Mechanics & Industry **21**, 107 (2020)
© AFM, EDP Sciences 2019
https://doi.org/10.1051/meca/2019044

Mechanics & Industry

Available online at:
www.mechanics-industry.org

REGULAR ARTICLE

Planar tool radius compensation for CNC systems based on NURBS interpolation

Jiangang Li*, Qian Wang, and Ganggang Zhong

Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, PR China

Received: 28 March 2019 / Accepted: 6 September 2019

Abstract. This paper introduces the realization of a tool radius compensation algorithm for NURBS trajectory. First, a single-segment NURBS trajectory tool radius compensation algorithm is developed. Different from the straight line and arc trajectory, the self-intersection phenomenon is prone to happen when calculating a single NURBS tool center trajectory, and the self-intersection will cause the overcut of workpiece. To avoid this situation, the algorithm introduced in this paper can detect whether the NURBS tool center track has caused overcut, and deal with the self-processing. Second, the tool radius compensation algorithm with multi-segment NURBS trajectory is implemented. The focus of this part is the tool radius compensation of the trajectory transfer, and the trajectory transfer is divided into two types: the extension type and the shortened type. For the shortened type transfer, cross-processing is needed to avoid the overcut of workpiece at the transfer. When calculating the tool radius compensation of the shortened type, we not only need to find the intersection of the tool center trajectory of two adjacent NURBS curves, but also need to select the intersection we need when a number of intersections exist. For the extension type transfer, in order to ensure the continuity of the tool center trajectory, we need to extend the tool center trajectory or add arc-segment at the transfer. The proposed algorithm can automatically decide where to extend the tool center trajectory or add arc-segment to achieve the best efficiency. Finally, the algorithm can output the calculated NURBS tool center trajectory in the form of linear segment interpolation G code or NURBS interpolation G code according to the processing needs. Simulations on VERICUT and experiments on three-axis CNC machine tool shows the effectiveness and validation of the tool path compensation algorithm.

Keywords: Tool radius compensation / CNC system / NURBS trajectory / machine tool

1 Introduction

CAM system has been able to use NURBS curve to describe contour trajectory. NURBS interpolation has also been studied [1–3] and seems to work well. Many high-end systems have been supporting NURBS curve interpolation, such as FAUNC, Siemens and other manufacturers. However, these CNC systems do not support tool radius compensation for flat NURBS. Generally, the NURBS isometric curve is approximated by numerical approximation in engineering practice [4]. In this paper, we focus on the NURBS tool radius compensation problem for tool geometry path rather than the speed planning and interpolation of NURBS trajectories studied in previous papers.

For that the NURBS isometric curve has applications in many aspects of actual processing, many scholars have conducted research on the calculation of the distance curve. Tiller et al. [5] first offset each edge of the control polygon of original curve to obtain the control polygon of the equidistant curve and then used the resulting control polygon to approximate the equidistant curve of the original curve. Coquillart [6] also calculated the equidistant curve by offsetting the control polygon. Unlike in [5,6] offset the vertices of the polygon to obtain a new set of control points, and then used the control polygon composed by these new control points to approximate the isometric curve. Hoscheck [7] studied the spline approximation of isometric curve mainly based on parameter optimization and geometric continuity. Klass [8] approximated the isometric curve through the Hermite curve. Pham [9] first calculated the corresponding equidistant curve points according to the sampling points on the original curve, and then obtained the control points of the equidistant curve by fitting these equidistant points with NURBS curve. Lee et al. [10] first approximated the circumference with a certain radius with a quadratic Bezier curve, then used the approximation of the resulting circular curve to sweep along the original curve, and finally expressed the approximation of the equidistant curve as the envelope swept by circular curve. Liu et al. [11] calculated the offset vector of each vertex of the control polygon mainly according to

* e-mail: jiangang_lee@163.com

the normal vector curve approaching the original curve, then obtained the new control polygon, and finally calculated the optimal approximation of the equidistant curve using the control polygon. Zhang et al. [12] proposed an approximation algorithm of equidistant curve based on Bezier curve of dual basis. Jiang et al. [13] proposed a specific method of trajectory recognition and transition processing, but too many partial calculations make the algorithm too complicated. The tool-adaptive offset path based on triangular meshes was proposed by Zhou et al. [14] and Kout [15] improved the algorithm with increased nodes and transition paths, but without specific criteria given in error control. Cai et al. [16] gave an improved algorithm for the generation of equidistant curves, which can effectively remove the self-intersection of the equidistant curve but does not give a concrete approach to the control of the error.

In the NURBS trajectory tool radius compensation algorithm, we not only need to consider the calculation of the NURBS equidistant curve, but also need to consider the Intersection of NURBS equidistant curve.

The function of tool radius compensation is to calculate the tool center trajectory, and the tool center trajectory can be regarded as an equidistant curve of the contour trajectory in differential geometry. Therefore, the study of NURBS trajectory tool radius compensation can not be separated from the study of its isometric curve.

The organizational structure of this paper is as follows: Section 1 introduces the state of the art of NURBS equidistant curve; Section 2 presents the tool radius compensation algorithm for single segment NURBS trajectory; Section 3 explains the tool radius compensation algorithm with multi-segment NURBS trajectory; Section 4 discusses the NURBS curve approximation of the NURBS equidistant curve; Section 5 shows simulation and experimental results; and Section 6 concludes this paper. Figure 1 shows the overall flow of the tool radius compensation algorithm.

