Kinematics Analysis and Computing of A 2-Arm Drilling Manipulator

Zhang Fuxiang, Li Wenzhong

1 College of Mechanical Engineering, Hebei University of Science and Technology, Shijiazhuang, China, zhangfx@hebust.edu.cn
2 College of Mechanical Engineering, Hebei University of Science and Technology, Shijiazhuang, China, wenzhonglee@163.com

Abstract

In order to realize drilling holes automatically by a two-arm drilling manipulator, its direct kinematics was analysis. In this case, its inverse kinematics was difficult to resolve by traditional analytical analysis. An inverse kinematics analysis method of the two-arm manipulator based on the particle swarm optimization (PSO) was brought forward. The aim of optimization was to find the right angle position of the arm by which they can realize the position and orientation of the drill demanded on the drilling work. An optimization program of the PSO algorithm in the Matlab environment was developed and the optimal calculation was done. The result proved the validity of the PSO based algorithm for the inverse kinematics of the 2-arm drilling manipulator.

Keywords: Kinematics Analysis, Inverse Kinematics, 2-Arm Drilling Manipulator, Particle Swarm Optimization (PSO)

1. Introduction

People did drilling work by using its drilling manipulator of a drilling machine. The drilling manipulator is a typical multi-arm-and-joint robotic system [1,2]. Accurate location of the drilling mechanism is the key factor which can improve the quality of drilled holes and the precondition of computer control is the solving of the kinematics equation [3,4,5]. In order to finish the drilling work, the drilling manipulator should at least have 5 degree-of-freedom (DOF) [6]. In order to eliminate self-interference and improve the flexibility of operation, a drilling manipulator usually has more than 5 DOF [7]. CMJ2-27 is a 2-arm drilling machine produced by Shijiazhuang Coal Mining Machinery Co. Ltd. One arm of the CMJ2-27 has 5 revolute joints and one translation joint with totally 6 DOF. In this paper, the authors made a kinematical analysis for the CMJ2-27 drilling machine by the D-H coordinate transformation and establish the direct kinematics equation of its 2-arm drilling manipulator [8,9]. The inverse kinematics equation of the manipulator would not be established directly for its complicated structure. The authors presented an intelligent calculation method of inverse kinematics which was based on Particle Swarm Optimization (PSO).

2. Mathematical model of the drilling manipulator

2.1. Mechanical structure of the drilling manipulator

Figure 1 is the schematic diagram of drilling manipulator of CMJ2-27. The body’s parts of the drilling machine are reciprocally linked by cross hinges. The movement of lift, swing and the parallel movement of feed cylinder are realized by cylinders fixed on the manipulator. The movement of rotation of the drilling arm is realized by a hydraulic motor with a worm reduction gear.

2.2. Linkages and their coordinate systems

The manipulator of the drilling machine is a mechanism consists of a set of linked linkages. In order to describe the relationship of the linkages, people should set a coordinate system on every linkage. The coordinate systems fixed on the linkages. The position and orientation of the linkages can be described by their coordinate systems. The relationship of the adjacent linkages can also be described by the coordinate transformation of their coordinate systems. If people can set all the coordinate systems of the linkages and get the coordinate transformation of adjacent linkages such as from base to
NO.1 joint, No.1 joint to No.2 joint, …, etc., so people can get the whole transformation matrix of the manipulator.


Figure 1. Drilling manipulator of CMJ2-27

In order to describe the kinematics more practically for the user, we also set a user coordinate system in the sketch. There are 4 independent parameters in the method of Denavit-Harbenberg (D-H) coordinate transformation, i.e. using a $4 \times 4$ homogeneous transfer matrix to describe the relationship two adjacent linkages. So the problem of solving direct kinematics equation is simplified into solving the $4 \times 4$ homogeneous transfer matrix between two adjacent linkages. The $O_XY_Z_u$ coordinate system fixed on the dummy perfect working plane which is in front of the jagged drilling working face. The two arms of the manipulator have similar mechanical structure. The one and only difference of the left arm and the right arm is the motion range of swing angle of the drilling arm. So the two arms have identical coordinate systems. The $O_XY_Z_u$ coordinate system is the base coordinate system of an arm. The $O_XY_1Z_1$, $O_XY_2Z_2$, $O_XY_3Z_3$, $O_XY_4Z_4$, $O_XY_5Z_5$ and $O_XY_6Z_6$ coordinate system are linkage coordinate systems of linkage 1 to linkage 6. Here, the $O_XY_6Z_6$ coordinate system is also the tool
coordinate system of an arm. The origin of OX,Y,Z is on the perpendicular bisector of the line through the two origins of the two base coordinate systems (i.e. OX₀,Y₀,Z₀).

