A trajectory planning algorithm for a multi-degree-of-freedom parallel robot based on NURBS research and simulation

Zhiqiang Wang1, Zhe Wang1 *, Panpan Xun1, Yuhui Shi1, Han Su1, Han Jiang1
1Northwest Institute of Mechanical & Electrical Engineering, Xianyang, Shaanxi, 712099, China
First author’s e-mail: wangzhiqiang1663@bq202.com

Abstract. Parallel robots are small in size and high in accuracy, so they are widely used in processing and production industries. In order to give full play to the characteristics of high-speed motion of parallel robots, trajectory planning has become a hot spot in the industry. This paper proposes a trajectory planning algorithm based on NURBS. Firstly, the algorithm uses the characteristics of NURBS continuous and derivable to fit interpolation points, and then we obtain a smooth and continuous robot motion trajectory. Secondly, It uses interpolation algorithm to discretize the curve to get the robot motion sequence. Finally, It completes the entire motion process of the robot in three-dimensional space. Experiments show that the algorithm can be applied to the parallel robot control system, which can give full play to the performance of the parallel robot and ensure the stable and high-speed operation of the robot.

1. Introduction
Parallel robots are fast, small in size, and high in control accuracy. They are widely used in many fields such as food, electronics, production and processing. In order to give full play to the characteristics of high motion of parallel robots and ensure that parallel robots complete tasks smoothly and quickly, parallel robot trajectory planning algorithm came into being.

The parallel robot trajectory planning algorithm is to control the robot to move according to the trajectory of task planning under the constraints of the robot motion index[1]. The basic trajectory planning algorithms include: T-shaped line planning, S-shaped line planning, polynomial planning, arc and segmented line segment planning, etc. For complex path planning, the piecewise line planning algorithm is usually used to approximate, but the piecewise line algorithm is difficult to deal with the speed of the transition section and has a large amount of calculation, which makes it difficult to use in practice. The NURBS curve has powerful modeling ability and curve fitting function, and it is more and more popular in the application of complex path approximation[2].

2. Kinematic modeling
Parallel robot trajectory planning is one of the important directions of parallel robot research, which belongs to the category of robot kinematics research. To study this problem, the kinematics model of robot and its solution must be defined first. The following is a description of the process of robot kinematics modeling.

The multi-degree-of-freedom parallel robot (hereinafter referred to as the parallel robot) studied in this paper is a widely used structure in the industrial and academic fields. As shown in Figure 1, the
structure is mainly composed of a moving platform, three moving branches and a moving platform. To simplify, we establish a right-handed coordinate system based on the plane where the static platform is located, and the established coordinate system is shown in Figure 2. Table 1 lists the meaning of the symbols in Figure 2.

![Figure 1: Robot structure schematic](image1)

![Figure 2: Robot simplified process](image2)

**Table 1. Model parameter comparison table**

| symbol | value | Symbol meaning |
|--------|-------|----------------|
| $A_iB_i$ | $L_a$ | Active arm |
| $B_iC_i$ | $L_b$ | Slave arm |
| $\theta_i$ | | Angle of active arm to static platform |
| $OA_i$ | $R$ | Circumcircle radius of static platform |
| $OB_i$ | $r$ | Circumscribed radius of moving platform |
| $\phi_i$ | | The angle between the vertex and the center of the static platform on the X-axis |

It can be seen from the above figure that the motion of the parallel robot is to drive the mechanical active arm and the slave arm to move through joint motions to achieve the expected trajectory and goal. Kinematics modeling analyzes the above-mentioned motion process and can describe its motion process, which include position solution, velocity analysis, singularity and workspace analysis, etc. Here trajectory planning we mainly focus on the problem of robot position solution (including forward kinematics solution and inverse kinematics solution). The process of obtaining the motion position of the robot in the three-dimensional space from the joint angle is called the forward kinematics solution; conversely, the process of obtaining the joint angle from the robot coordinates in the three-dimensional space is called the inverse kinematics solution. The forward and inverse kinematics solutions will be deduced below.

### 2.1. Positive kinematics solution

The positive kinematics solution is the process of obtaining the motion position of the robot in the three-dimensional space from the joint angle, which is obtaining the coordinates $(x, y, z)$ in the three-dimensional space from the corresponding $(\theta_1, \theta_2, \theta_3)$ in the joint space.