2 Tool radius compensation algorithm for single NURBS trajectory

Set the tool radius to be R, then the expression of the original flat NURBS curve is as follows:

$$\begin{cases} x(u) = \frac{\sum_{n=0}^{N} a_n w_i x_i N_{i,p}(u)}{\sum_{n=0}^{N} a_n w_i N_{i,p}(u)} \\ y(u) = \frac{\sum_{n=0}^{N} a_n w_i y_i N_{i,p}(u)}{\sum_{n=0}^{N} a_n w_i N_{i,p}(u)} \end{cases}. \quad (1)$$

The expression of equidistant curve $C_R(u) = (x_R(u), y_R(u))$ (the default is an internal equidistant curve) is as follows:

$$\begin{cases} x_R(u) = x(u) - R\frac{y\prime(u)}{\sqrt{x\prime(u)^2+y\prime(u)^2}} \\ y_R(u) = y(u) + R\frac{x\prime(u)}{\sqrt{x\prime(u)^2+y\prime(u)^2}} \end{cases}. \quad (2)$$

The denominator of the NURBS equidistant curve expression contains the root operation, which leads to the result that the NURBS equidistant curve cant be directly expressed as a reasonable form of the original curve. Therefore, only a series of NURBS equidistant position information can be obtained according to equation (2). Fortunately, these isometric points are not isolated data points, so they carry a lot of geometric information. According to the nature of a planar isometric curve, we can see that the coordinate value, the unit normal vector and the relative radius of curvature of any point on the isometric curve can be accurately determined if the parameter expression and the tool radius of the original curve are known.

In the differential geometry, there is no self-intersection phenomenon for the straight lines and arcs, as shown in Figure 2a and b. However, it is possible for NURBS equidistant curve to contain self-intersection, which can lead to overcut of workpiece. To avoid this phenomenon, we need to calculate the self-intersection of the NURBS equidistant curve and then remove the trajectory segments which will lead to overcut. To simplify the calculation of searching the intersection of NURBS equidistant curve, the equidistant curve needs to be discretized. In order to facilitate the processing of CNC machine tools, the NURBS equidistant curve eventually needs to be expressed in a G code form. Taking into account the actual processing needs, NURBS tool center track is finally expressed in two G code forms in our algorithm, respectively the straight line interpolation G code and NURBS interpolation G code. Therefore, the following two points need to be ensured during the discretization of NURBS equidistant curve: discretized small lines need to be smooth to ensure the processing accuracy; and discretized small segments should be as few as possible to improve the efficiency of NURBS tool. To obtain the discrete points of the equidistant curve, firstly we need to discrete the original NURBS curve.

2.1 Discretization of the NURBS curve

Since the newly obtained curve expression which obtained by tool radius compensation through normal equidistance is not a NURBS or an explicit curve form. Therefore, in this paper, the method calculates the normal equidistant point and then fits it into a curve. In order to ensure the smoothness of discrete small segments, it is necessary to control the chord error and deflection angles of discrete small segments in a given range. However, if there are too many discrete small segments, it is bound to reduce the efficiency of the intersection of NURBS. Therefore, considering both smoothness and simplicity, this paper proposes an adaptive discretization algorithm.

First, let us analyze the relationship among the radius of curvature, the chord difference and the deflection angle, as shown in Figure 3.

In a circle, the radius of curvature is constant, and if either of the chord and deflection angles is known, then the other can be figured out. According to Figure 3, the relationship among the chord error, deflection angle and the radius of curvature can be established as

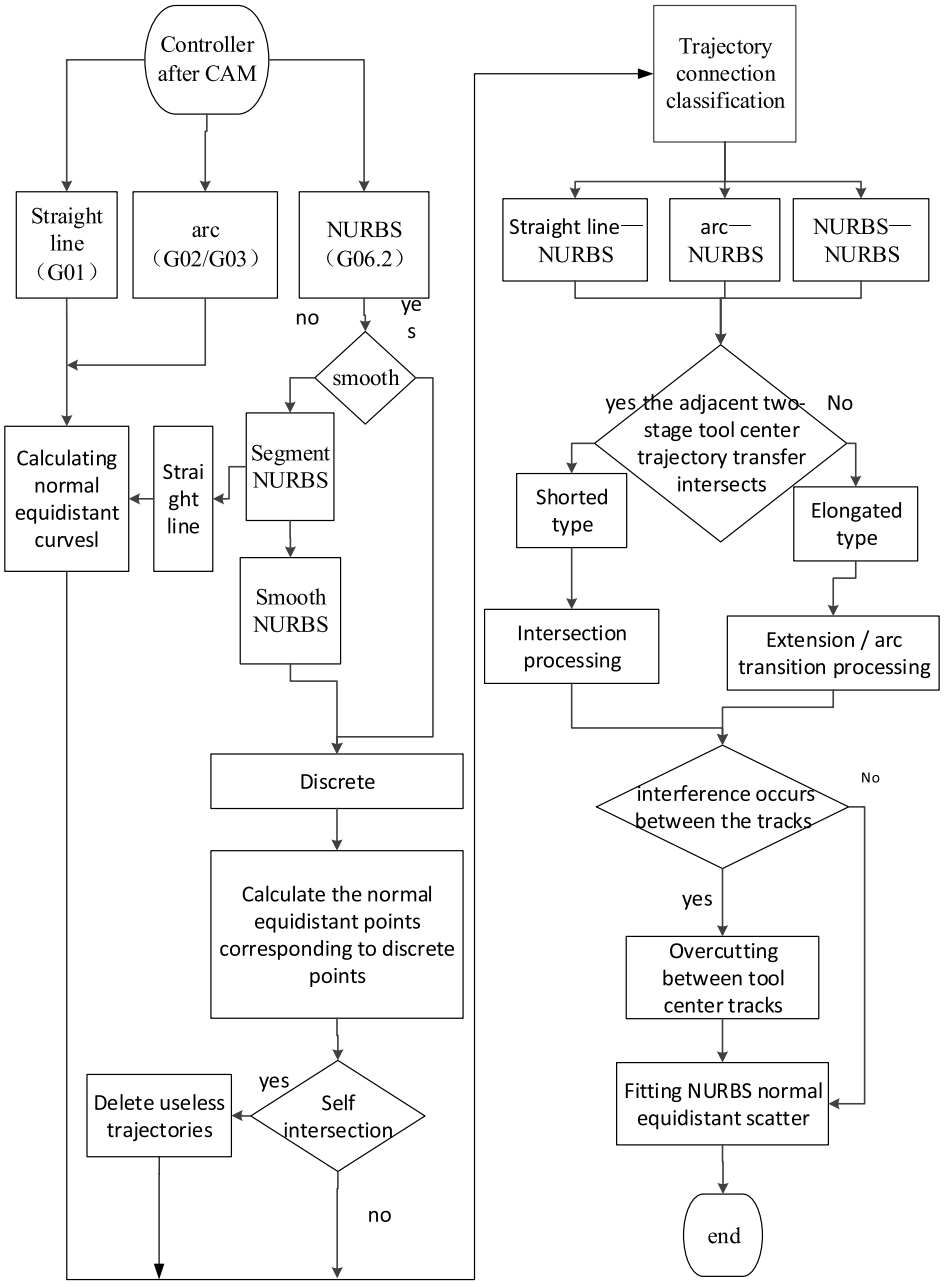


Fig. 1. Flow chart of tool radius compensation algorithm for complex contour trajectory.

the following equation

$$R = \frac{E}{1 - \sin(\frac{\pi - \alpha}{2})} \tag{3}$$

where R is the radius of curvature (mm), E is the chord error (mm), and α is the deflection angle (rad).

