A numerical method for forward kinematics of general Stewart manipulator using natural coordinates

Ruobing WANG*, Hongtao WU*, Yanli LIU*,**, Xiaolong YANG* and Xuexiang ZHANG*

*College of Mechanical & Electrical Engineering, Nanjing University of Aeronautics & Astronautics
29 Yudao Street, Nanjing 210016, China
E-mail: meeliuyanli@nuaa.edu.cn
**Department of Mechanical Engineering, Jiangsu College of Safety Technology
No.1 University Road, Xuzhou 221011, China

Received: 10 June 2018; Revised: 27 September 2018; Accepted: 12 December 2018

Abstract
In this paper, a numerical method for forward kinematics of general Stewart manipulator using natural coordinates is presented. The kinematic equations are in quadratic forms and the corresponding Jacobian matrix is a linear function of coordinates because of using natural coordinates. According to the characteristics of the kinematic equations, the Newton-Raphson algorithm is simplified to decrease the renewal time of iterations between equations and Jacobian matrix, and used to solve the kinematic equations. The singularity and convergence problems of the algorithm are discussed. Furthermore, the method using natural coordinates is compared with the traditional method using rotation matrix through numerical examples. Comparison results show that the method using natural coordinates is very accurate, more efficient, and has a greater convergence domain.

Keywords: Natural coordinates, Forward kinematics, Stewart manipulator, Newton-Raphson algorithm, Numerical method

1. Introduction

Natural coordinates, also called fully Cartesian coordinates, are introduced as a special method for modeling multi-body system. This method was proposed by García de Jalón et al. in 1986 (Jalón and Bayo, 1994). The position and orientation of a rigid body can be described by the Cartesian coordinates on an inertial reference frame of a number of reference points or reference vectors selected on the rigid body. The Cartesian coordinates of the selected reference points or reference vectors form the natural coordinates of multi-body system. When using natural coordinates to describe the pose of a multi-body system, the number of variables used is not minimal, and often more than the number of degrees of freedom of the system. Therefore, additional constraint equations need to be introduced. It is noteworthy that the constraint equations expressed in natural coordinates are in quadratic forms, and the corresponding Jacobian matrix is a linear function of the coordinates. No transcendental functions related to angular coordinates is required. Therefore, by combining appropriate numerical calculation methods, the numerical calculation efficiency can be greatly improved.

In this paper, we use natural coordinates to solve the forward kinematics of general Stewart manipulator. The Stewart manipulator is a type of 6-DOF spatial parallel manipulator. It is derived from the tire tester of Gough (1957a, 1962b) and the motion simulator of Stewart (1965), and is therefore also known as the Gough-Stewart manipulator. The Stewart manipulator has been widely studied in academic circles and has been widely used in motion simulation systems, medical robots, and force feedback devices due to its high stiffness, high motion accuracy, large output force and output torque, and low inertial (Merlet, 2002). The kinematic analysis of Stewart manipulator is very complicated due to the existence of multiple closed-loop kinematic chains. The kinematics of Stewart manipulator can be divided into inverse kinematics and forward kinematics. The inverse kinematics has become simple due to the existence of analytical solutions. However, the forward kinematics is a problem of dealing with complex nonlinear equations, and
there are multiple solutions. At present, no general form of analytical solution of the forward kinematics has been found. The methods for solving forward kinematics are mainly divided into polynomial method, numerical method, and extra sensors method. The polynomial method (Ji and Wu, 2001; Innocenti, 2001; Huang et al., 2010) is a method of obtaining a univariate polynomial equation by elimination. It can obtain all possible solutions to forward kinematics, but it is very sensitive to numerical errors. The numerical method (Merlet, 1993; Yang et al., 2010; Yang et al., 2017; Puglisi et al., 2017) is to solve the kinematic equations by a numerical iterative algorithm such as Newton-Raphson algorithm. It has high computational efficiency and calculation accuracy. The disadvantage is that different initial values of the iterations may lead to different results. The extra sensors method (Chiu and Perng, 2001; Vertechy and Parenti-Castelli, 2008; Cheng et al., 2017; Liu et al., 2018) can get more information by the auxiliary sensors, thereby reducing the calculation time or even solving the analytical form. But the measurement accuracy of the sensors will affect the accuracy of the calculation results and the economic costs will increase. In addition, methods such as interval analysis (Merlet, 2004), support vector machine (Morell et al., 2012), and neural network (Parikh and Lam, 2009) are also used to solve the forward kinematics of Stewart manipulator.

However, in most of former works, people use the coordinates transformation method to model the kinematic equations of Stewart manipulator. This will inevitably introduce transcendental functions, which will make the kinematic analysis and solution much more complicated. Zhao et al.(2006) used the natural coordinates to analyze the kinematics of the Stewart manipulator with planar vertexes distribution and established a kinematic model containing only quadratic equations. However, this method is not applicable to the Stewart manipulator with spatial vertexes distribution. In this paper, we use natural coordinates to solve the forward kinematics of general Stewart manipulator. We use the natural coordinates to establish a kinematic model that only contains quadratic equations for the general Stewart manipulator with spatial vertexes distribution. According to the characteristics of the kinematic equations, the Newton-Raphson algorithm is simplified to decrease the renewal time of iterations between equations and Jacobian matrix, and used to solve the kinematic equations. The singularity and convergence problems of the algorithm are discussed. Furthermore, the method using natural coordinates is compared with the traditional method using rotation matrix through numerical examples. Comparison results show that the method using natural coordinates is very accurate, more efficient, and has a greater convergence domain.

