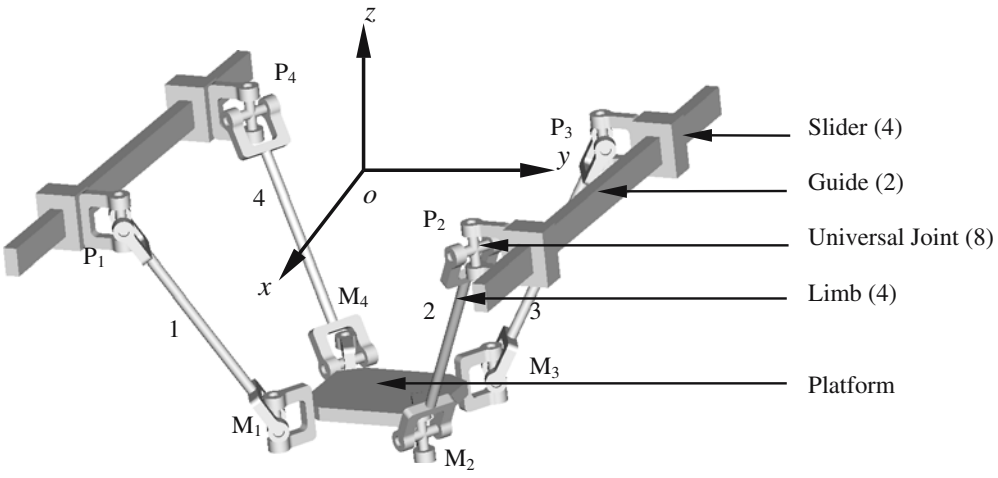

Fig. 1 A spatial parallel mechanism with 4-PUU

convergence speed than Newton-GMRES, which is described in theorem 1 and 2.

Second, when the differential coefficient matrix $G'(X)$ is nearly singular at the nearby of the real solutions, the Newton−Raphson method will be converged to the real solution at a very low speed, while Newton-GMRES algorithm is quite a good redeeming method to crack this problem because it can avoid calculating [11] the differential coefficient matrix.

3 Applications and discussions

3.1 The forward and inverse displacement of a kind of 4-PUU parallel manipulator

A spatial parallel manipulator, shown in Fig. 1, is made up of 4-PUU (1 prismatic joint and 2 universal joints) kinematic chains. The absolute coordinate system *oxyz* are created as Fig. 1 shows, where *z*-axis is perpendicular to the guide plane $P_1P_2P_3P_4$, the origin is on the midline of the two guides, *x* is superposed with the midline of the two guides and *y*-axis is perpendicular to the two guides. According to the method presented above, the local coordinate system $o_c x_c y_c z_c$ is shown in Fig. 2, where z_c-axis is perpendicular to the plane of the manipulator $M_1M_2M_3M_4$, the origin is superposed with the geometric center of $M_1M_2M_3M_4$, x_c and y_c axes are parallel to the two orthogonal sides of the manipulator.

Firstly, we will analyze the DoF of the manipulator as Fig. 1 shows. According to [9], we can find the dimension of the constraints spaces that all of the reciprocal screws, shown in Fig. 3, can be spanned is:

$$d = \dim span \left\{ \begin{matrix} \$^r_{B_1P_1M_1} \\ \$^r_{B_2P_2M_2} \\ \$^r_{B_3P_3M_3} \\ \$^r_{B_4P_4M_4} \end{matrix} \right\} = 2. \tag{11}$$

Therefore,

$$F = 6 - d = 6 - 2 = 4. \tag{12}$$

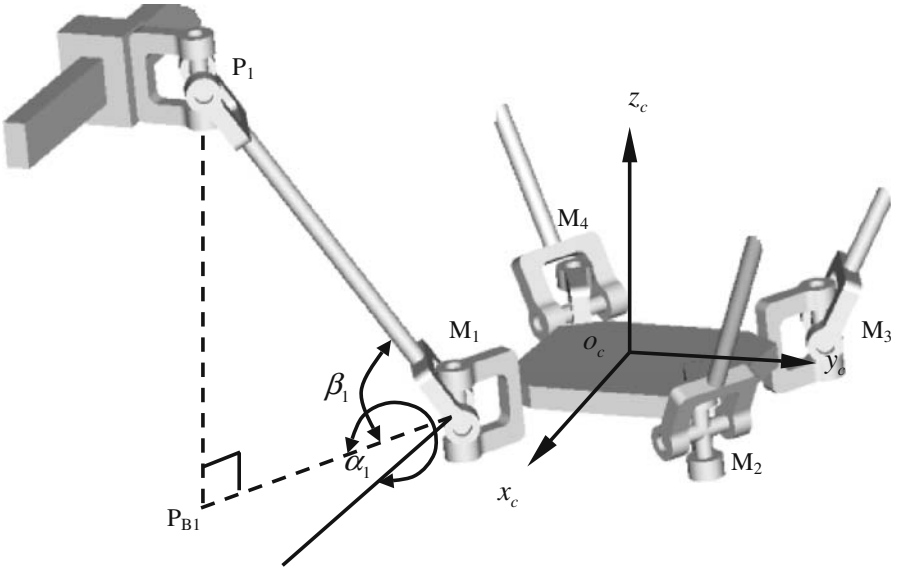

Fig. 2 The local coordinates of spatial parallel mechanism with 4-PUU

So, the manipulator shown in Fig. 1 has four DoFs, including three orthogonal translational movements and one rotational moment around z_c-axis.