2.3. Kinematics equation

According to the real structure of the manipulators, we give the D-H parameters of the robot arm as shown in Table 1.

<table>
<thead>
<tr>
<th>Linkage</th>
<th>Rotation angle ( (\theta) )</th>
<th>Linkage Distance ( (d) )</th>
<th>Linkage length ( (a) )</th>
<th>Torsion angle ( (\alpha) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \theta_1(-14º<del>47º/-47º</del>14º) )</td>
<td>( d_1=0 )</td>
<td>( a_1=115 )</td>
<td>( \alpha_1=90º )</td>
</tr>
<tr>
<td>2</td>
<td>( \theta_2(-16º~55º) )</td>
<td>( d_2=0 )</td>
<td>( a_2=60 )</td>
<td>( \alpha_2=90º )</td>
</tr>
<tr>
<td>3</td>
<td>( \theta_3(-180º~180º) )</td>
<td>( d_3=3400 )</td>
<td>( a_3=0 )</td>
<td>( \alpha_3=90º )</td>
</tr>
<tr>
<td>4</td>
<td>( \theta_4(-105º~15º) )</td>
<td>( d_4=0 )</td>
<td>( a_4=130 )</td>
<td>( \alpha_4=90º )</td>
</tr>
<tr>
<td>5</td>
<td>( \theta_5(-45º~45º) )</td>
<td>( d_5=479.5 )</td>
<td>( a_5=0 )</td>
<td>( \alpha_5=90º )</td>
</tr>
<tr>
<td>6</td>
<td>( \theta_6=0 )</td>
<td>( d_6(-3060~1260) )</td>
<td>( a_6=0 )</td>
<td>( \alpha_6=90º )</td>
</tr>
</tbody>
</table>

Table 1. D-H parameters of the arm

Usually, people use position-orientation matrix describes the position and orientation of the given end-effector (namely drilling pipe here). According to the homogeneous transfer matrix \( A_1, A_2, \ldots, A_6 \), we can get the kinematics equation of the drilling manipulator as follow:

\[
T_6 = A_6 A_5 A_4 A_3 A_2 A_1 \tag{1}
\]

Here we simplify the expression of the variable of \( \theta_i \) as the follows:

\[
s_i = \sin \theta_i, \quad c_i = \cos \theta_i
\]

Here, we use \( A_i \) denote the homogeneous transfer matrix from the coordinate system \( i (OX_i,Y_i,Z_i) \) to the coordinate system \( i-1 (OX_{i-1},Y_{i-1},Z_{i-1}) \). The homogeneous transfer can be realized by 4 transfers as:

\[
A_1 = \text{Rot}(Z,\theta_1) \cdot \text{Trans}(0,0,d_1) \cdot \text{Trans}(a_1,0,0) \cdot \text{Rot}(X,\alpha_1) \tag{2}
\]

\[
A_2 = \text{Rot}(Z,\theta_2) \cdot \text{Trans}(a_2,0,0) \cdot \text{Rot}(X,\alpha_2) = \begin{bmatrix}
c\theta_2 & 0 & s\theta_2 & a_2 c\theta_1 \\
s\theta_2 & 0 & -c\theta_2 & a_2 s\theta_1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{3}
\]

\[
A_3 = \text{Rot}(Z,\theta_3) \cdot \text{Trans}(a_3,0,0) \cdot \text{Rot}(X,\alpha_3) = \begin{bmatrix}
c\theta_3 & 0 & s\theta_3 & a_3 c\theta_2 \\
n\theta_3 & 0 & -c\theta_3 & a_3 s\theta_2 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{4}
\]

\[
A_4 = \text{Rot}(Z,\theta_4) \cdot \text{Trans}(a_4,0,0) \cdot \text{Rot}(X,\alpha_4) = \begin{bmatrix}
c\theta_4 & 0 & -s\theta_4 & 0 \\
s\theta_4 & 0 & c\theta_4 & 0 \\
0 & 1 & 0 & d_4 \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{5}
\]
Here, we use $A_i$ denote the homogeneous transfer matrix from the base coordinate system $(O_X0Y0Z0)$ to the user coordinate system $(O_XuYuZu)$. The homogeneous transfer can be realized by the transfers as:

$$A_i = \text{Rot}(Z,\theta_i) \cdot \text{Trans}(a_i,0,0) \cdot \text{Rot}(X,\alpha_i)$$

(6)

$$A_i = \text{Rot}(Z,\theta_i) \cdot \text{Trans}(0,0,d_i) \cdot \text{Rot}(X,\alpha_i)$$

(7)

$$A_i = \text{Trans}(0,0,d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(8)

In (8), for left arm $x < 0$ and for right arm $x > 0$.