Since there is only translation between the moving platform and the static platform during the movement, the vectors $\overrightarrow{OA_i}$, $\overrightarrow{A_iB_i}$, $\overrightarrow{B_iC_i}$, $\overrightarrow{Oo}$, $\overrightarrow{oC_i}$ constitute a closed figure during the movement, so:
\[ \overrightarrow{BC} = \overrightarrow{Oo} + \overrightarrow{O'C} - \overrightarrow{OA} - \overrightarrow{AB}_i \]  

(1)

It is also known that the description of each vector in the coordinate system is:

\[
\overrightarrow{OA} = \begin{bmatrix} R \cos \varphi \ \\
R \sin \varphi \ \\
0 \end{bmatrix}, \quad \overrightarrow{AB}_i = \begin{bmatrix} L_i \cos \theta \cos \varphi \ \\
L_i \cos \theta \sin \varphi \ \\
0 \end{bmatrix}, \quad \overrightarrow{O'C} = \begin{bmatrix} r \cos \varphi \ \\
r \sin \varphi \ \\
0 \end{bmatrix}
\]

Suppose the vector \( \overrightarrow{Oo} = [x, y, z] \), and \( \overrightarrow{OC} \) set the formula after being brought in. The coordinate \( (x, y, z) \) in the three-dimensional space in formula 2 is the result of the positive kinematics solution we require.

\[
[(r - R - L_i \cos \theta \cos \varphi + x)^2 + (r - R - L_i \cos \theta \sin \varphi + y)^2 + (z - L_i \sin \varphi)^2 = L_s^2 \quad (i = 1, 2, 3) \]  

(2)

The above formula 2 is actually a set of formulas, we can get:

\[
\begin{align*}
[a_1 + x]^2 + [b_1 + y]^2 + [c_1 + z]^2 &= LB^2 \\
[a_2 + x]^2 + [b_2 + y]^2 + [c_2 + z]^2 &= LB^2 \\
[a_3 + x]^2 + [b_3 + y]^2 + [c_3 + z]^2 &= LB^2
\end{align*}
\]

among them:

\[
a_1 = r - R - L_i \cos \theta; b_1 = 0; c_1 = -L_i \sin \theta \\
a_2 = -\frac{1}{2} (r - R - L_i \cos \theta); b_2 = \frac{\sqrt{3}}{2} (r - R - L_i \cos \theta); c_2 = -L_i \sin \theta \\
a_3 = -\frac{1}{2} (r - R - L_i \cos \theta); b_3 = \frac{\sqrt{3}}{2} (r - R - L_i \cos \theta); c_3 = -L_i \sin \theta
\]

Solve the above three equations simultaneously, eliminate \( x \) and \( y \), and get the quadratic equation about \( z \):

\[
D \cdot z^2 + E \cdot z + F = 0
\]

(3)

among them:

\[
D = e_1^2 + e_2^2 + 1; E = 2e_1 \cdot f_1 + 2e_2 \cdot f_2 + 2e_3; F = (a_1 + f_1)^2 + f_2^2 + c_1^2 - L_s^2 \\
e_1 = \frac{(a_1 - a_2) \cdot b_1 - (a_1 - a_3) \cdot b_2}{b_1 \cdot d_1 - b_2 \cdot d_2}, e_2 = \frac{(a_2 - a_1) \cdot b_1 - (a_2 - a_3) \cdot b_2}{b_1 \cdot d_1 - b_2 \cdot d_2}, e_3 = \frac{(a_3 - a_1) \cdot b_1 - (a_3 - a_2) \cdot b_2}{b_1 \cdot d_1 - b_2 \cdot d_2} \\
f_1 = \frac{(a_1 - a_2) \cdot b_1 - (a_1 - a_3) \cdot b_2}{b_1 \cdot d_1 - b_2 \cdot d_2}, f_2 = \frac{(a_2 - a_1) \cdot b_1 - (a_2 - a_3) \cdot b_2}{b_1 \cdot d_1 - b_2 \cdot d_2}, f_3 = \frac{(a_3 - a_1) \cdot b_1 - (a_3 - a_2) \cdot b_2}{b_1 \cdot d_1 - b_2 \cdot d_2} \\
d_1 = \frac{1}{2} (a_1^2 + b_1^2 + c_1^2) - a_1^2 - c_1^2; d_2 = \frac{1}{2} (a_2^2 + b_2^2 + c_2^2) - a_2^2 - c_2^2; d_3 = \frac{1}{2} (a_3^2 + b_3^2 + c_3^2) - a_3^2 - c_3^2
\]

The quadratic equation of \( z \) can be obtained by eliminating \( x \) and \( y \) by simultaneous solution of the above three expressions:

\[
\begin{align*}
x &= f_1 + e_1 \cdot z \\
y &= f_2 + e_2 \cdot z \\
z &= (-E + \sqrt{E^2 - 4DF}) / 2D
\end{align*}
\]

(4)

According to the above formula, the unique position corresponding to the robot in the three-dimensional space can be obtained according to the angle in the joint space.