Now we can select (x_c y_c z_c) and the rotational angle around z_c-axis, β, as the stance parameters.

If we presume the length of M_1M_2 to be $2a$ and the length of M_1M_4 to be $2b$, the local coordinates of the four vertexes of the manipulator can be obtained:

$$\begin{cases} \mathbf{r}_{M_1}^L = [\,b \quad -a \quad 0\,]^T \\ \mathbf{r}_{M_2}^L = [\,b \quad a \quad 0\,]^T \\ \mathbf{r}_{M_3}^L = [\,-b \quad a \quad 0\,]^T \\ \mathbf{r}_{M_4}^L = [\,-b \quad -a \quad 0\,]^T \end{cases}. \tag{13}$$

The transform matrix from the local coordinate system to the absolute one is:

$$A = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

With Eq. 2, we can obtain:

$$\begin{cases} \mathbf{r}_{M_1} = [\,x_c + b\cos\beta - a\sin\beta \quad y_c - a\cos\beta - b\sin\beta \quad z_c\,]^T \\ \mathbf{r}_{M_2} = [\,x_c + b\cos\beta + a\sin\beta \quad y_c + a\cos\beta - b\sin\beta \quad z_c\,]^T \\ \mathbf{r}_{M_3} = [\,x_c - b\cos\beta + a\sin\beta \quad y_c + a\cos\beta + b\sin\beta \quad z_c\,]^T \\ \mathbf{r}_{M_4} = [\,x_c - b\cos\beta - a\sin\beta \quad y_c - a\cos\beta + b\sin\beta \quad z_c\,]^T \end{cases}. \tag{15}$$

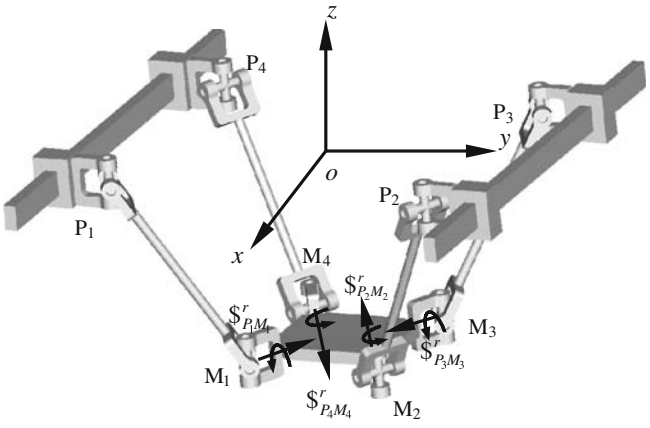

Fig. 3 The local coordinates of spatial parallel mechanism with 4-PUU

If we presume the distance between the two guides is $2c$ and the length of the limb is l, the absolute coordinates of the four vertexes of the base can be obtained:

$$\begin{cases} \mathbf{r}_{B_1} = [\,x_1 \quad -c \quad 0\,]^T \\ \mathbf{r}_{B_2} = [\,x_2 \quad c \quad 0\,]^T \\ \mathbf{r}_{B_3} = [\,x_3 \quad c \quad 0\,]^T \\ \mathbf{r}_{B_4} = [\,x_4 \quad -c \quad 0\,]^T \end{cases}. \tag{16}$$

With Eq. 3, we can obtain:

$$\begin{cases} |\mathbf{r}_{B_1} - \mathbf{r}_{M_1}| = \sqrt{[x_1 - (x_c + b\cos\beta - a\sin\beta)]^2 + (c + y_c - a\cos\beta - b\sin\beta)^2 + z_c^2} \\ |\mathbf{r}_{B_2} - \mathbf{r}_{M_2}| = \sqrt{[x_2 - (x_c + b\cos\beta + a\sin\beta)]^2 + [c - (y_c + a\cos\beta - b\sin\beta)]^2 + z_c^2} \\ |\mathbf{r}_{B_3} - \mathbf{r}_{M_3}| = \sqrt{[x_3 - (x_c - b\cos\beta + a\sin\beta)]^2 + [c - (y_c + a\cos\beta + b\sin\beta)]^2 + z_c^2} \\ |\mathbf{r}_{B_4} - \mathbf{r}_{M_4}| = \sqrt{[x_4 - (x_c - b\cos\beta - a\sin\beta)]^2 + (c + y_c - a\cos\beta + b\sin\beta)^2 + z_c^2} \end{cases},$$

