A Mixed Real-time Algorithm for the Forward Kinematics of Stewart Parallel Manipulator

WANG Sun-an, WAN Ya-min

Key Laboratory of Education Ministry for Modern Design and Rotor-Bearing System, Xi’an Jiaotong University Xi’an 710049 China

Abstract Aimed at the real-time forward kinematics solving problem of Stewart parallel manipulator in the control course, a mixed algorithm combining immune evolutionary algorithm and numerical iterative scheme is proposed. Firstly taking advantage of simpleness of inverse kinematics, the forward kinematics is transformed to an optimal problem. Immune evolutionary algorithm is employed to find approximate solution of this optimal problem in manipulator’s workspace. Then using above solution as iterative initialization, a speedy numerical iterative scheme is proposed to get more precise solution. In the manipulator running course, the iteration initialization can be selected as the last period position and orientation. Because the initialization is closed to correct solution, solving precision is high and speed is rapid enough to satisfy real-time requirement. This mixed forward kinematics algorithm is applied to real Stewart parallel manipulator in the real-time control course. The examination result shows that the algorithm is very efficient and practical.

Key words stewart parallel manipulator; forward kinematics; immune evolutionary algorithm; numerical iterative scheme; real-time control

Compared with the traditional serial manipulators, parallel manipulators have the advantages of high precision positioning, simple inverse kinematics solution, high structural rigidity and strong carrying capability, etc. As the most noted parallel manipulator, Stewart platform manipulator is a hot spot of manipulators study in recent years[1]. It has been used as automobile or flight simulator, the movable platform of large spherical radio telescope, virtual machine tools, etc.[2].

The inverse kinematics analysis of parallel manipulators finds the actuator displacements from the given position and orientation of the movable platform. The solution is unique, and can be simply determined. However, the forward kinematics analysis, which determines the position and orientation of the movable platform for the given actuator displacements, is generally very complicated. The problem involves system of high order non-linear equations.

The analytical method has been employed to solve the forward kinematics problem of parallel manipulators[3]. This method reduces the non-linear equations to one polynomial equation in one variable, from which other variables can be solved in closed form. This method can get all solutions, but it is complicated and abstruse. Reducing variables and solving polynomial equation need large amounts of computation.

There are various numerical schemes to deal with the forward kinematics problem[4]. The numerical schemes can be used to all parallel manipulators and are simple relatively. But the general numerical schemes can not get all solutions, and the solution is correlative to the iterative initialization.

Furthermore, the continuation method and neural network method are adopted to solve the problem[5-6]. But the precision or the solving speed of those methods are not satisfactory.

All the methods above are difficult to be employed in the real-time control course of parallel manipulators. The aim of this paper is to present a novel mixed method of the forward kinematics when Stewart parallel manipulator is running.

1 System Statement

A general Stewart parallel manipulator is shown in Fig.1. $B_1B_2B_3B_4B_5B_6$ is base, and $b_1b_2b_3b_4b_5b_6$ is movable platform. $L_1-L_6$ are extensible actuators, they connect base with movable platform by spherical joint.

The coordinate system $O-XYZ$ and $P-X'Y'Z'$ are
fixed to base and movable platform respectively. The position and orientation of movable platform are described by point P’s coordinate \((x_P, y_P, z_P)\) and coordinate system \(P-X'Y'Z'\)’s rotational angles \((\gamma, \beta, \alpha)\) relating to \(O-XYZ\).

We define \(q = (x_P, y_P, z_P, \gamma, \beta, \alpha)^T\).

1.1 Inverse Kinematics

The inverse kinematics calculates the actuators’ length from the given position and orientation of the movable platform. In this instance, we know the vectors \(OP, OB_i, Pb_i\), and orientation matrix \(T\). Then we can calculate the vector \(Ob_i\) by the equation

\[
Ob_i = OP + T \cdot Pb_i
\]

where

\[
T = \begin{bmatrix}
c \alpha \beta & \cos \beta \gamma & -s \beta \gamma \\
-s \alpha \beta & s \alpha \gamma & c \alpha \gamma \\
0 & c \gamma & s \gamma
\end{bmatrix}
\]

where \(c\) represents “cos”, \(s\) represents “sin”. Thus the actuators’ length \(L_i\) can be computed by

\[
L_i = |Ob_i - OB|
\]

1.2 Jacobian Matrix

Conventionally the manipulator’s Jacobian matrix is defined as a matrix relating actuators’ velocities to Cartesian velocities. For parallel manipulator, let \(dq/dt\) be the translational and rotational velocities of movable platform, \(dL/dx\) be the actuators’ velocities, then

\[
\frac{dq}{dt} = J \frac{dL}{dt}
\]

or

\[
\frac{dL}{dt} = J^{-1} \frac{dq}{dt}
\]

where \(J\) is Jacobian matrix. Let \(k_j = \frac{\partial L}{\partial q_j}\) be the \(ij\)-element of \(J^{-1}\).