Also in a circle, the relationship among the chord length, the chord difference and the deflection angle can also be established, as shown below:

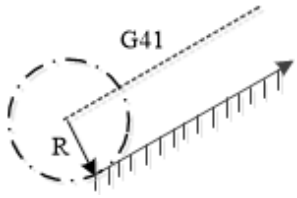
$$L = \frac{2E\sin(\frac{\alpha}{2})}{1 - \sin(\frac{\pi - \alpha}{2})}. \tag{4}$$

Here, we can calculate the chord length of the next discrete point by equation (4) according to the chord and deflection angle of the last discrete point. However, the focus is how to convert the chord length L to the parameter increment Δu. The specific algorithm is shown below.

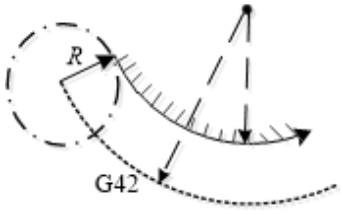
Suppose that $P(u_i)(x(u_i), y(u_i))$ is a point of the NURBS curve, $P(u_{i+1})(x(u_{i+1}), y(u_{i+1}))$ is the next point after the discretization. Then, we have

$$\Delta L = \parallel P(u_{i+1}) - P(u_i) \parallel \tag{5}$$

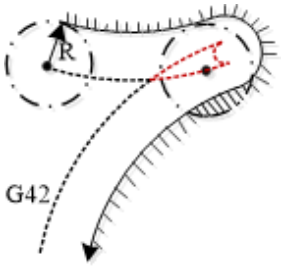
$$= \sqrt{(x(u_{i+1}) - x(u_i))^2 + (y(u_{i+1}) - y(u_i))^2}. \tag{6}$$



(a) Tool radius compensation for straight line



(b) Tool radius compensation for arc curve



(c) Tool radius compensation for NURBS curve

Fig. 2. Tool radius compensation algorithm for single trajectory.

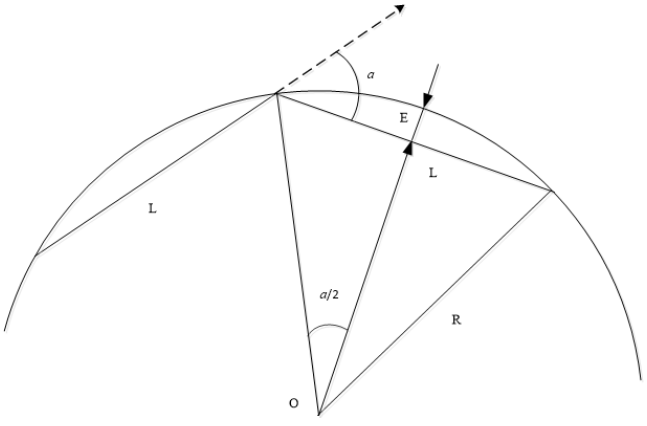


Fig. 3. Chord length, deflection angle and radius diagram in a circle.

We set $u_{i+1} = u_i + \Delta u$, then the following relationship can be obtained according to Taylor expansion:

$$x(u_{i+1}) = x(u_i) + x\prime(u_i)\Delta u + 0.5o(\Delta u^2) \tag{7}$$

$$y(u_{i+1}) = y(u_i) + y\prime(u_i)\Delta u + 0.5o(\Delta u^2) \tag{8}$$

Δu is very small, so we can remove the high order items and obtain

$$x(u_{i+1}) = x(u_i) + x\prime(u_i)\Delta u \tag{9}$$

$$y(u_{i+1}) = y(u_i) + y\prime(u_i)\Delta u \tag{10}$$

Substituting equation (8) and equation (9) into equation (5), we obtain

$$\Delta u = \frac{\Delta L}{\sqrt{x\prime(u_i)^2 + y\prime(u_i)^2}}. \tag{11}$$

When the curve is discrete, the given chord and deflection angle values should be set as discrete parameters. Substituting the two discrete parameters into equation (3), we can find out the corresponding radius of curvature which is called the critical radius of curvature R_{cri}.

Since the radius of curvature of the curve is not a fixed value, in order to obtain the chord length from the next discrete point, it is necessary to obtain the corresponding chord and deflection angle first. Therefore, the curvature radius value of the discrete points on the curve is first calculated before the curve is discretized, and then compared to the critical radius of curvature R_{cri}. We set the given chord difference as E_0, and the given deflection angle as α_0. In order to obtain the chord length corresponding to the next discrete point, it is necessary to determine the values of the chord error and deflection angle corresponding to this chord length. We consider the following three cases:

Case 1: $R > R_{\text{cri}}, E = E_0, \alpha < \alpha_0$
Case 2: $E < E_0, \alpha = \alpha_0$
Case 3: $R = R_{\text{cri}}, E = E_0, \alpha = \alpha_0$

Since the probability of the third case is very small, the third and first cases can be merged, which does not affect the results of the calculations. Finally, the obtained E and α can be taken into equation (4) to get the next step chord value, then we can obtain Δu according to Taylor expansion. Therefore, the NURBS curve can be discretized, and the specific steps of the algorithm are as follows:

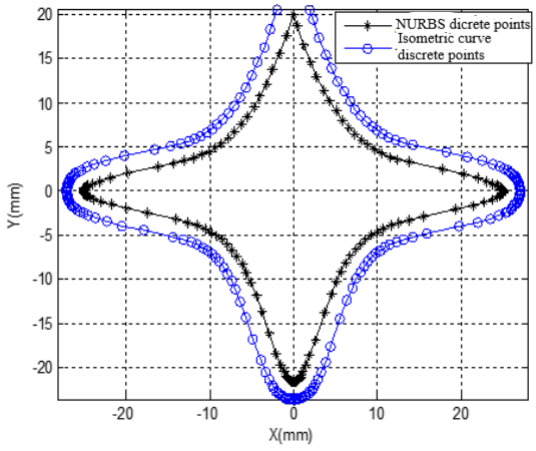
Step 1: Calculate the radius of curvature R_i of a point $P_i(i = 0, 1, 2, \ldots, n)$ on the NURBS curve and then compare to the critical radius of curvature R_{cri}. If $R_i > R_{\text{cri}}$, then go to step (2), otherwise, go to step (3).

Step 2: Calculate the curvature radius R_i corresponding to the chord error E_i and deflection angle α_i, and set $E_i{=}E_0$, $\alpha_i{=}\pi - 2\text{acrsin}(1 - E_0/R_i)$. Then go to step (4).

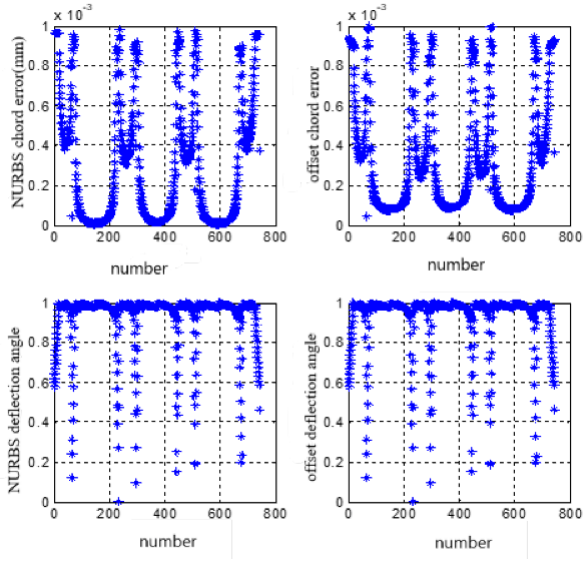
Step 3: Calculate the chord error E_i and deflection angles α_i corresponding to the radius of curvature R_i, and set $\alpha_i{=}\alpha_0, E_i{=}R_i(1 - \sin((\pi - \alpha_0)/2))$. Then go to step (4).

Step 4: Calculate the chord length L_i from the next discrete point according to equation (4), and then find out the corresponding parameter step Δu_i according to equation (10), and the next discrete point parameter is obtained by $u_i = u_i + \Delta u_i$. Then go to step (5).

Step 5: Examine whether or not the actual chord difference and the deflection angle value corresponding to the obtained discrete points satisfy the condition. If not, let it satisfy the given range by shortening the parameter step. Then go to step (6).



(a) NURBS and its isometric curve



(b) The deflection angle and chord error of NURBS and its isometric curves

Fig. 4. Star NURBS curve and its isometric curve discrete simulation.

Step 6: Examine whether $u_i \leq 1$ is satisfied. If yes, then go to step (1), otherwise stop the discretization process.

We can get NURBS scattered points that meet the requirement according to the above adaptive algorithm. Set the chord error $E_0 = 0.001$ mm, deflection angle = 1 degree, and tool radius = 2 mm, Discrete MATLAB simulation is shown in Figure 4a, The deflected angle and the chord error after discretization is shown in Figure 4b, in which the discrete actual chord and deflection angles are within the set range.

2.2 Self-dealing of NURBS equidistant curve

In the process of calculating the NURBS equidistant curve, if the relative curvature of a part of the original curve is opposite to the relative curvature of its isometric curve, then the equidistant curve will contain self-intersection [17,18], which is called the first type self-intersection in this paper, as shown in Figure 5. If the

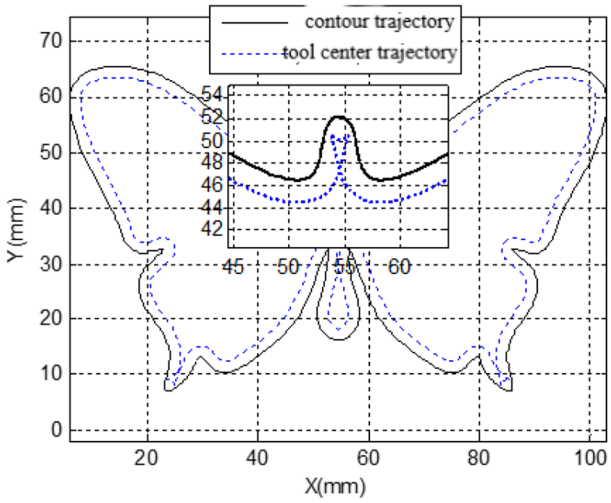


Fig. 5. First type of self-intersection of NURBS normal equidistant curve.

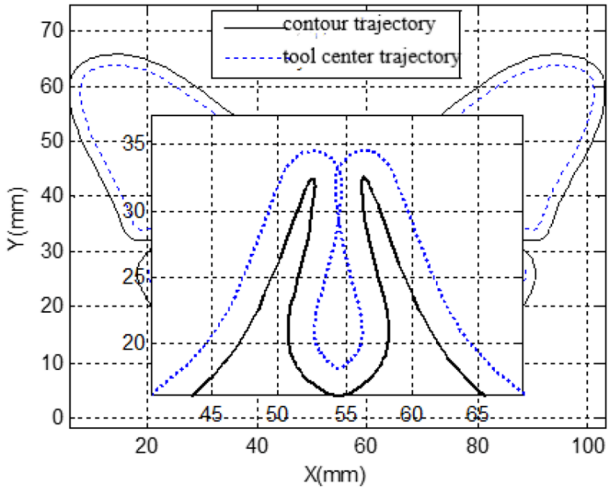


Fig. 6. Second type of self-intersection of NURBS normal equidistant curve.

original curve trajectory contains a "gourd mouth" shape, then the self-intersection phenomenon will also probably occur in this area, which is called the second type self-intersection in this paper, as shown in Figure 6. When calculating the tool center trajectory of the NURBS contour trajectory, if we fail to identify above two types of self-intersection phenomenon,then it may lead to the overcut of workpiece.