2. Kinematic equations of general Stewart manipulator using natural coordinates

The structure of general Stewart manipulator is shown in Fig. 1. The moving platform is connected to the fixed base via six UPS kinematic chains. Each kinematic chain is connected to the fixed base at the point \( A_i \) (i=1–6) through a universal joint, and connected to the moving platform at the point \( B_i \) (i=1–6) through a spherical joint. There is a prismatic joint between each \( A_i \) and \( B_i \) to change the leg length \( l_i \) (i=1–6). Static coordinate frame \( O_{xyz} \) and moving coordinate frame \( O_{uw} \) are attached to the fixed base and the moving platform respectively.

The structure of general Stewart manipulator is shown in Fig. 1. The position vectors of points \( A_i \) (i=1–6) and points \( B_i \) (i=1–6) in coordinate frame \( O_{xyz} \) are defined as \( a_i \) (i=1–6) and \( b_i \) (i=1–6) respectively. And the length of each leg is defined as \( l_i \) (i=1–6). In order to describe the pose of the moving platform, we select four reference points on the moving platform named \( P_1 \), \( P_2 \), \( P_3 \) and \( P_4 \), which are not coplanar. Their position vectors in frame \( O_{uw} \) are constant: \( B P_1 = (e_1, f_1, g_1)^T \), \( B P_2 = (e_2, f_2, g_2)^T \), \( B P_3 = (e_3, f_3, g_3)^T \), \( B P_4 = (e_4, f_4, g_4)^T \). The prefixed superscript \( B \)
represents the referenced coordinate frame $O_{uvw}$ . We use the coordinates of the four reference points in frame $O_{xyz}$ to form the natural coordinates $q$ of the Stewart manipulator, they can be expressed as follows:

\[
\begin{align*}
^4P_1 &= (x_1, y_1, z_1)^T \\
^4P_2 &= (x_2, y_2, z_2)^T \\
^4P_3 &= (x_3, y_3, z_3)^T \\
^4P_4 &= (x_4, y_4, z_4)^T \\
q &= (x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3, x_4, y_4, z_4)^T
\end{align*}
\]

(1)

The prefixed superscript $A$ represents the referenced coordinate frame $O_{Axyz}$ . In this paper, we regard the various components of the Stewart manipulator as rigid bodies, so the distance between two reference points on the moving platform is a constant value. Thus the six constraint equations about the rigid body properties can be obtained as follows:

\[
\begin{align*}
&\begin{pmatrix} 2 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 3 \end{pmatrix} - \begin{pmatrix} 1 & 4 \end{pmatrix} - \begin{pmatrix} 2 & 4 \end{pmatrix} = 0 \\
&\begin{pmatrix} 2 & 3 \end{pmatrix} + \begin{pmatrix} 1 & 4 \end{pmatrix} - \begin{pmatrix} 1 & 3 \end{pmatrix} - \begin{pmatrix} 2 & 3 \end{pmatrix} = 0 \\
&\begin{pmatrix} 2 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 3 \end{pmatrix} - \begin{pmatrix} 1 & 4 \end{pmatrix} - \begin{pmatrix} 2 & 4 \end{pmatrix} = 0 \\
&\begin{pmatrix} 3 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 3 \end{pmatrix} - \begin{pmatrix} 3 & 4 \end{pmatrix} = 0 \\
&\begin{pmatrix} 3 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 4 \end{pmatrix} - \begin{pmatrix} 3 & 4 \end{pmatrix} = 0
\end{align*}
\]

(2)

Since the reference points $P_1, P_2, P_3$ and $P_4$ are not coplanar, the coordinates of any point $P_m$ on the moving platform can be written as a linear combination of the coordinates of the reference points. The positional relationship between point $P_m$ and the reference points is shown in Fig. 2. We define:

\[
\begin{align*}
^4P_m &= (x_m, y_m, z_m)^T \\
r_{m1} &= ^4P_m - ^4P_1 = (x_m - x_1, y_m - y_1, z_m - z_1)^T \\
r_{m2} &= ^4P_m - ^4P_2 = (x_m - x_2, y_m - y_2, z_m - z_2)^T \\
r_{m3} &= ^4P_m - ^4P_3 = (x_m - x_3, y_m - y_3, z_m - z_3)^T \\
r_{m4} &= ^4P_m - ^4P_4 = (x_m - x_4, y_m - y_4, z_m - z_4)^T
\end{align*}
\]

(3)

Since the reference points are not coplanar, the vectors $r_{m2}, r_{m3}$ and $r_{m4}$ are linearly independent and constitute a set of bases in a three-dimensional space. So there exist constants $c_{m1}, c_{m2}$ and $c_{m3}$ such that:
\[ \mathbf{n}_m = c_{m1}\mathbf{n}_2 + c_{m2}\mathbf{n}_3 + c_{m3}\mathbf{n}_4 \]  