Here, we set the whole kinematics equation of the drilling manipulator as follow:

$$T_u = A_i T_0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(10)

In which, simplify the $\sin \theta_i$ as $s_i$ and $\cos \theta_i$ as $c_i$, the matrix elements are as follows:

$$n_x = -s_i \left(c_2c_3c_4s_5 + c_2s_3c_5 - s_2s_4c_5\right) + c_1 \left(s_3c_4c_5 - c_3s_5\right)$$

$$n_y = s_i \left(c_2c_3c_4s_5 + c_2s_3c_5 - s_2s_4c_5\right) - s_1 \left(s_3c_4c_5 - c_3s_5\right)$$

$$o_x = s_i \left(c_2s_3s_5 + s_2s_4c_5\right) - c_1 s_i s_5$$

$$o_y = c_2c_4 - s_i c_3 s_4$$

$$a_x = c_i \left(c_2c_3s_5 + s_2s_4c_5\right) + s_1 s_i s_5$$

$$a_y = c_i \left(c_2c_3s_5 + s_2s_4c_5\right) + c_1 s_i s_5$$

$$p_x = -s_i c_2c_3s_5d_6 - s_i c_2c_3a_5 + s_1 c_2c_3s_5d_5 + s_1 c_2c_3a_5d_6 + s_1 s_i s_5d_6 + s_1 s_i a_5d_6$$
In this case, the manipulators consist of two robot arms; every one of them is a serial robot. For a serial robot, to resolve the inverse kinematics is more difficult than that of direct kinematics. According to the demand of drilling task, the target holes on work plane with the position of \( p_z = 0 \). The expression of the matrix elements are so complex, we cannot find the analytical solution of the inverse kinematics equation. Sometimes, people cannot get the analytical solution. So to get the numerical solution is a feasible method to solve kinematics equation of a robot.

3. Particle swarm optimization based kinematics Computing

3.1. Introduction of PSO

Particle swarm optimization (PSO) is a new evolutionary algorithm of global optimization invented by Eberhart and Kennedy [10]. Rooted in the simulation of the avian predatory behavior, this algorithm has outstanding performance on nonlinear optimization. It is also simple to realize. Now, it has become an important optimization tool.

In PSO, the positions of the particles denote possible solutions of the optimization. We can evaluate a solution by the particles’ fitness. First, we should initialize a group of random particles, and then get the optimal solution through repetitious iterative calculation. During the process of each iterative calculation, the particles renew themselves by tracking two optimal values: one is the optimal value of one particle, namely individual optimal value which is expressed as \( p_{best} \); the other is that of the particles swarm which is expressed as \( g_{best} \). After the particles find the two optimal values, they can renew themselves by the following two formulas.

\[
\begin{align*}
V_{i+1}^{k} &= w \cdot V_{i}^{k} + c_{1} \cdot r_{1} \left( p_{i}^{k} - x_{i}^{k} \right) + c_{2} \cdot r_{2} \left( g_{best}^{k} - x_{i}^{k} \right) \\
x_{i+1}^{k} &= x_{i}^{k} + V_{i+1}^{k}
\end{align*}
\]

In the formulas, \( k \) is the generation of calculation; \( i \) is the number of a particle; \( V \) is the velocity of the particles; \( x \) is the current position of the particles; \( p \) is the vector form of \( p_{best} \); \( g_{best} \) is the vector form of \( g_{best} \); \( r_{1} \) and \( r_{2} \) are random numbers between 0 to 1; \( c_{1} \) and \( c_{2} \), usually called as learning factors, are chosen for \( c_{1} = c_{2} = 2 \); \( w \) is weighting coefficient between 0.1 to 0.9.