Before eliminating the self-intersection part of NURBS equidistant curve, we should first identify which part of the trajectory will lead to the overcut of workpiece and find out the intersection point of the effective trajectory and the invalid trajectory. For the case that each trajectory is continuous from the starting point to the end point, both types of self-intersection have a same feature, i.e., the trajectory part which leads to overcut is a ring circuit. As shown in Figures 7 and 8, both the starting point and the end point of the NURBS equidistant trajectory corresponding to curve $r_2(t)$ are the same point O. Here, we call this loop track as self-intersection loop track.

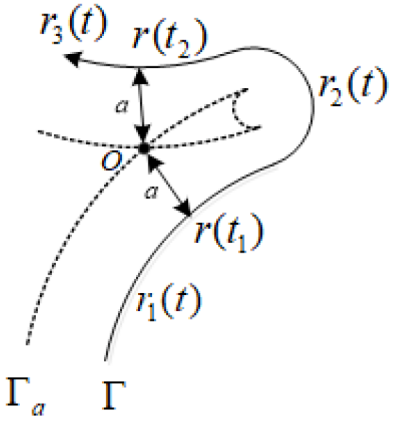


Fig. 7. First type of self-intersection.

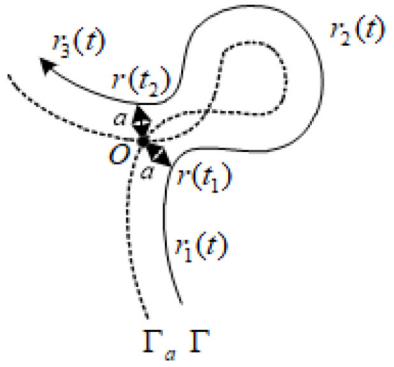


Fig. 8. Second type of self-intersection.

And the key point to deal with self-intersection is to identify the self-intersection loop track and eliminate it. After we deal with the self-intersection phenomenon, the actual effective tool center trajectory becomes continuous.

No matter which type of self-intersection, before we deal with it, we first need to find out the intersection point. To facilitate the process of finding the intersection point of NURBS equidistant curve, a series of small straight lines are used to describe the NURBS equidistant curve. Therefore, it is vital in our algorithm of dealing with the self-intersection phenomenon to find which two straight lines intersect with each other, and then find the intersection. We first analyze the relative position of any two straight segments on the plane, then the necessary condition for the intersection point of any two straight segments is that both the intersections of their x and y fields are not null. With this condition, we can quickly locate the position of the intersection point.

When self-intersection occurs in NURBS equidistant curve, the intersection point can be determined by the following steps:

Step 1: Store data of the discrete points and their corresponding equidistant points into the array container;

Step 2: Let $Line_i = C_d(u_i)C_d(u_{i+1})$ $(i = 0, 1, \ldots, n-1)$ be a reference line segment, and $Line_i = C_d(u_i)C_d(u_{i+1})$ $(j = i + k, k = 1, 2, \ldots, n-1-i)$ be a line segment for comparison, and compare the first baseline line segment with all subsequent straight lines and examine if there is an intersection: if not, then continue to detect if there is an intersection for the next reference line segment; otherwise, record the bit of the line segment where the intersection is, and then by the infinite separability of equidistant points and the binary iterative algorithm, we can compute the intersection of the equidistant curve and the corresponding parameter u_n and $u_m(u_n < u_m)$ of the original NURBS curve with a very high precision.

After determining the position of the line segment where the intersection is located and its corresponding original NURBS curve parameters u_n and u_m, we also need to determine which side of the intersection is the effective trajectory, and which side needs to be removed. As shown in Figure 9, the intersection of NURBS equidistant curves is the point O, which is the intersection of equidistant curve segments $C_d(u_i)C_d(u_{i+1})$ and $C_d(u_i)C_d(u_{i+1})$, and the corresponding original NURBS curve segments are $C(u_i)C(u_{i+1})$ and $C(u_j)C(u_{j+1})$. And points $C(u_n)$ and $C(u_m)$ on the original NURBS curve correspond to the intersection point O. From the direction of the original curve we can obtain that $u_i \le u_n \le u_{i+1} < u_j \le u_m \le u_{j+1}$. For the convenience of analysis, we can divide the curve $C(u)$ to three segments by points $C(u_n)$ and $C(u_m)$, with the sub-curves $C_1(u)$, $C_2(u)$ and $C_3(u)$.

According to the above analysis, the steps to judge which trajectory segment should be removed are as follows:

Step 1: Calculate the two tangent vectors T_n and T_m of the intersection O, which are respectively equal to the tangent vectors of points $C(u_n)$ and $C(u_m)$ of the original NURBS curve. Step 2: If the dot product of vector $\overrightarrow{OC(u_n)}$ and tangent vector T_m is negative and the dot product of vector $\overrightarrow{OC(u_m)}$ and tangent vectors T_n is positive, then we should eliminate the equidistant curve corresponding to the original NURBS curve whose parameter satisfies $u \in (u_n, u_m)$.

Above all, to achieve tool radius compensation for single-segment smoothing NURBS curve, we not only need to calculate the equidistant curve of original NURBS curve, but also need to detect if the equidistant curve contains self-intersection phenomenon, and we need to do different treatments for different self-intersection types. This is the biggest difference between it and tool radius compensation algorithm for straight line and arc trajectory.

The contour trajectory of Figure 10 is a 3-order NURBS curve. Suppose that the tool radius is 2mm, we do right tool radius compensation for it, and the resulting tool center trajectories is shown in Figure 10a. Figure 10b shows tool center trajectories after the treatment of self-intersection phenomenon. Comparing these two figures, the tool radius compensation algorithm for single smooth NURBS curve segment introduced in this section is proved to be effective.