We should know that \( c_{m1}, c_{m2} \) and \( c_{m3} \) are only determined by the relative positions of the reference points, and have nothing to do with the pose of the moving platform. From Eq. (3) and Eq. (4) we can get:

\[ A\mathbf{P}_m = (1-c_{m1}-c_{m2}-c_{m3})A\mathbf{P}_1 + c_{m1}A\mathbf{P}_2 + c_{m2}A\mathbf{P}_3 + c_{m3}A\mathbf{P}_4 \]  

If we make \( k_{m1} = (1-c_{m1}-c_{m2}-c_{m3}) \), \( k_{m2} = c_{m1} \), \( k_{m3} = c_{m2} \) and \( k_{m4} = c_{m3} \), according to Eq. (5) we can express the position vectors \( \mathbf{b}_i \) (i=1~6) of points \( B_i \) (i=1~6) in frame \( O_{xyz} \) as follows:

\[ \mathbf{b}_i = \begin{pmatrix} k_{i1}x_i + k_{i2}x_2 + k_{i3}x_3 + k_{i4}x_4 \\ k_{i1}y_i + k_{i2}y_2 + k_{i3}y_3 + k_{i4}y_4 \\ k_{i1}z_i + k_{i2}z_2 + k_{i3}z_3 + k_{i4}z_4 \end{pmatrix} \]

where \( k_{ij} \) (i=1~6, j=1~4) are determined only by the geometry of the moving platform. Assume that \( \mathbf{a}_i = (a_{ix}, a_{iy}, a_{iz})^T \) (i=1~6) in frame \( O_{xyz} \), then according to the characteristics of the UPS kinematic chains, six constraint equations can be expressed as follows:

\[
\begin{align*}
(k_{11}x_1 + k_{12}x_2 + k_{13}x_3 + k_{14}x_4 - a_{ix})^2 + (k_{11}y_1 + k_{12}y_2 + k_{13}y_3 + k_{14}y_4 - a_{iy})^2 &= 0 \\
(k_{11}z_1 + k_{12}z_2 + k_{13}z_3 + k_{14}z_4 - a_{iz})^2 - l_1^2 &= 0 \\
(k_{21}x_1 + k_{22}x_2 + k_{23}x_3 + k_{24}x_4 - a_{ix})^2 + (k_{21}y_1 + k_{22}y_2 + k_{23}y_3 + k_{24}y_4 - a_{iy})^2 &= 0 \\
(k_{21}z_1 + k_{22}z_2 + k_{23}z_3 + k_{24}z_4 - a_{iz})^2 - l_2^2 &= 0 \\
(k_{31}x_1 + k_{32}x_2 + k_{33}x_3 + k_{34}x_4 - a_{ix})^2 + (k_{31}y_1 + k_{32}y_2 + k_{33}y_3 + k_{34}y_4 - a_{iy})^2 &= 0 \\
(k_{31}z_1 + k_{32}z_2 + k_{33}z_3 + k_{34}z_4 - a_{iz})^2 - l_3^2 &= 0 \\
(k_{41}x_1 + k_{42}x_2 + k_{43}x_3 + k_{44}x_4 - a_{ix})^2 + (k_{41}y_1 + k_{42}y_2 + k_{43}y_3 + k_{44}y_4 - a_{iy})^2 &= 0 \\
(k_{41}z_1 + k_{42}z_2 + k_{43}z_3 + k_{44}z_4 - a_{iz})^2 - l_4^2 &= 0 \\
(k_{51}x_1 + k_{52}x_2 + k_{53}x_3 + k_{54}x_4 - a_{ix})^2 + (k_{51}y_1 + k_{52}y_2 + k_{53}y_3 + k_{54}y_4 - a_{iy})^2 &= 0 \\
(k_{51}z_1 + k_{52}z_2 + k_{53}z_3 + k_{54}z_4 - a_{iz})^2 - l_5^2 &= 0 \\
(k_{61}x_1 + k_{62}x_2 + k_{63}x_3 + k_{64}x_4 - a_{ix})^2 + (k_{61}y_1 + k_{62}y_2 + k_{63}y_3 + k_{64}y_4 - a_{iy})^2 &= 0 \\
(k_{61}z_1 + k_{62}z_2 + k_{63}z_3 + k_{64}z_4 - a_{iz})^2 - l_6^2 &= 0 
\end{align*}
\]

The system of Eq. (2) together with the system of Eq. (7) constitute the kinematic model of Stewart manipulator using natural coordinates \( \mathbf{q} \). We can see that due to the use of natural coordinates, the constraint equations in kinematic model are all converted to quadratic forms.