According to Ref.[7], \(k_i\) can be calculated by the structural parameters of manipulator and the position and orientation of the movable platform. So the inverse of Jacobian matrix is just the function of position and orientation.

\[
J^{-1} = f(x_P, y_P, z_P, \gamma, \beta, \alpha)
\]

2 Numerical Iterative Scheme of Forward Kinematics

Although Stewart parallel manipulator have 40 forward kinematics solutions[8], every time there is a possible solution when manipulator is running. Because the running is continual, this possible solution is very close to the last period position and orientation. When forward kinematics is used in real-time control, it is unnecessary to get all forward solutions, just the real forward solution is important.

Multiplying both sides of Eq.(4) by tiny period of time \(\Delta t\), we get

\[
\begin{bmatrix}
\Delta x_P \\
\Delta y_P \\
\Delta z_P \\
\Delta \beta \\
\Delta \alpha
\end{bmatrix} =
J
\begin{bmatrix}
\Delta L_1 \\
\Delta L_2 \\
\Delta L_3 \\
\Delta L_4 \\
\Delta L_5 \\
\Delta L_6
\end{bmatrix}
\]

Using above equation, we can calculate the increment of position and orientation by the increment of actuators’ length. So we can find direct kinematics solution of Stewart parallel manipulator by below numerical iterative scheme.

Known terms: the structural parameters of manipulator, actuators’ length \(L_{in}\) solution error \(e\).

Question: the position and orientation of the movable platform \(q (x_P, y_P, z_P, \gamma, \beta, \alpha)\).

Numerical iterative scheme:

Step 1: Select an initialization of position and orientation \(q_1\) (the last period value in running course).

Step 2: Calculate the six actuators’ length \(L_i\) corresponding to initialization by Eq.(3).

Step 3: Find the difference between this length and given length, \(\Delta L = L_{in} - L_i\).

Step 4: If \(\sum |\Delta L| < e\), finish solving process. Step 5: Calculate the inverse of Jacobian matrix \(J^{-1}\) by Eq.(6).

Step 6: Calculate Jacobian matrix \(J\).

Step 7: Calculate the increment of position and orientation \(\Delta q\) by Eq.(7).
Step 8: Get new position and orientation \( \mathbf{q}_i = \mathbf{q}_i + \Delta \mathbf{q} \). They are more correct than previous position and orientation.

Step 9: Go to Step 2 to continue.

Above method is validated by a real example. The manipulator is shown in Fig.2, it is being studied in Xi’an Jiaotong University.

The circumradius of movable platform and base are 740 mm and 970 mm respectively, and the angles corresponding to short side of platform hexagon are 11.6° and 8.9° respectively.

Suppose the given actuators’ length \( L_0 \) is \((1400, 1450, 1500, 1250, 1300, 1350)\), solution error \( e \) is 0.001 mm. The initialization of position and orientation is \((20, -5, 1100, 0, 0, 0)\), it is close to real position and orientation. Then the iterative course of solution is shown in Tab.1 and Tab.2. The position and orientation corresponding to final solution are shown in Fig.3.

According to the above experimentation, it needs only 4 step iterations to get a very precise forward kinematics solution when allowed error is 0.001 mm. It takes only 0.1 ms at computer of PIV 1.4G CPU. The iterative course is fast, and it will be faster if the iteration initialization is the last period position and orientation in running course. So this numerical iterative scheme is efficiently employed in real-time control of parallel manipulator.

### 3 The Initialization Problem of Numerical Iterative Scheme

In the numerical iterative course of last section, the initialization of position and orientation is close to real solution. The iteration initialization can be selected as the last period position and orientation when manipulator is running. But at the beginning of the running, there is no last period, namely, there is no initialization close to real solution. Then is the iterative solution correct if the iterative initialization is random?

Now we validate the numerical iterative scheme using different iterative initialization. The problem and given parameters are same to section 2.