2.2. Kinematics inverse solution

There are many ways to solve the kinematics of parallel robots. Here we use the method in Zhang Wei's paper[3] to introduce trigonometric functions to solve the problem:
\[
\begin{align*}
t & = \tan\left(\frac{\theta}{2}\right)(\theta \neq 2k\pi + kk\pi, k \in Z) \\
\sin \theta & = \frac{2t}{1 + t^2} \\
\cos \theta & = \frac{1 - t^2}{1 + t^2}
\end{align*}
\]
(6)

Put the above value into formula 2 to get:

\[K_i t_i^2 + U_i t_i + V_i = 0 (i = 1, 2, 3)\]
(7)

Among them:

\[t_i = \tan\left(\frac{1}{2} \theta_i\right), U_i = 4L_i \cdot z\]

\[K_i = x_i^2 + y_i^2 + z_i^2 + (R - r)^2 + L_i^2 - L_i^2 - 2[x \cos(\phi_i) + y \sin(\phi_i)] \cdot (R - r)\]

\[+2L_i \cdot [x \cos(\phi_i) + y \sin(\phi_i) - R + r]\]

\[V_i = x_i^2 + y_i^2 + z_i^2 + (R - r)^2 + L_i^2 - L_i^2 - 2[x \cos(\phi_i) + y \sin(\phi_i)] \cdot (R - r) - 2L_i \cdot [x \cos(\phi_i) + y \sin(\phi_i) - R + r]\]

Solving the above equations can get:

\[
\begin{cases}
\theta_i = \begin{cases}
2 \arctan\left(\frac{-U_i \pm \sqrt{\Delta_i}}{2K_i}\right) \ldots K_i \neq 0 \text{ and } \Delta_i \geq 0 \\
\text{无解} \ldots K_i \neq 0 \text{ and } \Delta_i < 0 \\
2 \arctan\left(\frac{-U_i}{2K_i}\right) \ldots K_i = 0
\end{cases}
\end{cases}
\]
(8)

It can be seen from the results that the inverse solution of the parallel robot has multiple solutions, except for special cases, there are two sets of solutions, and the other set of solutions of the robot can be excluded according to the robot mechanism parameters to finally obtain a unique set of solutions.

3. Trajectory planning algorithm description

The purpose of trajectory planning is to generate a smooth robot motion trajectory according to the fitting of the input points. In the industry, piecewise polynomials or piecewise straight lines are generally used to fit complex curves, but the switching speed between the segmented fitting segments and the segment is dialed. The movement is too large to achieve continuous speed and acceleration. The NURBS curve is a set of piecewise polynomial curves determined by the control points. Its fitting curve has the characteristics of continuous and differentiable. The above problems can be solved by using the NURBS curve for trajectory planning. The process of using the NURBS curve for trajectory planning can be divided into fitting parts and interpolation part.

3.1. NURBS fitting

First introduce the expression form of the cubic NURBS curve. Here we use its matrix form to describe the form[4], where \( u \) is the control variable with a value of \([0, 1]\), \( n \) is the number of interpolation points, and \( \omega_i \) is the weight factor, which corresponds to the interpolation points one-to-one. \( d_i \) is the control point.

\[
F_i(t) = \frac{\sum_{j=3}^{j=3} \omega_j d_j N_{j,3}(u)}{\sum_{j=3}^{j=3} \omega_j N_{j,3}(u)} = \left[\begin{array}{c}
t^0 \\
t^1 \\
t^2 \\
t^3
\end{array}\right] N_i \left[\begin{array}{c}
\omega_{i-3} d_{i-3} \\
\omega_{i-2} d_{i-2} \\
\omega_{i-1} d_{i-1} \\
\omega_i d_i
\end{array}\right]^T, 3 \leq i \leq n
\]
(9)

In the above formula:
The above method can be used to obtain the representation of each segment of the NURBS curve, which can be transformed into the following expression form:

\[
N_i = \begin{bmatrix}
    n_{i1} & n_{i2} & n_{i3} & n_{i4} \\
    n_{i5} & n_{i6} & n_{i7} & n_{i8} \\
    n_{i9} & n_{i10} & n_{i11} & n_{i12} \\
    n_{i13} & n_{i14} & n_{i15} & n_{i16}
\end{bmatrix}
\]

\[
\begin{bmatrix}
    \sum_{j=1}^{i-1} \frac{(\nabla_j^1)^2}{\nabla_j^{i-1}} - n_{i1} - n_{i3} & (\nabla_j^{i-1})^2 & 0 \\
    -3n_{i1} & 3n_{i1} - n_{i3} & (\nabla_j^{i-1})^2 & 0 \\
    3n_{i1} & -3n_{i1} - n_{i3} & (\nabla_j^{i-1})^2 & 0 \\
    -n_{i1} & n_{i1} - n_{i3} - n_{i4} & (\nabla_j^{i-1})^2 & 0 \\
\end{bmatrix}
\]