3. Numerical algorithm for forward kinematic equations

The main problem of forward kinematics of Stewart manipulator is to solve nonlinear equations. Newton-Raphson algorithm is a method to solve nonlinear equations by numerical iteration, and is widely used in the real-time control because it is relatively easy to implement and converges quickly. However, with the emergence of more and more high-frequency control systems, higher requirements are imposed on the calculation time of forward kinematics, and the computational efficiency of Newton-Raphson algorithm has been challenged. Therefore, how to improve the computational efficiency of Newton-Raphson algorithm has become a key issue in the current research of forward kinematics. Because the kinematics model of the Stewart manipulator described by the natural coordinates is a set of quadratic equations, we can take advantage of this property to simplify the calculation process of Newton-Raphson algorithm to decrease the renewal time of iterations between equations and Jacobian matrix, thereby improving the computational efficiency.
3.1 Simplification of the Newton-Raphson algorithm

We expand and simplify Eq. (2) and Eq. (7) to obtain a set of quadratic equations as follows:

\[
F(q) = \begin{pmatrix}
\frac{1}{2}q^TQ_iq + S_i^Tq + C_i \\
\frac{1}{2}q^TQ_jq + S_j^Tq + C_j \\
\vdots \\
\frac{1}{2}q^TQ_{12}q + S_{12}^Tq + C_{12}
\end{pmatrix} = 0
\tag{8}
\]

where \( q \) is the natural coordinates of the Stewart manipulator, \( Q_i \) (i=1~12) are the 12-dimensional symmetric matrices determined by the geometric parameters of the Stewart manipulator, \( S_i \) (i=1~12) are the 12-dimensional vectors determined by the geometric parameters of the Stewart manipulator, and \( C_i \) (i=1~12) are the constants determined by the geometric parameters and leg lengths of the Stewart manipulator. \( Q_i, S_i \) and \( C_i \) are all invariant in the process of solving forward kinematics. They can be calculated by the following formula:

\[
Q_i = \begin{pmatrix}
2I_{3x3} & -2I_{3x3} & 0_{3x3} & 0_{3x3} \\
-2I_{3x3} & 2I_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3}
\end{pmatrix}
\]

\[
Q_j = \begin{pmatrix}
2I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 2I_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3}
\end{pmatrix}
\]

\[
Q_k = \begin{pmatrix}
2I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3}
\end{pmatrix}
\]

\[
Q_l = \begin{pmatrix}
2I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\
0_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3}
\end{pmatrix}
\]

\[
S_i = -2[k_{i1}^T, k_{i2}^T, k_{i3}^T, k_{i4}^T] \quad i = 1 \sim 6
\]

\[
S_j = \theta_{6x1} \quad i = 7 \sim 12
\]

where \( I_{nxn} \) is an n-dimensional unit matrix and \( \theta_{nx1} \) is an n-dimensional zero matrix. We calculate the Jacobian matrix of the system of Eq. (8) to obtain:

\[
J(q) = \frac{\partial F(q)}{\partial q} = (Q_1q + S_1, Q_2q + S_2, ..., Q_{12}q + S_{12})^T
\tag{10}
\]
Since the Jacobian matrix is a linear function of the natural coordinates $q$, we define:

\[ Q(q) = (Q_1 q, Q_2 q, \ldots, Q_{12} q)^T \]

\[ S = (S_1, S_2, \ldots, S_{12})^T \]

\[ C = (C_1, C_2, \ldots, C_{12})^T \]  \hspace{1cm} (11)

Bringing Eq. (11) into Eq. (8) and Eq. (10), we have:

\[ J(q) = Q(q) + S \]

\[ F(q) = \frac{1}{2} Q(q)q + Sq + C \]  \hspace{1cm} (12)

We assume that $q_r$ is a real root of the system $F(q) = \theta$ and $q_0$ is an estimate of $q_r$. Let's expand the first-order Taylor series of $F(q)$ at $q_0$:

\[ F(q) = F(q_0) + J(q_0)(q - q_0) \]  \hspace{1cm} (13)

Substituting $q = q_r$ into the above formula, when the system's Jacobian matrix $J(q)$ is non-singular, we can get:

\[ q_r = q_0 - J(q_0)^{-1} F(q_0) \]  \hspace{1cm} (14)

From this we get the iterative formula of the Newton-Raphson algorithm:

\[ q_{k+1} = q_k - J(q_k)^{-1} F(q_k) \]  \hspace{1cm} (15)

where $k$ represents the iteration order. Bring Eq. (12) into the above formula, we can get the simplified iterative formula:

\[ q_{k+1} = \frac{1}{2} q_k - (Q(q_k) + S)^{-1} (\frac{1}{2} Sq_k + C) \]  \hspace{1cm} (16)

In this way, during each iteration, only $Q(q_k)$ is calculated and $F(q_k)$ and $J(q_k)$ need not be calculated, thereby decreasing the renewal time of iterations between equations and Jacobian matrix. So the computational efficiency is improved due to the simplification of the calculation process.