3.2. Improved algorithm

During the late searching process in the solution space by using the basic algorithm of PSO, the particles may continue oscillating near the global optimal solution. In order to solve this problem, we make the following improvements [11,12]: Along with the iterative calculation, make the weighting coefficient of \( w \) decrease from the maximum value \( w_{max} \) to the minimum value \( w_{min} \) linearly.

\[
w = \frac{w_{max} - w_{min}}{G_{max}} \cdot G
\]

In (13), \( G \) is the current generation; \( G_{max} \) is the times of total iterative calculation. After the improvement, the value of \( w \) is relatively large in the early optimization calculation, and relatively small in the late optimization calculation. Thus, the fine-searching ability of the PSO algorithm is improved.
In order to guarantee the convergence rate, particles’ searching should be made in a limited solution space. So it is necessary to restrict the position space and the velocity space of the particles. For (10), the limited conditions are shown in (14) and (15) [13, 14].

\[
x_r = \left( x_{1\text{min}}, x_{1\text{max}}, x_{2\text{min}}, x_{2\text{max}}, x_{3\text{min}}, x_{3\text{max}}, \ldots, x_{4\text{min}}, x_{4\text{max}}, x_{5\text{min}}, x_{5\text{max}}, x_{6\text{min}}, x_{6\text{max}} \right)
\]

\[
v_r = \left( v_{1\text{min}}, v_{1\text{max}}, v_{2\text{min}}, v_{2\text{max}}, v_{3\text{min}}, v_{3\text{max}}, \ldots, v_{4\text{min}}, v_{4\text{max}}, v_{5\text{min}}, v_{5\text{max}}, v_{6\text{min}}, v_{6\text{max}} \right)
\]

3.3. Parameter encoding

In order to solve \( x \), we should code the design variables properly to generate the particles used in SPO. According to the characteristics of PSO, we can directly use real numbers to denote the parameters. For every particle of the mathematical model shown in (10), we use \( x = \{\theta_1, \theta_2, \theta_3, \theta_4, d_6\} \) to denote the current position, and \( v = \{v_1, v_2, v_3, v_4, v_5, v_6\} \) to denote the velocity. The code of \( x \) is:

\[
\begin{array}{cc}
\theta_1, \theta_2, \theta_3, \theta_4, d_6 & v_1, v_2, v_3, v_4, v_5, v_6 \\
\text{Particle position} & \text{Particle speed}
\end{array}
\]

3.4. Fitness of the inverse kinematics computing

To solve the inverse kinematics equation is to find a right joints position get the expected position and orientation of the drill. In the calculation process of PSO, every particle can get a position and orientation of the drill by using direct kinematics computing [15]. We think that a particle is better than another one when the position and orientation of the drill represented by it is more close to the expected position and orientation of the drill. The position and orientation of the drill can be expressed by 3 orientation vector \( n, o, a \) and 1 position vector \( p \).

To express the degree of the closeness of two position vectors, we can simply calculate the distance of two points. We denote the expected position by \( pt \) and denote the position of a particle by \( pu \). In Matlab, we can use the function of norm() calculate the distance of two position vectors. Here we use similar_p denote the degree of closeness between the position vectors of a particle and that of the expected drill. The Matlab expression is shown in (16).

\[
\text{similar}_p = \frac{\text{norm}(pt-pu)}{\text{maxdistance}};
\]

In (16), maxdistance is the maximum distance between the expected position and all the particles in 1st generation of PSO. The value of similar_p is in the interval of 0 to 1.

To express the degree of closeness of two orientation vectors, we can reference to the cosine theorem of two space vectors [11]. The cosine theorem can be expressed by (17).

\[
\cos \alpha = \frac{a \cdot b}{|a||b|}
\]

In (17), \( a \) and \( b \) are two space vectors, and \( \alpha \) is the angle between \( a \) and \( b \).

According to (21), if we want \( a \) close to \( b \), the value is between 0 to 1 and the higher the better. We denote the expected orientation by \( nt, ot \) and \( at \). The orientations of a particle can be denoted by \( nu, ou \) and \( au \). Here we use similar_n, similar_o and similar_a denote the degree of closeness between the orientation vectors of a particle and that of the expected drill. The Matlab expression are shown in (18), (19) and (20).