#### 3.1 Iterative Initialization(0, 0, 1200, 0, -5, -15)

Using this iterative initialization, it needs 10 step iterations to get a forward kinematics solution. The solution is \((-486.249, -243.400, 673.308, 66.258, -230.314, 98.794)\). The position and orientation
corresponding to final solution are shown in Fig.4. This solution is one of 40 theoretic forward kinematics solutions. It is fit for 6 actuators’ length requirement, but it is an impossible position and orientation solution.

3.2 Iterative Initialization (0,1200,0,10,−5)

Using this iterative initialization, the solution is (−486.249, −243.400, 673.308, −113.742, 50.314, −81.206). It also needs 10 step iterations to get this forward kinematics solution. Although the solution is different to section 3.1, the position and orientation are alike to each other in the shape.

3.3 Iterative Initialization (0,0,1200,0,−5,−10)

Using this iterative initialization, the iterative course can not converge.

3.4 Iterative Initialization (0,0,1200,0,−5,0)

Using this iterative initialization, it needs 9 step iterations to get a forward kinematics solution (22.071, −10.902, 1151.873, 0.100, 349.498, −3.637). Although the solution is different from the correct solution, the position and orientation are alike to each other in the shape.

3.5 Iterative Initialization (0,0,1200,10,5,−5)

Using this iterative initialization, it needs 12 step iterations to get a forward kinematics solution (393.101, −272.850, 792.555, 266.154, −66.637, −297.191). The position and orientation corresponding to final solution are shown in Fig.5. This solution is also one of 40 theoretic forward kinematics solutions and fit for 6 actuators’ length requirement, but it is also an impossible solution.

According to above verification experimentation, if the iterative initialization is not suitable in the numerical iterative scheme, the final solution could be incorrect or the iterative course can not converge. So at the beginning of manipulator work, another efficient method is necessary to confirm an iterative initialization which is close to the correct forward kinematics solution.

4 IA-based Scheme of Confirming Iterative Initialization

The forward kinematics problem can be transformed into an optimal problem. Firstly a presumptive forward kinematics solution \( q(x, y, z, \gamma, \beta, \alpha) \) comes into being randomly. The corresponding actuators’ length \( L_i \) is calculated using simple inverse kinematics. The optimization object function is selected as

\[
 f(q) = \min \sum_{i=1}^{6} \left| L_{0i} - L_i \right|
\]  

(8)

The final optimizing solution \( q \) that satisfies the object function is the forward kinematics solution.

The optimal problem can be solved by random optimization schemes, such as Immune Algorithm(IA).

4.1 Immune Evolutionary Algorithm Based on Information Entropy

The immune system is our basic defense system against bacteria, viruses and other disease-causing organisms. By means of studying work mechanism of biologic immune system, many artificial immune algorithms are proposed, and Immune Evolutionary Algorithm based on Information Entropy (IEAIE) is a important kind of IA\(^{(9)}\). The IA is based on somatic theory and network theory, and integrates with genetic algorithm. The calculation flow of IEAIE is shown in Fig.6.

![Flow chart of IEAIE](image)

The detailed steps of IEAIE are as follows.

Step 1: Recognition of antigens. The objective function and constraints operate as the antigens of immune algorithm.

Step 2: Production of initial antibodies. \( N_I \) initial antibodies are randomly created on feasible space, and
initial antibodies are chosen from memory cell.

Step 3: Calculation of affinities. The affinity $a_{xv}$ between antigen and antibody $v$ and the affinity $a_{yw}$ between antibodies $v$ and $w$ are calculated.

Step 4: Send superior antibodies to memory cell. The antibodies that have high affinity with the antigen are added to the memory cell.

Step 5: Termination criterion. If the termination criterion is satisfied, the optimization procedure ends.

Step 6: Promotion and suppression of antibody production. The expectation value $e_{v}$ of every antibody is calculated. It is related to antibody’s affinity and density. The antibodies with high affinity are promoted and the antibodies with high density are suppressed.

Step 7: Production of antibodies. The antibodies which will perform the next optimization procedure are made by mutation and crossover.

Step 8: The optimization routine restarting from Step 3 till the termination criterion is satisfied.

The basic concepts of IEAIE are as follows.

Antigen: the object function.

Antibody: the feasible solution.

The affinity between antibodies: The affinity between the antibody $v$ and $w$ is defined as

$$a_{yw} = \frac{1}{1 + H(2)}$$

where $H(2)$ is the information entropy of antibody $v$ and $w$. $H(2)=0$ represents that the genes of antibody $v$ and $w$ are same.

Information entropy: Let the immune system be composed of $N$ antibodies having $M$ genes, as shown in Fig.7.