That is,

$$\begin{cases} (x_c + b\cos\beta - a\sin\beta - x_1)^2 + (y_c - a\cos\beta - b\sin\beta + c)^2 + z_c^2 = l^2 \\ (x_c + b\cos\beta + a\sin\beta - x_2)^2 + (y_c + a\cos\beta - b\sin\beta - c)^2 + z_c^2 = l^2 \\ (x_c - b\cos\beta + a\sin\beta - x_3)^2 + (y_c + a\cos\beta + b\sin\beta - c)^2 + z_c^2 = l^2 \\ (x_c - b\cos\beta - a\sin\beta - x_4)^2 + (y_c - a\cos\beta + b\sin\beta + c)^2 + z_c^2 = l^2 \end{cases} \tag{17}$$

where l—the length of the limb.

The inverse displacement problem is very simple, because when the position and posture of the manipulator are given, the stance parameters (x_c y_c z_c) and β are known, and therefore, the displacement of each kinematic chain can be gained by extracting Eq. 17 directly. In order to

solve the forward displacement of the manipulator, we can assume function $G(\mathrm{var}_1,\mathrm{var}_2,\cdots,\mathrm{var}_F)=\mathbf{0}$:

$$G(x_c,y_c,z_c,\beta)=\begin{bmatrix}(x_c+b\cos\beta-a\sin\beta-x_1)^2+(y_c-a\cos\beta-b\sin\beta+c)^2+z_c^2-l^2\\(x_c+b\cos\beta+a\sin\beta-x_2)^2+(y_c+a\cos\beta-b\sin\beta-c)^2+z_c^2-l^2\\(x_c-b\cos\beta+a\sin\beta-x_3)^2+(y_c+a\cos\beta+b\sin\beta-c)^2+z_c^2-l^2\\(x_c-b\cos\beta-a\sin\beta-x_4)^2+(y_c-a\cos\beta+b\sin\beta+c)^2+z_c^2-l^2\end{bmatrix} \tag{18}$$

According to Eq. 5, there are:

$$J_{4\times4}=\frac{\partial G(x_c,y_c,z_c,\beta)}{\partial(x_c,y_c,z_c,\beta)}$$
$$=2\begin{bmatrix}x_c+b\cos\beta-a\sin\beta-x_1 & y_c-a\cos\beta-b\sin\beta+c & z_c & [a(y_c+c)+b(x_1-x_c)]\sin\beta+[a(x_1-x_c)-b(y_c+c)]\cos\beta\\x_c+b\cos\beta+a\sin\beta-x_2 & y_c+a\cos\beta-b\sin\beta-c & z_c & [a(c-y_c)+b(x_2-x_c)]\sin\beta+[a(x_c-x_2)-b(y_c-c)]\cos\beta\\x_c-b\cos\beta+a\sin\beta-x_3 & y_c+a\cos\beta+b\sin\beta-c & z_c & [a(c-y_c)+b(x_c-x_3)]\sin\beta+[a(x_c-x_3)+b(y_c-c)]\cos\beta\\x_c-b\cos\beta-a\sin\beta-x_4 & y_c-a\cos\beta+b\sin\beta+c & z_c & [a(y_c+c)+b(x_c-x_4)]\sin\beta+[a(x_4-x_c)+b(y_c+c)]\cos\beta\end{bmatrix} \tag{19}$$

Let $|J_{4\times4}|=0$, we can obtain:

$$\begin{vmatrix}x_c+b\cos\beta-a\sin\beta-x_1 & y_c-a\cos\beta-b\sin\beta+c & z_c & [a(y_c+c)+b(x_1-x_c)]\sin\beta+[a(x_1-x_c)-b(y_c+c)]\cos\beta\\x_c+b\cos\beta+a\sin\beta-x_2 & y_c+a\cos\beta-b\sin\beta-c & z_c & [a(c-y_c)+b(x_2-x_c)]\sin\beta+[a(x_c-x_2)-b(y_c-c)]\cos\beta\\x_c-b\cos\beta+a\sin\beta-x_3 & y_c+a\cos\beta+b\sin\beta-c & z_c & [a(c-y_c)+b(x_c-x_3)]\sin\beta+[a(x_c-x_3)+b(y_c-c)]\cos\beta\\x_c-b\cos\beta-a\sin\beta-x_4 & y_c-a\cos\beta+b\sin\beta+c & z_c & [a(y_c+c)+b(x_c-x_4)]\sin\beta+[a(x_4-x_c)+b(y_c+c)]\cos\beta\end{vmatrix}=0$$