\[
\text{similar}_n = 1-\text{dot}(nt,nu)/(\sqrt{\text{sum}(nt.*nt)})*\text{sqrt}(\text{sum}(nu.*nu));
\]

\[
\text{similar}_o = 1-\text{dot}(ot,ou)/(\sqrt{\text{sum}(ot.*ot)})*\text{sqrt}(\text{sum}(ou.*ou));
\]
similar_a = 1 - dot(at, au) / (sqrt(sum(at.*at)) * sqrt(sum(au.*au))); \hspace{1cm} (20)

The value of similar_n, similar_o, similar_a is in the interval of 0 to 1. According to (16), (18), (19) and (20), we get the Matlab expression of the fitness of a particle as shown in (21).

\[ \text{fitness} = c_1 \times \text{similar}_n + c_2 \times \text{similar}_o + c_3 \times \text{similar}_a + c_4 \times \text{similar}_p; \hspace{1cm} (21) \]

In (21), c1, c2, c3 and c4 are weight factors of n, o, a and p. Here we adopt the factors as c3=3, c1=c2=c4=1. It means that similar_a is more important than the other factors. Because similar_a denotes the direction of drill axis which is the most important in a drilling work.

3.5. Parameter encoding

(1) Set PSO parameters such as group size \( S_{\text{pop}} \), dimension number \( S_{\text{dim}} \), weighting factor \( w \), position space \( x_r \) and velocity space \( v_r \).

(2) Set correlative design constants such as \( \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, d_s \).

(3) Initialize the particles swarm: including random initializing the position and the velocity of the particles in the parameter interval and remove the invalid particles according to the design constraints listed in the equation of (10).

(4) Calculate the fitness of the particles.

(5) Initialize current position of the particles as \( p_{\text{best}} \) and the minimum of all \( p_{\text{best}} \) as \( g_{\text{best}} \).

(6) Renew the position and the velocity of the particles according to the equations of (15) and (16).

(7) Calculate the fitness of the particles again.

(8) Renew \( p_{\text{best}} \) and \( g_{\text{best}} \) of the particles.

(9) Redo the steps from step (6) to step (8) until the iterative calculation reach set precision or set generations.

(10) Export \( g_{\text{best}} \) and get the parameters of the experimental mechanism.

4. Results and analysis

In order to validate the PSO algorithm, we have written the program under the Matlab environment and have made simulation. In the simulation, we set the parameters of PSO as follows: maxgeneration = 100, \( S_{\text{pop}} = 20, S_{\text{dim}} = 6, w_{\text{min}} = 0.1, w_{\text{max}} = 0.9 \), we set the other parameter according to actual structure as follows: \( x_r = -312.5 \) for left arm and \( x_r = 312.5 \) for right arm, \( y_u = 1500, z_u = 5505 \). According to Table 1, we can get the vectors of \( x_r \) and \( v_r \).

For left arm, the Matlab expression of \( x_r \) is:

\[
x_r = [-14*\pi/180, 47*\pi/180, -106*\pi/180, 135*\pi/180, -180*\pi/180, 180*\pi/180, -15*\pi/180, 105*\pi/180, -135*\pi/180, -45*\pi/180, -3060, -1260];
\]

For left arm, the Matlab expression of \( x_r \) is:

\[
x_r = [-47*\pi/180, 14*\pi/180, -106*\pi/180, 135*\pi/180, -180*\pi/180, 180*\pi/180, -15*\pi/180, 105*\pi/180, -135*\pi/180, -45*\pi/180, -3060, -1260] \text{ for right arm.}
\]

The Matlab expression of \( v_r \) is:

\[
v_r = [-0.5, 0.5, -0.6, 0.6, -\pi, \pi, -1, 1, -0.8, 0.8, -900, 900].
\]

As an example, we set the expected position and orientation of a working hole drilled by the left arm as:

\[
T = \begin{bmatrix}
0.7071 & -0.7071 & 0 & -915.44 \\
0.7071 & 0.7071 & 0 & 2342.25 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The optimization procedure of the PSO algorithm is shown in Fig. 3.
The computing results are $\theta_1 = 4.322$, $\theta_2 = -80.4569$, $\theta_3 = 45.7245$, $\theta_4 = 80.2371$, $\theta_5 = 85.9172$, $d_s = -1907.4$. The results show that the PSO based method is valid for the inverse kinematics computing.

5. Acknowledgement

The authors also want to give their thanks to the Distinguished Young Projects “Research Direction of Electromechanical Control and Robotics” funded by Hebei University of Science and Technology for the funding support. This research work is also supported by the Science and Technology Development Project of Shijiazhuang City (11108171A-1).

6. References