3.2 Singularity and convergence discussion

Since the Newton-Raphson algorithm is only valid when the Jacobian matrix is non-singular, and the iterative algorithm may not converge or converge to the point far away from the real pose when the Jacobian matrix is close to singular (Burden and Faires, 1988), so it is very necessary to analyze the singularity of $J(q)$. We can get the time derivative of Eq. (7) as follows:

\[ J_l I + J_q q = 0 \]  \hspace{1cm} (17)

where $l = (l_1, l_2, \ldots, l_6)^T$, $J_l$ is called inverse-kinematics Jacobian matrix, and $J_q$ is called forward-kinematics Jacobian matrix. When $J_l$ is singular, the serial manipulator is in a singular configuration. When $J_q$ is singular, the parallel manipulator is in a singular configuration (Gosselin and Angeles, 2002). We derive the Eq. (2) and define the Jacobian matrix of Eq. (2) as $\Phi_q$: 

\[
\Phi_q \hat{q} = 0
\]
\[
\Phi_q = 2 \begin{pmatrix}
-\mathbf{r}_2^T & \mathbf{r}_2^T & \mathbf{0}_{6 \times 3} & \mathbf{0}_{6 \times 3} \\
-\mathbf{r}_3^T & \mathbf{0}_{6 \times 3} & \mathbf{r}_3^T & \mathbf{0}_{6 \times 3} \\
-\mathbf{r}_4^T & \mathbf{0}_{6 \times 3} & \mathbf{0}_{6 \times 3} & \mathbf{r}_4^T \\
\mathbf{0}_{6 \times 3} & -\mathbf{r}_2^T & \mathbf{r}_2^T & \mathbf{0}_{6 \times 3} \\
\mathbf{0}_{6 \times 3} & \mathbf{0}_{6 \times 3} & -\mathbf{r}_4^T & \mathbf{r}_4^T \\
\mathbf{0}_{6 \times 3} & \mathbf{0}_{6 \times 3} & \mathbf{0}_{6 \times 3} & -\mathbf{r}_4^T
\end{pmatrix}
\]

(18)

where \( \mathbf{r}_i = (x_i - x_i, y_i - y_i, z_i - z_i)^T \) \( (i \neq j \quad i, j = 1 \sim 4) \). We have found that because the reference points are not coplanar, the matrix \( \Phi_q \) is always of full row rank. Due to:

\[
J(q) = \begin{pmatrix}
\mathbf{J}_q \\
\Phi_q
\end{pmatrix}
\]

(19)

So the matrix \( J(q) \) is singular if and only if the matrix \( \Phi_q \) is singular. Therefore, we can conclude that the algorithm we use is always valid when the parallel manipulator is not in a singular configuration.

We know that for a given set of leg lengths, Stewart manipulator may have multiple sets of poses to correspond with. When using the Newton-Raphson algorithm to solve forward kinematics, different initial values of iterations may result in different poses, and the results may not converge when the initial value of the iteration is far from the real pose. According to the local convergence theorem (Burden and Faires, 1988), for some pose \( q_r \), there exist a neighborhood \( U = \{ q \mid || q - q_r || \leq \delta \} \) of \( q_r \), where using any \( q_0 \in U \) as the initial values of the iteration always allow the iterative results to converge to \( q_r \). We call \( U \) as the convergence domain of the Newton-Raphson algorithm and \( \delta \) as the convergence radius. When using the Newton-Raphson algorithm to solve forward kinematics, we need to guarantee \( || q_0 - q_r || \leq \delta \), so as to ensure that the calculation results converge to the real pose of Stewart manipulator. Through numerical experiments, we find that the Newton-Raphson iteration using natural coordinates can converge in a larger range than other methods, which allows us to select the initial value of the iteration in a larger range. This will be demonstrated by numerical examples in the following text.

4. Comparison with rotation matrix method through numerical examples

In order to verify the accuracy and efficiency of the method for forward kinematics we use, we compare the natural coordinates method with the traditional rotation matrix method through numerical examples. The results show that the method using natural coordinates is very accurate, more efficient, and has a greater convergence domain.

4.1 The rotation matrix method

The rotation matrix method uses three position parameters and three angle parameters to describe the pose of the moving platform. The coordinates of the system can be expressed as six independent variables:

\[
\theta = (p^T, \Psi^T)^T = (x, y, z, \alpha, \beta, \gamma)^T
\]

(20)

We define \( \theta \) as the independent coordinates of the system, where vector \( p = (x, y, z)^T \) represents the position of the moving platform, and vector \( \Psi = (\alpha, \beta, \gamma)^T \) represents the Euler angle of the moving platform. Assume that the rotation order is z-y-x, then the rotation matrix \( R \) is defined as follows:

\[
R = \begin{pmatrix}
c \beta \gamma & c \gamma \sin \beta - c \alpha \gamma & c \alpha \beta + s \alpha \gamma \\
c \beta \sin \gamma + s \alpha \beta \gamma & -c \gamma \sin \beta + c \alpha \gamma \\
-s \beta & c \beta \alpha & c \alpha \beta
\end{pmatrix}
\]

(21)

where \( s \) and \( c \) denote sine and cosine respectively. The kinematic equations of Stewart manipulator using rotation matrix can be expressed as follows:
\[ \| p + R^b b_i - a_i \| \|^2 - l_i^2 = 0 \quad i = 1 \sim 6 \]  

(22)

where \( a_i \) denote the position vectors of points \( A_i \) in frame \( O_{1-xyz} \), \( b_i \) denote the position vectors of points \( B_i \) in frame \( O_{2-uvw} \). We can find that the kinematic model described by the rotation matrix method has fewer equations, but there are transcendental functions in the equations. The kinematic equations using rotation matrix will be solved by the Newton-Raphson algorithm for forward kinematics.