3 Tool radius compensation with multiple NURBS segments

The overall profile of the machined part is usually made up of multi-segment trajectories, so the compensation

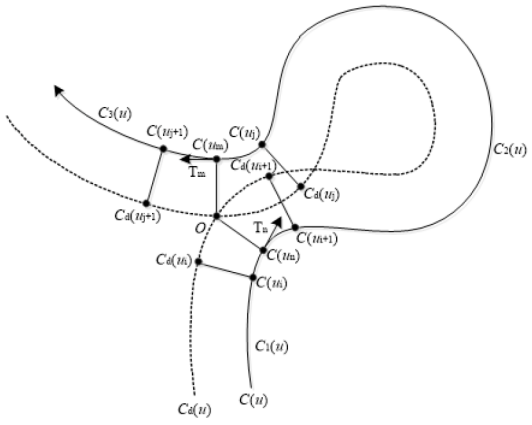
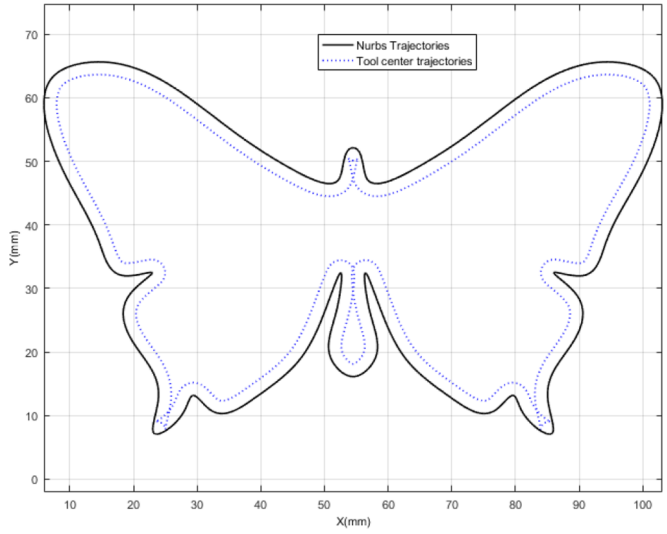
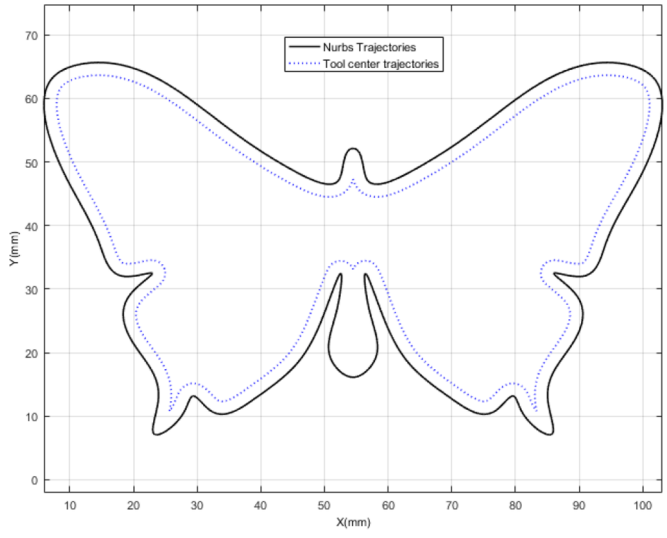


Fig. 9. Self-intersection loop track.

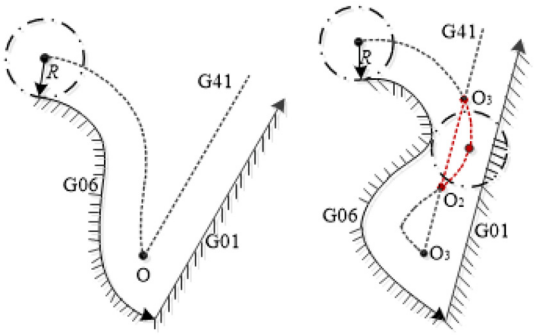


(a) Equidistant NURBS curve before the self-intersection treatment



(b) Equidistant NURBS curve after the self-intersection treatment

Fig. 10. Order butterfly NURBS equidistant trajectory self-processing simulation diagram.



(a) One intersection exists (b) several intersections exist

Fig. 11. Schematic diagram of shortened type tool radius compensation for NURBS trajectory.

between the adjacent tracks is very important in the tool radius compensation algorithm. For multi-segment contours, since the generated tool center trajectory is not continuous at the transition point, the transfer process is required. For shortened type of transfer, in order to avoid the overcut at the transfer segment, we need to find the intersection point. And for elongation type of transfer, to ensure continuity at the transfer segment, the extension or arc transition is needed.

3.1 Treatment of two types of transfer

Both the elongation type and the shortening type are determined based on the vector deflection angle and the tool compensation direction of the adjacent trajectories [19]. As the method is now mature and widely used, so we do not repeat it here. The main work of this section is to improve the existing algorithm.

3.2 The shortened type tool radius compensation

For straight segments and arc trajectories, there is only one intersection at the transition, the shortened type tool radius compensation calculation for these two trajectories is relatively simple. Nevertheless, when calculating the shortened type tool radius compensation for the NURBS trajectory, it is also necessary to accurately identify the intersection points required for the calculation to avoid the overcut if there are several intersections at the transition, as shown in Figure 11.

As shown in Figure 11b, it is obvious in this case that the intersection O_3 is needed for the tool radius compensation calculation, but not O_1 nor O_2. In order to improve the efficiency of the algorithm, we have to find the required intersection accurately in the shortest time to simplify the calculation. It can be seen from the third section that the simplification method is also needed when the NURBS equidistant curve is represented by a series of small line segments. When calculating the shortened type tool radius compensation for NURBS curve, the process can be divided into two cases according to the position of the trajectories.