As a result, the singularity criteria of the manipulator can be simplified as:

$$z_c\begin{vmatrix}x_1-x_2+x_3-x_4 & 0 & [b(-x_1+x_2+x_3-x_4)]\sin\beta+[a(-x_1-x_2+x_3+x_4)]\cos\beta\\2a\sin\beta-x_3+x_4 & 2a\cos\beta-2c & [-2ay_c+b(x_4-x_3)]\sin\beta+[a(2x_c-x_3-x_4)-2bc]\cos\beta\\-2b\cos\beta+x_1-x_4 & 2b\sin\beta & [b(2x_c-x_1-x_4)]\sin\beta+[a(x_4-x_1)+b(2y_c+2c)]\cos\beta\end{vmatrix}=0 \tag{20}$$

Considering the real applications of the manipulator shown in Fig. 1, we can assume that $z_c\geq0$. Therefore, if the manipulator is not working at the singular position and posture, Eq. 20 will not hold. So we can create an iterative function:

Presume $X=\begin{bmatrix}x_c\\y_c\\z_c\\\beta\end{bmatrix}$, the *ith* iterative value of X are denoted as $X_i=\begin{bmatrix}x_{ci}\\y_{ci}\\z_{ci}\\\beta_i\end{bmatrix}$; we can get the Newton iterative function with Eq. 7:

$$X_{i+1}=X_i-J_{4\times4}^{-1}G(X_i). \tag{21}$$

In the following we will illustrate some numerical results to show the benefits and drawbacks of the Newton–Raphson method and Newton-GMRES algorithm.

Firstly, we introduce the criteria of the iterations' termination:

For the Newton–Raphson method, we terminate the iteration after $\|X_{i+1}-X_i\|_2\leq1e-6$ and $rcond(G'(X))<1e-8$.

Table 1

Iterating step	x_c	y_c	z_c	β
0	0.001	0.001	−0.001	0.0001
1	0.01007256754118	0.00089598775801	−4.50098842914592	−0.00005574077247
2	0.01186081905120	−0.00007108435438	−2.25113873217699	0.00000659364519
3	0.01199933845403	0.00000097949565	−1.12667639967012	−0.00000000957219
4	0.01199999998841	−0.00000000000704	−0.56554561756899	0.00000000000000
5	0.01200000000000	0.00000000000000	−0.28717041240213	0.00000000000000
6	0.01200000000000	0.00000000000000	−0.15224572743287	0.00000000000000
7	0.01200000000000	0.00000000000000	−0.09245859605975	0.00000000000000
8	0.01200000000000	−0.00000000000000	−0.07312831618227	−0.00000000000000
9	0.01200000000000	−0.00000000000000	−0.07057349378481	−0.00000000000000
10	0.01200000000000	−0.00000000000000	−0.07052725037860	0.00000000000000
11	0.01200000000000	−0.00000000000000	−0.07052723521813	−0.00000000000000
12	0.01200000000000	−0.00000000000000	−0.07052723521812	0.00000000000000

(Unit: 10^4 mm)

Where *rcond*(*X*) is an estimate for the reciprocal of the condition of *X* in the 1-norm obtained by the LAPACK condition estimator. If *X* is well conditioned, *rcond*(*X*) is near 1.0. If *X* is badly conditioned, *rcond*(*X*) is near zero.

For the Newton-GMRES algorithm, we terminate the iteration after

$$\frac{\|G(X_{i+1})\|_2}{\|G(X_i)\|_2} < 1e-6$$

As an example, we give a set of parameters below: $a = 100,\quad b = 120,\quad c = 1000,\quad l = 1500,\quad x_1 = x_2 = 1210.87121146357, x_3 = x_4 = -970.87121146357.$

The numerical solutions can be gained:

$$\begin{cases} x_c = 120.0000 \\ y_c = -0.0000 \\ z_c = -705.2724 \\ \beta = 0.0000 \end{cases}.$$

No matter what the initial iterating point is chosen, the Newton−Raphson method will get to the above solutions in about 12 steps (see Tables 1 and 2). However, the Newton-GMRES algorithm can do it only in the condition that the choice of β_0 is in the field of (−0.34, 0.33) with different steps and accuracy for each one (see Tables 3 and 4), which seems that the Newton−Raphson method is much better than Newton-GMRES algorithm. Naturally, this will pose a new question—why should we utilize the worse one? It is because the Newton−Raphson method will come across a lot of trouble when solving some problems in which Jacobian matrices are nearly singular at the nearby of the real solutions.