It should be noted that both the independent coordinates \( \theta \) and the natural coordinates \( q \) are the representations of the poses of Stewart manipulator, so they are equivalent when representing the same pose. In order to facilitate the comparison between the natural coordinates method and the rotation matrix method in the following text, we introduce the conversion relationship with the following:

\[
q = \begin{pmatrix}
p + R^b p_1 \\
p + R^b p_2 \\
p + R^b p_3 \\
p + R^b p_4 
\end{pmatrix}
\]

(23)

4.2 Numerical examples

In the following we compare the two methods through numerical examples. We fix the real pose of Stewart manipulator and use a huge number of different initial values to solve the forward kinematics. The accuracy and efficiency of the two methods are compared. At the same time, because these initial values can not always make the calculation results of forward kinematics converge to the real pose, we compare the convergence ratio and convergence radius of the two methods. In this article, the length units are all m, and the angle units are all rad.

Assuming there is a general Stewart manipulator, the coordinate matrices of points \( A_i \) and points \( B_i \) in their respective frames are as follows:

\[
A_{1-6} = \begin{pmatrix}
2.5219 & 2.5219 & -0.9362 & -1.5857 & -1.5857 & -0.9362 \\
-0.3750 & 0.3750 & 2.3715 & 1.9965 & -1.9965 & -2.3715 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

(24)

\[
B_{1-6} = \begin{pmatrix}
0.7934 & 0.7934 & 0.4675 & -1.2608 & -1.2608 & 0.4675 \\
-0.9978 & 0.9978 & 1.1860 & 0.1882 & -0.1882 & -1.1860 \\
-0.0900 & 0 & -0.0900 & 0 & -0.0900 & 0
\end{pmatrix}
\]

(25)

The reference points \( P_1, P_2, P_3 \) and \( P_4 \) are selected as follows:

\[
\begin{aligned}
&^B p_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\
&^B p_2 = \begin{pmatrix} 0.7933 \\ -0.9978 \\ -0.0900 \end{pmatrix} \\
&^B p_3 = \begin{pmatrix} 0.4675 \\ 1.1860 \\ -0.0900 \end{pmatrix} \\
&^B p_4 = \begin{pmatrix} -1.2608 \\ -0.1882 \\ -0.0900 \end{pmatrix}
\end{aligned}
\]

(26)

According to Eq. (5), Eq. (6) and Eq. (25), the matrix \( K \) can be calculated as follows:

\[
K = \begin{pmatrix}
0 & 1 & 0 & 0 \\
1 & -0.1502 & 0.6376 & -0.4874 \\
0 & 0 & 1 & 0 \\
1 & -0.4874 & -0.1502 & 0.6376 \\
0 & 0 & 0 & 1 \\
1 & 0.6376 & -0.4874 & -0.1502
\end{pmatrix}
\]

(27)
where the elements of $K$ are the $k_{ij}$ in Eq. (7). Assume that the real pose of the moving platform is:

$$
\theta_r = (0, 0, 0.312, 0, 0, 0)^T \tag{28}
$$

$$
q_r = (0, 0, 3.312, 0.7934, -0.9978, 3.222, 0.4675, 1.186, 3.222, -1.2608, -0.1882, 3.222)^T \tag{29}
$$

The leg lengths corresponding to the real pose are:

$$
l_r = (3.709, 3.7875, 3.709, 3.7875, 3.709, 3.7875)^T \tag{30}
$$

Now we take $\theta_r$ as the center and draw the area $D$ in the pose space defined by the independent coordinates as follows:

$$
D = \{ (x, y, z, \alpha, \beta, \gamma)^T \mid x, y \in (-1, 1), z \in (2.313, 4.312), \alpha, \beta, \gamma \in (-0.524, 0.524) \} \tag{31}
$$

We select points as $\theta_0$ in the area $D$ according to the following rules:

$$
\theta_0 = \left( \begin{array}{c}
-1 + 0.2i_1 \\
-1 + 0.2i_2 \\
2.313 + 0.2i_3 \\
-0.524 + 0.1048i_4 \\
-0.524 + 0.1048i_5 \\
-0.524 + 0.1048i_6 \\
\end{array} \right)
\quad i_1, i_2, i_3, i_4, i_5, i_6 = 0, 1, \ldots, 10
\tag{32}
$$

Under these rules, $\theta_0$ has a total of 1771561 different values. We take the pose corresponding to each $\theta_0$ as the initial estimate of real pose, and use the natural coordinates method and rotation matrix method to solve the real pose respectively. The natural coordinates $q_0$ corresponding to $\theta_0$ can be calculated by Eq. (23). When using natural coordinates, the initial value of iteration is set as $q_0$, the leg lengths are set as $l_r$, and the termination condition of iteration is set as $|| \Delta q || \leq 10^{-9}$. When using rotation matrix, the initial value of iteration is set as $\theta_0$, the leg lengths are set as $l_r$, and the termination condition of iteration is set as $|| \Delta \theta || \leq 10^{-9}$. The results of both methods are converted into independent coordinates and compared with $\theta_r$ to determine the solution errors. Assume that one of the results of the calculation is $\theta_n$, and if $|| \theta_n - \theta_r || \leq 10^{-9}$, then the result can be considered to converge to $\theta_r$. The initial value resulting in the non-convergence of the calculation result is denoted as $\theta^*$, then we can define the convergence radius as $\delta = \min(|| \theta_0 - \theta_r ||)$.