The above formula uses the control points to obtain the NURBS curve. The input obtained in the actual use process is more of the points on the curve (hereinafter referred to as the type value point). Here is a way to use the type value point to obtain the control point. The specific process is as follows:

1) Determine the node vector

Assuming that there are n+1 type value points, there are corresponding n+3 control points and n+3 weights. The parameterization method of accumulated chord length is introduced:

\[
\begin{align*}
\begin{cases}
u_0 = u_1 = u_2 = u_3 = 0 \\
u_{i+2} = u_{i+3} = u_{i+4} = u_{i+5} = u_{i+6} = 1 \\
\sum_{j=1}^n [p_j - p_{j+1}] = 0
\end{cases}
\end{align*}
\]

(11)

2) Inverse control point

According to the continuity theorem between segments, the following boundary conditions are introduced:

\[
\begin{align*}
\begin{cases}
F_{i+3}(0) &= p_0 \\
F_{i+3}(1) &= F_{i+3}(0) = p_i, i = 1, 2, \ldots, n
\end{cases}
\end{align*}
\]

(12)

Put t=0 into it to get n+1 equations:

\[
p_i = F_{i+3}(0) = \left[\begin{array}{ccc}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{array}\right] \left[\begin{array}{c}
\omega_{i-3} \omega_{i-2} \omega_{i-1} \omega_i \\
\omega_{i-2} \omega_{i-1} \omega_i \\
\omega_{i-1} \omega_i
\end{array}\right] = \left[\begin{array}{c}
\frac{\partial D_j}{\partial \omega_i} (d_i-d_0) \\
\frac{3}{\omega_{i+3}} (d_{i+2} - d_{i+1})
\end{array}\right]
\]

(13)

There are a total of n+3 control points, supplemented with two tangent vector conditions:

\[
F_{i+3}(0) = \frac{3}{\omega_{i+3}} (d_{i+2} - d_{i+1}), F_{n+3}(0) = \frac{3}{\omega_{n+2}} (d_{n+2} - d_{n+1})
\]

The n+3 control points can be solved by the above conditions.
Inverse calculation factor

With default weights (usually 1), the process of constructing weight factors is as follows:

\[ h_i = \sum_{j=1}^{4} \omega_j N_{i,j}^{3,3}(u_{i,j}) = \langle n_{i1}, n_{i2}, n_{i1} \rangle \omega_i, \omega_i, \omega_i \rangle^T, \quad i = 0, 1, \ldots, n \] (14)

Similar to the above, two boundary conditions need to be added:

\[ h_0' = \frac{3(\omega_0 - \omega_3)}{V_3}, \quad h_n' = \frac{3(\omega_{n+2} - \omega_{n+1})}{V_{n+2}} \]

3.2. NURBS interpolation

It can be seen from the above derivation that the NURBS curve is a function of the control variable \( u \), but the input required in the actual motion of the robot is a set of sequences about time \( t \), so \( u \) must be expressed by \( t \) to find the coordinate points \( \{F(t_0), F(t_1), \ldots, F(t_n)\} \). The process is the interpolation process[5].

The above interpolation process is actually a process of seeking the relationship between \( u \) and \( t \), and because the robot control system is a discrete control system, the above relationship is a discrete form, and the relationship between \( u \) and \( t \) can be expressed as[6]:

\[ u(t) = u(t_i) + u'(t)(t - t_i) + \frac{u''(t)}{2!}(t - t_i)^2 + O((t - t_i)^3) \] (15)

Due to the short time in the actual interpolation process, the normalizability of \( O((t - t_i)^3) \) can be explained. Its value is considered to almost zero, and the form of discretization is adopted here. If take \( t = t_{i+1} \), and the complement period is substituted \( T_i = t_{i+1} - t_i \), we can get:

\[ u(t_{i+1}) = u(t_i) + u'(t_i)T_i + \frac{u''(t_i)T_i^2}{2!} \] (16)

In the above formula, it is required to take the first derivative and the second derivative of \( u \) with respect to \( t \), and it is also known:

\[ V(u) = \left\| \frac{dF(u)}{dt} \right\| = \left\| \frac{dF(u)}{du} \right\| \frac{du}{dt} \]