Every gene has $s$ selections, $k_1, k_2, \cdots, k_s$. Information entropy of $N$ antibodies represents the diversity of immune system, is defined as

$$H(N) = \frac{1}{M} \sum_{j=1}^{M} H_j(N)$$

where $H_j(N) = \sum_{i=1}^{s} -p_{ij} \log p_{ij}$, it is the information entropy of $j$th gene of antibodies. $p_{ij}$ is the probability that $i$th allele $k_i$ comes out at the $j$th gene.

The affinity between antigen and antibody: the affinity means combination intensity between antigen and antibody $v$, it can be defined as fitness function

$$a_{x} = \text{fitness}(v)$$

The density of antibody: the density of antibody $v$ is

$$c_{v} = \frac{1}{N} \sum_{w=1}^{N} a_{yw}$$

where $a_{yw} = \begin{cases} 1, & a_{yw} \geq T_{w_1} \\ 0, & \text{otherwise} \end{cases}$ $T_{w_1}$ is a threshold.

The expectation value of antibody: it is related to antibody’s affinity $a_{x}$ and density $c_{v}$ is defined as

$$e_{v} = \frac{a_{x} \prod_{i=1}^{N} (1 - a_{x_i})}{c_{v} \sum_{i=1}^{N} a_{x_i}}$$

where $a_{x_i} = \begin{cases} a_{yw}, & a_{yw} \leq T_{w_2} \\ 0, & \text{otherwise} \end{cases}$ $T_{w_2}$ is another given threshold.

4.2 IEAIE Scheme of Forward Kinematics

To deal with forward kinematics problem of parallel manipulator employing IEAIE, some actual operation must be finished.

4.2.1 Coding Scheme

The conventional coding methods include binary coding and real coding. Furthermore Gray coding is proposed for better local searching ability. Float coding and character coding are proposed for lower computation complexity and higher efficiency.

We employ binary coding in forward kinematics problem. According to the real Stewart parallel manipulator’s structure parameters and workspace, in solving course 6 variables are limited in definite range: $x \in (-100,100) \ , \ y \in (-100,100) \ , \ z \in (1,000,1,400) \ , \ y \in (-10,10) \ , \ \beta \in (-10,10) \ , \ \alpha \in (-20,20) .$

Every chromosome is a binary number of 48 bits, it represents a optimization value of position and orientation. Just like Fig.8, in chromosome $x$, $y$ and $z$ occupy 10 bits respectively, $\gamma$, $\beta$ and $\alpha$ occupy 6 bits respectively.
4.2.2 Production of Initial Population

Considering the diversity requirement, the probabilistic methodology is used to produce the initial population in chromosome space. At first optimizing procedure, there is nothing at memory cell, so memory population is produced by probabilistic method too. After an optimizing course, the superior chromosomes are put into memory cell.

4.2.3 Affinity and Expectation Value of Antibodies

The affinity of antigen and antibody can be defined as fitness function $f(v)$. The aim of optimal searching course is to find the maximum fitness of the individual chromosome, so it is necessary to transform the minimal objective function into the maximum fitness function.

$$f(v) = \frac{1}{f(q) + \lambda}$$

where $f(q)$ is defined in Eq.(8), and $\lambda$ is a constant, $\lambda = 0.01$, which avoids the disadvantage that IA operation can not proceed when the minimum value of $f(q)$ is zero.

The expectation value of antibody $v$ can be calculated by following concise formula

$$e_v = \frac{a_v}{c_v}$$

where $c_v$ is antibody’s density. Compared with Eq.(13), this method is not only efficient but also simple and practical.

4.2.4 Genetic Operation

The roulette wheel selection is used in this paper. The excellent and diverse antibodies are selected as the parents to reproduce offspring.

Crossover operation exchanges randomly genetic information between two antibodies. Because the answer is contained in the population as a whole and only by combining chromosomes will the best solution be found.

Mutation operation is to produce some new genes because of the possibility that the initial population don’t contain all of the information necessary to the problem. Mutation operation randomly changes some bits every generation based upon a specified mutation probability.

4.2.5 Termination Criterion

The biggest generation $G_{\text{max}}$ and fitness requirement $F_0$ must be given beforehand. If the fitness of the best individual is bigger than $F_0$ in optimization course, IEAIE stops and the best individual is the solution. If the fitness is smaller than $F_0$ after $G_{\text{max}}$ generation, initial population is produced randomly and IEAIE starts again. At algorithm $F_0$ is 0.5,

$$f(v) = \frac{1}{(f(q) + \lambda)} = \frac{1}{\min(\sum_{i=1}^{6} |L_i| + \lambda)} > F_0 = 0.5$$

so

$$\min(\sum_{i=1}^{6} |L_i|) < 2 - \lambda = 1.99$$

namely, final solution must satisfy the condition that the errors sum of actuators’ length is smaller than 1.99 mm.