For instance, given another set of parameters, $a = 100,\quad b = 400,\quad c = 1000,\quad l = 3500,\ x_1 = 515.49,\ x_2 = 1284.51,\ x_3 = -515.49,\ x_4 = -1284.51.$

The numerical solutions can be obtained through 2242 steps' iterating with the Newton−Raphson method when the initial values are given as x_c=10, y_c=10 and β=1. We

Table 2

Iterating step	x_c	y_c	z_c	β
0	0.001	0.001	−0.001	0.00002
1	0.01196385088824	0.00024595006218	−2.59486518091089	−0.00000027100355
2	0.01199991758001	−0.00000001065950	−1.29839105444424	0.00000000000066
3	0.01200000000000	0.00000000000000	−0.65111100980747	0.00000000000000
4	0.01200000000000	0.00000000000000	−0.32937519988093	−0.00000000000000
5	0.01200000000000	0.00000000000000	−0.17223839749491	−0.00000000000000
6	0.01200000000000	0.00000000000000	−0.10055875165743	−0.00000000000000
7	0.01200000000000	−0.00000000000000	−0.07501163843902	−0.00000000000000
8	0.01200000000000	0.00000000000000	−0.07066128023211	0.00000000000000
9	0.01200000000000	0.00000000000000	−0.07052736236036	0.00000000000000
10	0.01200000000000	0.00000000000000	−0.07052723521824	0.00000000000000
11	0.01200000000000	0.00000000000000	−0.07052723521812	−0.00000000000000

(Unit: 10^4 mm)

Table 3

Iterating step	x_c	y_c	z_c	β
0	0.001	0.001	−0.001	0.00003000000000
1	0.00703600650712	−0.00154910430670	−0.02128962900894	0.00020289204720
2	0.01175982851880	−0.00022269780653	−0.03294687815762	0.00013999558678
3	0.01176130035341	−0.00022073354397	−0.04064788904709	0.00012886660095
4	0.01176401887640	−0.00021746380533	−0.04449993639555	0.00012179881998
5	0.01177056088405	−0.00020977948801	−0.05027790740158	0.00010989783645
6	0.01177753729412	−0.00020182866980	−0.05364903472042	0.00010157659594
7	0.01178478207342	−0.00019374460278	−0.05614699350572	0.00009464590032
8	0.01179250266049	−0.00018528986217	−0.05821762051648	0.00008831782550
9	0.01180837725932	−0.00016822207060	−0.06165566397320	0.00007682177637
10	0.01177056088405	−0.00015244678663	−0.06391605925383	0.00006776657074
11	0.01183874575703	−0.00013724539418	−0.06560561745820	0.00005985573396
12	0.01186916920659	−0.00010763125145	−0.06824332130305	0.00004546977559
13	0.01189605255771	−0.00008307224763	−0.06960357429456	0.00003465024853
14	0.01194546009674	−0.00004024544655	−0.07111596475897	0.00001659746081
15	0.01193213723058	−0.00006360705948	−0.07112842819196	−0.00000261153248
16	0.01200019859025	−0.00000020283175	−0.07054228326054	0.00000000070427
17	0.01199999999545	−0.00000000000338	−0.07052727830511	−0.00000000000000
18	0.01199999999999	−0.00000000000001	−0.07052723533576	−0.00000000000000

(Unit: 10^4 mm)

also find that the number of steps doesn't change a lot with respect to the criteria of the iterations' termination. The numerical solutions are:

$$\begin{cases} x_c = 0.0000 \\ y_c = -0.0000 \\ z_c = -3234.5257 \\ \beta = -1.11023852799 \approx -63.6120. \end{cases}$$

However, if we use the Newton-GMRES algorithm, the similar precision values can be obtained only in about 12 steps when the initial values are also give as x_c=10, y_c=10, z_c = −10 and β=1, which is shown in Table 5.

Generally, we can utilize the Newton−Raphson method to solve the forward displacement problems. However, if it costs too many steps (e.g. more than 50 steps) or terminates for the singularity of Jacobian matrix, we had better turn to the Newton-GMRES method for help just as the above example shows.

3.2 The forward and inverse displacement of a kind of 3-UPU parallel manipulator

Another kind of spatial parallel manipulator, shown in Fig. 4, is made up of 3-UPU (one prismatic joint and two universal joints) kinematic chains.