Based on the above, you can get:

\[ \frac{du}{dt} = \frac{V(u)}{\left\| \frac{dF(u)}{du} \right\|} d^2u = \frac{-V^2(u)}{\left\| \frac{dF(u)}{du} \right\|^2} \left[ \frac{dF(u)}{du} \cdot \frac{d^2F(u)}{d^2u} \right] \]

In summary, the robot motion interpolation formula can be obtained:

\[ u_{i+1} = u_i + \frac{V(u)T_i}{\left\| F'(u) \right\|^2} \left[ F'(u) \cdot F'(u) \right] \frac{1}{2 \left\| F'(u) \right\|^2} \] (17)

4. Simulation and analysis

In order to verify the effectiveness of the above-mentioned NURBS curve fitting and NURBS interpolation algorithms, the above-mentioned algorithms are verified in MATLAB. At the same time, in order to explain the performance of the algorithms, a comparison is made with the aid of the piecewise S-shaped straight line planning algorithm. As shown in the figure, it is the experimental interpolation point and the weight at that point.
Table 2. Interpolation points and weights

| Serial number | Type point          | unit | Weight |
|---------------|---------------------|------|--------|
| 1             | (0,0,317.2306) mm   |      | 1      |
| 2             | (40,56,317.2306) mm |      | 1      |
| 3             | (10,50,317.2306) mm |      | 1      |
| 4             | (0,68,317.2306) mm  |      | 1      |
| 5             | (-10,50,317.2306) mm|      | 1      |
| 6             | (-40,56,317.2306) mm|      | 1      |
| 7             | (0,0,317.2306) mm   |      | 1      |

Due to the electrical constraints of the robot, the trajectory planning must follow this condition. The constraints of the robot are shown in Table 3.

Table 3. Description of simulation constraints

| Restrictions          | value   | unit      |
|-----------------------|---------|-----------|
| Interpolation period Ts | 5       | ms        |
| Maximum linear speed Vmax | 0.1    | mm/ms     |
| Maximum acceleration Amax | 0.0005 | mm/ms²    |
| Maximum jerk Jmax     | 0.0005  | mm/ms³    |

Figure 3 shows the trajectory planning and interpolation results in the three-dimensional space of the robot. It can be seen that the traditional segmented S-shaped planning algorithm and the NURBS planning algorithm have completed the task better through the interpolation points, but the NURBS curve planning path is smoother and has better transition at the interpolation point, and the transition speed of the segmented S-shaped trajectory changes too fast at the interpolation point.

Figure 4 shows the speed comparison results of the two trajectory planning algorithms. It can be seen that the speed of the NURBS curve is continuously smooth, and there is no speed of 0 in the middle. The average speed of the overall movement process is larger of the NURBS trajectory planning algorithm, which is faster and more rapid than the segmented S-type planning, and it can complete the task well.
Conclusion

This paper proposes a NURBS trajectory planning algorithm based on multi-degree-of-freedom parallel robot, which uses the powerful modeling ability of NURBS curves and simple mathematical forms to deal with the trajectory planning problem of complex curves. Compared with the traditional segmented interpolation algorithm, this algorithm does not need to control the frequent start and stop of the motor during the segment-to-segment switching process. It solves the problem of frequent speed fluctuations and zero fluctuations. It also realizes the smooth and continuous operation of the robot during the movement. Finally, it improves the robot exercise efficiency and overall stability.

References

[1] Mihelj M, Bajd T, Ude A, et al. (2019) Trajectory Planning. Robotics. Springer International Publishing, Switzerland, pp. 123-132.

[2] Li Y, Huang T, Chetwynd D G. (2018) An approach for smooth trajectory planning of high-speed pick-and-place parallel robots using quintic B-splines. Mechanism and Machine Theory, 126: 479-490.

[3] Zhang W, Zhao X. (2003) Position analysis of the 3-RTT parallel manipulator. Journal of Tianjin University of Technology, 19(3): 32-35.

[4] Yu X, Tang X, Ye B, et al. (2016) Time-optimal path tracking for articulated robot along spline curves based on master-slave strategy. 2016 International Conference on Control, Automation and Information Sciences (ICCAIS). In: South Korea. pp. 149-154.

[5] Huang J, Hu P, Wu K, et al. (2018) Optimal time-jerk trajectory planning for industrial robots. Mechanism and Machine Theory, 121: 530-544.

[6] Atkeson C G, Hollerbach J M. (1985) Kinematic features of unrestrained vertical arm movements. Journal of Neuroscience, 5(9): 2318-2330.