The numerical calculation is performed under the condition of Win 8.1/ Intel Core i5 2.4GHz/ 8GB RAM/ Matlab R2016b. The accuracy and efficiency of the two methods are compared in Table 1, and the convergence of the two methods are compared in Table 2. It should be noticed that when comparing the accuracy and efficiency of the two methods, we only consider those cases where the calculation results of the two methods both converge to the real pose.

| Table 1: Accuracy and efficiency comparison between natural coordinates method and rotation matrix method |
|--------------------------------------------------|-------------------------------|-------------------|-------------------|------------------|------------------|------------------|
| Method                                           | Mean number of iterations     | Mean calculation  | $x[10^{16}m]$    | $y[10^{16}m]$    | $z[10^{16}m]$    | $\alpha[10^{-16}rad]$ | $\beta[10^{-16}rad]$ | $\gamma[10^{-16}rad]$ |
| Natural coordinates method                      | 5.946                         | 0.457             | 4.249            | 4.156            | 6.069            | 1.875            | 1.482            | 7.729             |
| Rotation matrix method                           | 5.629                         | 1.519             | 2.239            | 3.766            | 0.505            | 1.305            | 1.027            | 1.291             |
4.3 Results analysis

According to Table 1, we found that the natural coordinates method and rotation matrix method converge after almost the same number of iterations, and the solution errors are within the allowable range, indicating that both methods are accurate. The mean calculation time of the natural coordinates method is only 0.457 ms, while that of the rotation matrix method is 1.519 ms, which is 3.324 times longer than the former. This shows that the natural coordinates method has a great advantage in computational efficiency. It can be seen from the iterative formula of Newton-Raphson algorithm that the complexity of the Jacobian matrix has a great influence on the computational efficiency. Although the Jacobian matrix of the natural coordinates method is 12-dimensional, which is much larger than the 6-dimensional Jacobian matrix of the traditional rotation matrix method, the kinematic equations are in quadratic forms and the corresponding Jacobian matrix is a linear function of coordinates because of using natural coordinates. Based on this character, we simplify the iterative formula of Newton-Raphson algorithm to Eq. (16), which can greatly reduce the time consumed by the updating of the equations and the Jacobian matrices. However, due to the existence of the transcendental functions, the rotation matrix method takes more time to calculate highly nonlinear equations and Jacobian matrices for each iteration, so the calculation time is much longer than that of natural coordinates method.

It can be seen from Table 2 that the natural coordinates method has a larger convergence domain, and the convergence radius is 1.54 times larger than the convergence radius of the rotation matrix method. The larger convergence domain allows us to select the initial values of iteration that lead to the convergence of the calculation results in a wider range. At present, there is no mathematical theorem that can be used to directly solve the convergence domain of Newton-Raphson algorithm or compare the iterative convergence domains of two different functions. According to Kantorovich's theorem (Tapia, 1971), we can solve the lower bound of the convergence radius of Newton-Raphson algorithm for a certain zero point of the function, but numerical experiments show that the Newton-Raphson algorithm can always converge in a larger range. The Newton-Raphson iterative formula is derived based on the local linearization of the nonlinear equations, so the degree of nonlinearity of the equations to be solved has a great impact on the convergence of the iterative algorithm. Therefore based on the results of numerical examples, we assume that the more the equations to be solved are like a linear equations, the more easily the Newton-Raphson algorithm converges. The kinematics equations of Stewart manipulator are in quadratic forms because of using natural coordinates. We can consider the quadratic form as the simplest nonlinearity, so the kinematic equations represented by natural coordinates have a low degree of nonlinearity, which makes the Newton-Raphson algorithm have a large convergence domain. In contrast, the kinematic equations established by the rotation matrix method have a higher degree of nonlinearity due to the existence of the transcendental functions, so the convergence domain is smaller.

It should be noticed that in the numerical examples given in this paper, all forward kinematics calculations are to solve the same real poses, that is, the nonlinear equations to be solved do not change, but the initial values of the iterations are different. We know that the change of real pose will affect the results in Tables 1 and 2. Therefore we choose different real poses to repeat the above numerical experiments. The results show that the specific data of Table 1 and Table 2 may change, but the conclusion keeps the same, which is the calculation time of the natural coordinates method is shorter and the convergence radius is larger than that of the rotation matrix method.

5. Conclusion
(1) This paper studies a numerical method to solve the forward kinematics of Stewart manipulator. Twelve natural coordinates are used to describe the position and orientation of the Stewart manipulator. Six constraint equations about rigid body properties and six constraint equations about kinematic chains constitute the kinematic model of the Stewart manipulator. These equations are all in quadratic forms.

(2) The Newton-Raphson algorithm is simplified according to the nature of the kinematic equations and is used to solve the kinematic equations. The singularity and convergence of the algorithm are discussed. It is proved that the algorithm we use is always effective in the non-singular workspace of Stewart manipulator.

(3) The method we use is verified by comparing it with the rotation matrix method. Numerical examples show that the calculation time of the natural coordinates method is 3.324 times shorter and the convergence radius is 1.54 times larger than that of the rotation matrix method.