Firstly, we will create an absolute coordinates *oxyz* as Fig. 4 shows. The plane *xoy* is parallel to the fixed base $B_1B_2B_3$ and *z*-axis is perpendicular to $B_1B_2B_3$. Assuming that the origin of *oxyz* is superposed with the geometric center and the radius of the circumcircle of triangle $B_1B_2B_3$ is *R*, there are:

$$B_1\,(0\ -R\ \ 0),\ B_2\left(\frac{\sqrt{3}}{2}R\ \ \frac{1}{2}R\ \ 0\right),\ B_3\left(-\frac{\sqrt{3}}{2}R\ \frac{1}{2}R\ 0\right).$$

If we presume that the geometric center of manipulator $M_1M_2M_3$ is *C*, the local coordinate system $o_cx_cy_cz$ will be shown in Fig. 4, whose original is superposed

Table 4

Iterating step	x_c	y_c	z_c	β
0	0.001	0.001	−0.001	0
1	0.00122893254899	0.00097921533333	−0.05384191155220	−0.00000000172773
2	0.01200355974511	−0.00000031776022	−0.07408489295672	0.00000000004688
3	0.01199999896092	−0.00000000063830	−0.07062215886114	0.00000000000010
4	0.01199999996792	−0.00000000001679	−0.07052755799145	−0.00000000000001
5	0.01199999999989	−0.00000000000006	−0.07052723609999	0.00000000000000

(Unit: 10^4 mm)

Table 5

Iterating step	x_c	y_c	z_c	β
0	0.01	0.01	−0.01	0.001
1	0.01002727590201	0.00994029542740	−2.44342429403359	0.00099293405537
2	0.01644033037639	−0.00473981880509	−3.66521478023394	−0.00074325278941
3	0.28668086385939	−0.00995806266809	−3.38486715802365	0.00006819729530
4	0.00660132645414	−0.02404652275379	−3.33423994880415	−0.00059992246340
5	−0.00349516046401	0.00189863017706	−3.27047407677854	−0.00085555168006
6	0.00025727098337	−0.00011488013470	−3.25153778967230	−0.00098351171497
7	−0.00000854149353	0.00000390711998	−3.24296029956272	−0.00104684242980
8	0.00000008441472	−0.00000004512427	−3.23877698906782	−0.00107821869979
9	−0.00000002712139	0.00000000973937	−3.23669515556281	−0.00109389030365
10	−0.00000001354354	0.00000000466689	−3.23564427759435	−0.00110180831694
11	−0.00000000724799	0.00000000244750	−3.23508958764150	−0.00110598872076
12	−0.00000000436682	0.00000000144994	−3.23475759891854	−0.00110849088849

(Unit: 10^3 mm)

with point C. The vertexes of the manipulator are denoted by M_i,(i=1,2,3), and the prismatic pairs are denoted by P_i(i=1,2,3).

In the absolute coordinate system, the vector coordinates of M_i,(i=1,2,3) are r_{M_1}, r_{M_2} and r_{M_3}, the vector coordinates of C are (x_c,y_c,z_c).

With the similar process of 3.1, we can find the DoF of the manipulator in Fig. 4 is three. According to the screw theory and the analysis, the three orthogonal rotations of the manipulator will be forbidden by the reciprocal screws and the manipulator can only execute three independent translational movements along x, y, and z axes. Therefore, the manipulator $M_1M_2M_3$ will always be parallel to the base $B_1B_2B_3$; and therefore, the transform matrix A=I, where I denotes the unitary matrix.

If we presume that the radius of the circumcircle of triangle $M_1M_2M_3$ is r, the coordinates of the three vertexes of the manipulator in the local coordinate system are as follows:

$$M_1(0 \quad -r \quad 0), M_2\left(\tfrac{\sqrt{3}}{2}r \quad \tfrac{1}{2}r \quad 0\right), M_3\left(-\tfrac{\sqrt{3}}{2}r \quad \tfrac{1}{2}r \quad 0\right).$$

Considering the geometry characteristics, we can obtain the following equation:

$$\mathbf{r}_{B_iM_i} = \mathbf{r}_{o_c} + A\mathbf{r}_{o_cM_i} - \mathbf{r}_{B_i} = \mathbf{r}_{o_c} + \mathbf{r}_{o_cM_i} - \mathbf{r}_{B_i}, (i = 1, 2, 3) \tag{22}$$

$$\|\mathbf{r}_{B_iM_i}\| = l_i, (i = 1, 2, 3) \tag{23}$$

where

l_i,(i=1,2,3)—the lengths of the limbs.