4.3 Solution Result of Forward Kinematics Based on IEAIE

We validate the IEAIE by an actual example. The control parameters of IEAIE are shown in Tab.3.

<table>
<thead>
<tr>
<th>Control parameter</th>
<th>Population size</th>
<th>Memory population size</th>
<th>Chromosome length</th>
<th>Biggest generation $G_{\text{max}}$</th>
<th>Crossover probability</th>
<th>Mutation probability</th>
<th>Density threshold</th>
<th>Fitness requirement $F_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>100</td>
<td>30</td>
<td>48</td>
<td>500</td>
<td>0.50</td>
<td>0.05</td>
<td>0.85</td>
<td>0.50</td>
</tr>
</tbody>
</table>

Suppose the given actuators’ length is (1306.905, 1524.331, 1359.391, 1478.857, 1388.626, 1454.098). The correct forward kinematics solution should be (10, 50, 1200, 3, 0, 10).

Using above IEAIE, the best individual is got at 260 generation. Total crossover number is 12479 and total mutation is 120386. The final solution of position and orientation is (10.156, 50.781, 1199.609, 2.969, 0, 10.000), its fitness is 0.522521.

The corresponding position and orientation to the maximum fitness among the generations are shown in Fig.9 to Fig.12. The errors among three position solutions with correct solutions among generations are shown in Fig.13. The orientation errors are shown in Fig.14. The errors sum of actuators’ length among generations is shown in Fig.15. The biggest fitness among generations is shown in Fig.16.
Fig. 9 Position x in iteration

Fig. 10 Position y in iteration

Fig. 11 Position z in iteration

Fig. 12 Orientation in iteration

Fig. 13 Three position errors in iteration

Fig. 14 Three orientation errors in iteration

Fig. 15 The errors sum of actuators’ length

Fig. 16 The biggest fitness in iteration
It can be seen from the above results that the IEAIE solving of forward kinematics has good precision. Every position error is smaller than 1mm and every orientation error is smaller than 0.1°. This solution can be used as the initialization of numerical iterative scheme to get more precision solution.

As a comparison, genetic algorithm is employed to deal with forward kinematics problem. All operations and parameters are same as IEAIE. Using genetic algorithm the best individual is got at 472 generation. The corresponding position and orientation is \((7.812, 45.312, 1199.219, 3.047, -0.313, 10.000)\) and fitness is 0.08114. The maximum position error is approximate 5 mm and the orientation error is approximate 0.3°. The solution precision is worse than IEAIE.

6 Conclusions

The forward kinematics problem of Stewart parallel manipulator is very complicated, but it is necessary to solve real-timely this problem in the control course. A mixed algorithm combining IEAIE and numerical iterative scheme is proposed to deal with this problem in this paper.

Taking advantage of simpleness of inverse kinematics, the forward kinematics is transformed to an optimal problem. Before manipulator running, there is no real-time requirement in solving course, so IEAIE is employed to find approximate solution in whole workspace.

Using above solution as iterative initialization, numerical iterative scheme is employed to get more precise solution. In the manipulator running course, the iteration initialization can be selected as the last period position and orientation. Because the initialization is closed to correct solution, solving precision is high and speed is rapid enough to satisfy to real-time requirement.

This mixed forward kinematics algorithm is applied to real Stewart parallel manipulator in the real-time control course. The examination result shows that the algorithm is very efficient and practical.

References


Brief Introduction to Authors

WANG Sun-an (王孙安) was born in Jiangsu Province, China, in 1957. He received his Ph.D. degree from Xi’an Jiaotong University in 1989. As a visiting scholar, from 1997 to 1998, he joined the Rensselaer Polytechnic Institute. He is now a professor of Xi’an Jiaotong University. His research activities have been concerned with intelligent control, intelligent algorithm and distributed artificial intelligence system.

WAN Ya-min (万亚民) was born in Shaanxi Province, China, in 1972. He graduated from University of Science and Technology Beijing and entered Xi’an Precision Machinery Research Institute in 1993. He is now pursuing his Ph.D degree at Xi’an Jiaotong University. His specialization is intelligent control of mechatronics system and computer simulation.