Acknowledgment

This research is supported by the National Key Research and Development Program of China (Grant No. 2018YFC0309102 and 2018YFC0309103), the National Natural Science Foundation of China (Grant No. 51375230) and the Natural Science Foundation of Jiangsu Higher Education Institutions of China (Grant No. 17KJB460003). The authors gratefully acknowledge these support agencies.

References

Burden, Richard L., and J. D. Faires, Numerical analysis: 4th ed (1988), pp.31-36, PWS Publishing Company.
Cheng, Shili, Y. Liu, and W. U. Hongtao, A new approach for the forward kinematics of nearly general Stewart platform with an extra sensor, Journal of Advanced Mechanical Design Systems & Manufacturing, Vol.11, No.2 (2017), JAMDSM0032-JAMDSM0032.
Chiu, Yu Jen, and M. H. Perng, Forward Kinematics of a General Fully Parallel Manipulator with Auxiliary Sensors, International Journal of Robotics Research, Vol.20, No.5 (2001), pp.401-414.
D. Stewart, A Platform with Six Degrees of Freedom, Proc. instn. mech. engrs, Vol.180, No.1 (1965), pp.371-386.
Gosselin, Clement, and J. Angeles, Singularity analysis of closed-loop kinematic chains, IEEE Transactions on Robotics & Automation, Vol.6, No.3 (2002), pp.281-290.
Gough, V. E., and S. G. Whitehall, Universal Tyre test Machine, Proc. FISITA 9th Int. Technical Congr. (1962), pp.117-137.
Gough, V. E., Contribution to discussion to papers on research in automobile stability and control and in type performance, Pro Auto Device Instruction Mech. Engineering, Vol.171 (1957), pp.392-395.
Huang, Xiguang, Q. Liao, and S. Wei, Closed-form forward kinematics for a symmetrical 6-6 Stewart platform using algebraic elimination, Mechanism & Machine Theory, Vol.45, No.2 (2010), pp.327-334.
Innocenti, K., Forward Kinematics in Polynomial Form of the General Stewart Platform, Journal of Mechanical Design, Vol.123, No.2 (2001), pp.254-260.
Jalón, Javier García De, and E. Bayo, Kinematic and Dynamic Simulation of Multibody Systems (1994), pp.44-63, Springer New York.
Ji, Ping, and H. Wu, A closed-form forward kinematics solution for the 6-6P Stewart platform, IEEE Transactions on Robotics & Automation, Vol.17, No.4 (2001), pp.522-526.
Merlet, J. P., Direct kinematics of parallel manipulators, IEEE Transactions on Robotics & Automation, Vol.9, No.6 (1993), pp.842-846.
Merlet, J. P., Parallel Robots (2006), pp.62-93, Springer Netherlands.
Merlet, J. P., Solving the forward kinematics of Gough-type parallel manipulator with interval analysis, International Journal of Robotics Research, Vol.23, No.3 (2004), pp.221-236.
Morell, A., Acosta, L., and Toledo, J., An artificial intelligence approach to forward kinematics of Stewart platforms, Proceedings of 2012 20th Mediterranean Conference on Control and Automation (2012), pp.433-438.
Parikh, Pratik J., and S. S. Lam, Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy, International Journal of Advanced Manufacturing Technology, Vol.40, No.5-6
Puglisi, L. J., Saltaren, R., Garcia, C., Cardenas, P., and Moreno, H., Implementation of a generic constraint function to solve the direct kinematics of parallel manipulators using Newton-Raphson approach, Control Engineering & Applied Informatics, Vol.19, No.2 (2017), pp.71-79.

Tapia, R. A., The Kantorovich Theorem for Newton's Method, American Mathematical Monthly, Vol.78, No.4 (1971), pp.389-392.

Vertechy, Rocco, and V. Parenti-Castelli, Robust, Fast and Accurate Solution of the Direct Position Analysis of Parallel Manipulators by Using Extra-Sensors, Parallel Manipulators, towards New Applications. InTech (2008), pp.133-154.

Yang, C. F., Zheng, S. T., Jin, J., Zhu, S. B., and Han, J. W., Forward kinematics analysis of parallel manipulator using modified global Newton-Raphson method, Journal of Central South University, Vol.17, No.6 (2010), pp.1264-1270.

Yang, X., Wu, H., Li, Y., and Chen, B., A dual quaternion solution to the forward kinematics of a class of six-DOF parallel robots with full or reductant actuation, Mechanism & Machine Theory, Vol.107 (2017), pp.27-36.

Yanli, L., Shili, C., Surong, J., Xiaolong, Y., Yao, L., and Hongtao, W., Forward kinematics of 6-UPS parallel manipulators with one displacement sensor, Journal of Mechanical Engineering, Vol.54, No.5 (2018), pp.1-7 (in Chinese).

Zhao, J. S., Yun, Y., Wang, L. P., Wang, J. S., and Dong, J. X., Investigation of the forward kinematics of the Gough-Stewart manipulator with natural coordinates, International Journal of Advanced Manufacturing Technology, Vol.30, No.7-8 (2006), pp.700-716.