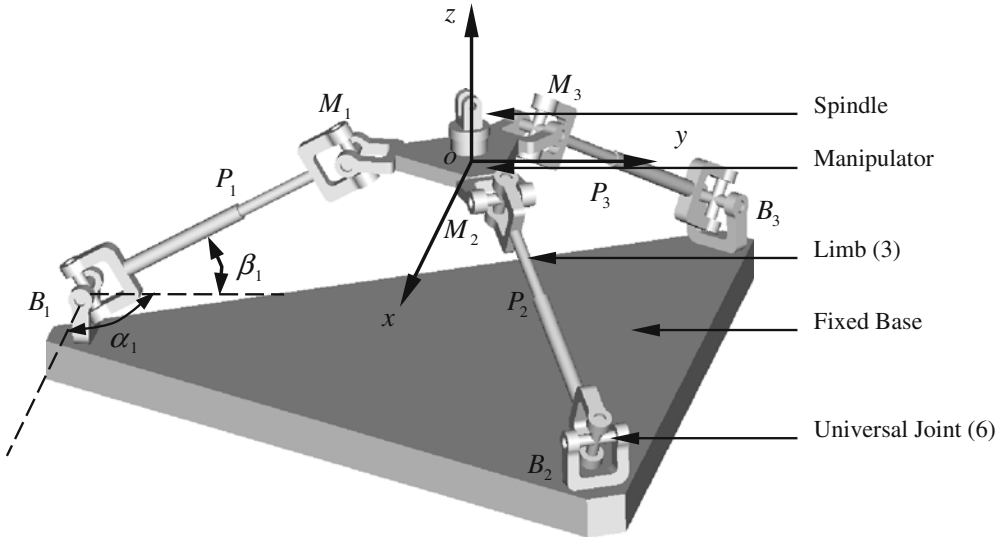

Fig. 4 A spatial parallel mechanism with 3-PUU

$$\mathbf{r}_{B_1}=\begin{bmatrix}0\\-R\\0\end{bmatrix},\mathbf{r}_{B_2}=\begin{bmatrix}\frac{\sqrt{3}}{2}R\\\frac{1}{2}R\\0\end{bmatrix},\mathbf{r}_{B_3}=\begin{bmatrix}-\frac{\sqrt{3}}{2}R\\\frac{1}{2}R\\0\end{bmatrix} \tag{24}$$

So, if the absolute coordinates of point o_c are (x_c y_c z_c), then the vector of point o_c can be denoted as:

$$\mathbf{r}_{o_c}=\begin{bmatrix}x_c\\y_c\\z_c\end{bmatrix} \tag{25}$$

$$\mathbf{r}_{o_cB_1}=\begin{bmatrix}0\\-r\\0\end{bmatrix},\mathbf{r}_{o_cB_2}=\begin{bmatrix}\frac{\sqrt{3}}{2}r\\\frac{1}{2}r\\0\end{bmatrix},\mathbf{r}_{o_cB_3}=\begin{bmatrix}-\frac{\sqrt{3}}{2}r\\\frac{1}{2}r\\0\end{bmatrix}. \tag{26}$$

Therefore:

$$\begin{cases}x_c^2+(y_c-r+R)^2+z_c^2=l_1^2\\\left(x_c+\frac{\sqrt{3}}{2}r-\frac{\sqrt{3}}{2}R\right)^2+\left(y_c+\frac{1}{2}r-\frac{1}{2}R\right)^2+z_c^2=l_2^2\\\left(x_c-\frac{\sqrt{3}}{2}r+\frac{\sqrt{3}}{2}R\right)^2+\left(y_c+\frac{1}{2}r-\frac{1}{2}R\right)^2+z_c^2=l_3^2\end{cases}. \tag{27}$$

The inverse displacement solutions of the manipulator can be obtained:

$$\begin{cases}l_1=\sqrt{x_c^2+(y_c-r+R)^2+z_c^2}\\l_2=\sqrt{\left(x_c+\frac{\sqrt{3}}{2}r-\frac{\sqrt{3}}{2}R\right)^2+\left(y_c+\frac{1}{2}r-\frac{1}{2}R\right)^2+z_c^2}\\l_3=\sqrt{\left(x_c-\frac{\sqrt{3}}{2}r+\frac{\sqrt{3}}{2}R\right)^2+\left(y_c+\frac{1}{2}r-\frac{1}{2}R\right)^2+z_c^2}\end{cases}. \tag{28}$$

To solve the forward displacement of the manipulator, we can presume:

$$X\begin{bmatrix}x_c\\y_c\\z_c\end{bmatrix},G(X)=\begin{bmatrix}x_c^2+(y_c-r+R)^2+z_c^2-l_1^2\\\left(x_c+\frac{\sqrt{3}}{2}r-\frac{\sqrt{3}}{2}R\right)^2+\left(y_c+\frac{1}{2}r-\frac{1}{2}R\right)^2+z_c^2-l_2^2\\\left(x_c-\frac{\sqrt{3}}{2}r+\frac{\sqrt{3}}{2}R\right)^2+\left(y_c+\frac{1}{2}r-\frac{1}{2}R\right)^2+z_c^2-l_3^2\end{bmatrix}.$$

Therefore,

$$J_{3\times3}=G'(X)=2\begin{bmatrix}x_c & y_c-r+R & z_c\\x_c+\frac{\sqrt{3}}{2}r-\frac{\sqrt{3}}{2}R & y_c+\frac{1}{2}r-\frac{1}{2}R & z_c\\x_c-\frac{\sqrt{3}}{2}r+\frac{\sqrt{3}}{2}R & y_c+\frac{1}{2}r-\frac{1}{2}R & z_c\end{bmatrix}. \tag{29}$$