The first case: the previous track is a NURBS curve, as shown in Figure 12 left. In order to find the intersection

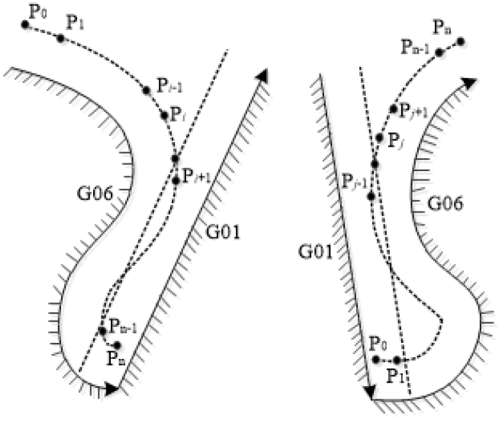
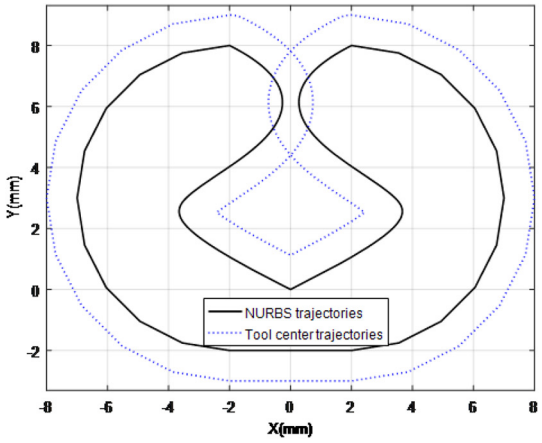
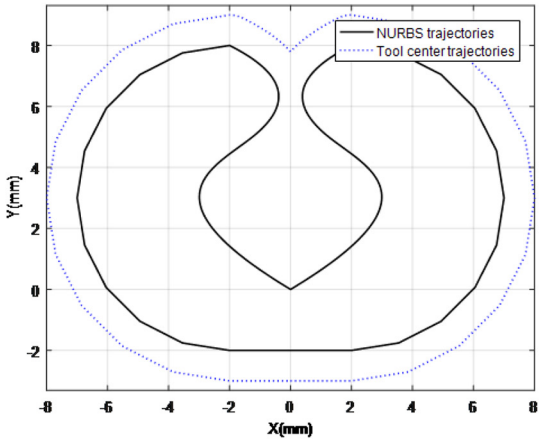


Fig. 12. Searching order to find the required intersection for NURBS curve.



(a) Tool center trajectory without positive or reverse intersection searching algorithm



(b) Tool center trajectory with positive or reverse intersection searching algorithm

Fig. 13. Tool center trajectory at shortened type transfer segment for the adjacent NURBS curve trajectories.

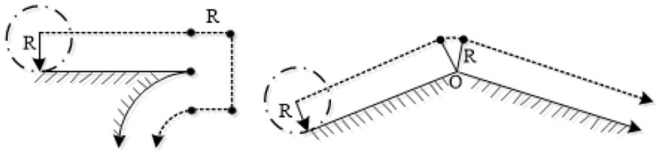


Fig. 14. Special situations for elongation transfer type.

accurately required for the shortened tool radius compensation, it is necessary to make a positive sequence search for the small line segments consisting of the NURBS equidistant points. That is, for the equidistant points set P_i $(i = 0, 1, \ldots, n)$, first, we need to determine whether there is an intersection point for small line segment P_0P_1 and the equidistant curve of the next trajectory. If the intersection does not exist, we need to go on detecting the next small line segment in accordance with the positive sequence. Otherwise, we need to locate the intersection and calculate the accurate corresponding parameter u for the intersection according to the infinite separability and the binary iterative search method for equidistant points, and then eliminate the trajectory segment which will lead to overcut.

The second case: the latter trajectory is the NURBS curve, as shown in Figure 12 right. For the equidistant points set P_i $(i = 0, 1, \ldots, n)$, first we need to determine whether there is an intersection point for small line segment P_nP_{n-1} and the equidistant curve of the previous trajectory. If the intersection does not exist, we need to go on detecting the next small line segment in accordance with the reverse sequence. Otherwise, we need to locate the intersection and calculate the accurate corresponding parameter u for the intersection according to the infinite separability and the binary iterative search method for equidistant points, and then eliminate the trajectory segment which will lead to overcut.

When calculating the equidistant curve at shortened type transfer segment for the adjacent NURBS curve trajectories, it may lead to the overcut of workpiece as shown in Figure 13a if the aforementioned positive or reverse intersection searching algorithm is not used. We can also avoid this situation by using a tool with a smaller tool radius.

3.3 Tool radius compensation for elongation type transfer segment

For elongated tool radius compensation, the existing processing method only considers the geometric information of the trajectory [19,20], but ignores some parameter information given by the actual machine processing, such as turning time, tool radius, maximum acceleration for machine tools, and so on. If the information is taken into account when calculating the tool radius compensation, the efficiency of the machining can be improved. For elongation transfer segment, we usually extend the tool center track as shown in Figure 14a to avoid a too long air machining distance. The method is easy to implement, but it will bring two corners in some cases which are ought to reduce the processing efficiency. Another method is to plug in an arc segment, but when the radius of the transition arc is very small as shown in Figure 14b, the machine will run at a very low speed when machining the additional track, which can also lead to a reduction in processing efficiency.

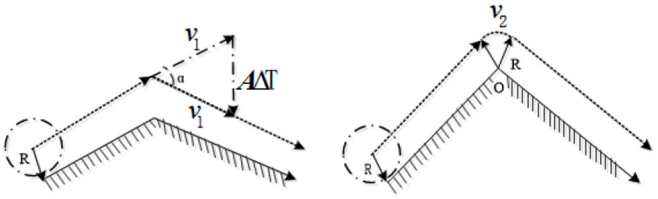


Fig. 15. Transition processing for elongation transfer type.

In summary, always using any of the above methods is impractical when calculating the tool center track for elongation transfer type, and the idea in this paper is that we should determine which method introduced above to adopt according to the deflection angle of adjacent tracks α, the tool radius R, the maximum acceleration of machining A, and the turn time is δT.