The determinate of $J_{3\times3}$ is:

$$\begin{aligned}|J_{3\times3}|&=8\begin{vmatrix}x_c & y_c-r+R & z_c\\x_c+\frac{\sqrt{3}}{2}r-\frac{\sqrt{3}}{2}R & y_c+\frac{1}{2}r-\frac{1}{2}R & z_c\\x_c-\frac{\sqrt{3}}{2}r+\frac{\sqrt{3}}{2}R & y_c+\frac{1}{2}r-\frac{1}{2}R & z_c\end{vmatrix}\\&=-12\sqrt{3}(R-r)z_c.\end{aligned} \tag{30}$$

Therefore, the singularity criterion of the Newton–Raphson method to solve the forward displacement of the manipulator shown in Fig. 4 are $R=r$ or $z_c=0$, which are also the singular criteria of the machanism. In fact, $R>r$ and $z_c>0$, so the iterative process with Eq. 7 can be executed without any singularity.

Because the manipulator shown in Fig. 4 only has three translational DoFs, we can solve its analytical solutions of the forward displacement problems directly. To obtain the analytical solutions, Eq. 27 can be equivalently transformed into:

$$\begin{cases}x_c^2+(y_c-r+R)^2+z_c^2=l_1^2\\\sqrt{3}x_c+3y_c=\frac{l_1^2-l_2^2}{R-r}\\-\sqrt{3}x_c+3y_c=\frac{l_1^2-l_3^2}{R-r}\end{cases}. \tag{31}$$

Considering $z_c\geq0$, we can obtain the analytical solutions of the forward displacement problem as:

$$\begin{cases}x_c=\frac{l_3^2-l_2^2}{2\sqrt{3}(R-r)}\\y_c=\frac{2l_1^2-l_2^2-l_3^2}{6(R-r)}\\z_c=\sqrt{l_1^2-x_c^2-(y_c-r+R)^2}\end{cases}. \tag{32}$$

Therefore, if the manipulator only has translational DoFs, the forward displacement problems can also be solved with analytical method.

4 Conclusion

In this paper, we investigate a numerical methodology to study the singularity, the forward and inverse displacement of spatial parallel manipulators. The distinctive benefit of the methodology is that the parameter expressions of the manipulator's coordinates are often easily obtained through

coordinates transformation and the whole analysis process can be shortened for those manipulators with identical limbs. Therefore, we can efficiently obtain the forward and inverse displacement of the manipulator with numerical method, and the singularity criteria are the cases when the differential coefficient matrix will be singular. Besides, with examples we demonstrate that the Newton-GMRES algorithm might be a much better remedying method in solving the forward displacement of spatial parallel manipulators when the differential coefficient matrix is nearly singular at the nearby of real solutions. Considering the merits and demerits of the both methods, the combinations of the Newton-GMRES algorithm and Newton-Raphson method should be widely used in engineering applications.

References

1. Šika Z, Kočandrle P, Stejskal V (1998) An investigation of properties of the forward displacement analysis of the generalized Stewart platform by means of general optimization methods. Mech Mach Theory 33(3):245−253
2. Stewart D (1965) A platform with six degrees of freedom. Proc Inst Mech Eng Part I 180(15):371−386
3. Ku D-M (1999) Direct displacement analysis of a Stewart platform mechanism. Mech Mach Theory 34(3):453−465
4. Notash L, Podhorodeski RP (1995) On the forward displacement problem of three-branch parallel manipulators. Mech Mach Theory 30(3):391−404
5. Han L, Liao Q, Liang C (2000) Forward displacement analysis of one kind of general 5–5 parallel manipulators. Mech Mach Theory 35(2):271−289
6. Dhingra AK, Almadi AN, Kohli D (2001) Closed-form displacement analysis of 10-link 1-DOF mechanisms: Part 2-polynomial solutions. Mech Mach Theory 36(1):57−75
7. Kamra R, Kohli D, Dhingra AK (2002) Forward displacement analysis of a six-dof parallel manipulator actuated by 3R3P and 4R2P chains. Mech Mach Theory 37(6):619−637
8. Heath MT (1997) Scientific computing: an introductory survey. McGraw-Hill, New York
9. Zhao J-S, Zhou K, Mao D-Z, Gao Y-F, Fang Y (2004) A new method to study the degree of freedom of spatial parallel mechanisms. Int J Adv Manuf Technol (3−4):288−294
10. Dhingra AK, Almadi AN, Kohli D (2000) Closed-form displacement analysis of 8, 9 and 10-link mechanisms Part I: 8-link 1-DOF mechanisms. Mech Mach Theory 35(6):821−850
11. Kelley CT (1995) Iterative methods for linear and nonlinear equations. North Carolina State University, Society for Industrial and Applied Mathematics, Philadelphia