Suppose that the maximum acceleration of machining is, the turning time is. For extending tool center track method as shown in Figure 15a, the relationship among the turning speed, the acceleration and the turning time can be described as the following equation:

$$v_1 = \frac{\Delta T A}{2sin(\frac{\alpha}{2})} \tag{12}$$

where v_1 is the turning speed when extending tool center track (mm/s), ΔT is turning time (s), A is the maximum acceleration for machining (mm/s^2), α is the deflection angle of adjacent tracks (rad).

When adopt arc transition as shown in Figure 15b, the relationship among the turn speed v_1, the acceleration v_1 and the turn time ΔT can be described as the following equation:

$$v_2 = \sqrt{AR} \tag{13}$$

where v_2 is the turning speed when adopt arc transition (mm/s), A is the maximum acceleration for machining (mm/s^2), R is the tool radius (mm).

Which elongation transition method to choose is determined by the deflection angle between adjacent contour tracks. Then, we can obtain the critical deflection angle by combining equation (11) and equation (12):

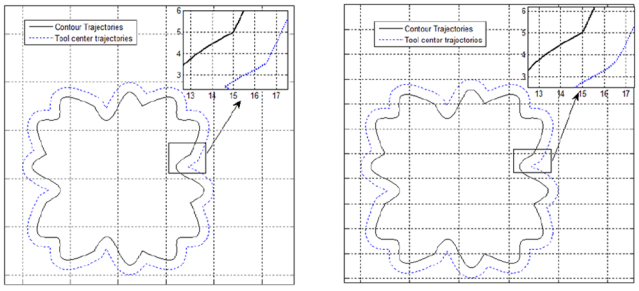
$$\alpha_{\text{cri}} = 2arcsin(\frac{\Delta\sqrt{AR}}{2R}). \tag{14}$$

Comparing α to α_{cri}, the situation can be divided to two cases:

Case 1: $\alpha < \alpha_{\text{cri}}$, the deflection angle of adjacent tracks is relatively small, and to let the turning speed be as big as possible, we choose the extending tool center method.

Case 2: $\alpha \geq \alpha_{\text{cri}}$, the deflection angle of adjacent tracks is relatively large, and we choose the arc transition method.

The contour trajectory in Figure 16 consists of a series of NURBS curves, the validity of the elongation type transfer processing algorithm is verified by the adjacent NURBS curves. In Figure 16a, the tool radius is 2 mm, the maximum acceleration of the machine is 200 mm/s^2, and the turning time is 0.5 s. In Figure 16b, the tool radius is 2 mm, the maximum acceleration of the machine is 200 mm/s^2, and the turning time is 0.02 s.



(a) Extending tool center method (b) Arc transition method

Fig. 16. Compare of two kinds methods

According to the comparison of Figure 16a and b, it can be seen that even if the tool radius and the maximum acceleration of the machine are the same for the same trajectory, the difference of the turning time will lead to the difference of the critical deflection angle, so as to change the processing method of the elongation transfer. Moreover, the maximum acceleration and the tool radius can also affect the critical deflection angle. Therefore, changes of these factors may all lead to the change of transition processing method.

4 The NURBS curve approximation of equidistant point

In the third and fourth sections, in order to facilitate the solution of the NURBS equidistant curve intersection problem, this paper describes NURBS equidistant curve with a series of equidistant points. The main task of this section is to denote the NURBS equidistant point in the corresponding NURBS curve form, which facilitates the output of the tool center track. Considering that these equidistant points have accurate curvature information, this section uses the feature point extraction method to express these equidistant points in the NURBS curve form. The algorithm of curve approximation based on feature point extraction can be found in [21,22].

When self-intersection occurs in the NURBS equidistant curve in Section 3, it is necessary to remove the invalid trajectory in the NURBS equidistant line in order to prevent the over-cut of the workpiece, and then the equidistant curve will contain sharp corners. As shown in Figure 17, the self-intersection point of the equidistant curve is the point O, which corresponds to the points $C(u_m)$ and $C(u_n)$ on the original curve, the original NURBS curve to three parts, respectively sub-curve $C_1(u)$, sub-curve $C_2(u)$, and sub-curve $C_2(u)$. And the equidistant curve corresponding to sub-curve $C_2(u)$ may lead to the over-cut of workpiece, thus the equidistant curve segment should be removed. In order to output the tool center trajectory in the form of NURBS curve, it is necessary to fit the equidistant curve segment corresponding to the two effective sub-curves $C_1(u)$ and $C_3(u)$ with NURBS curve.

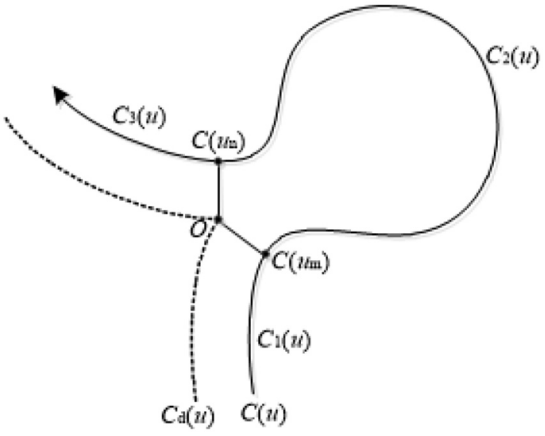


Fig. 17. Tool center trajectory after self-intersection processing for equidistant curve.

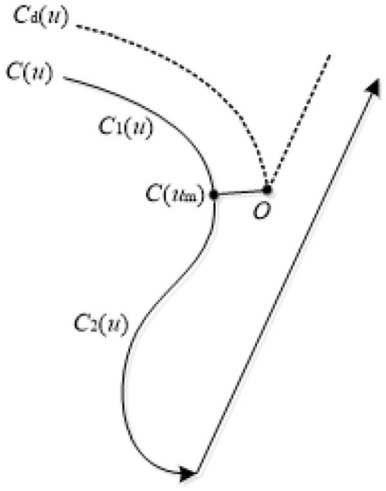


Fig. 18. Tool center trajectory after tool radius compensation with shortened type transition.

In Section 4, when the NURBS equidistant curve intersects with another NURBS equidistant curve of the adjacent trajectory at the shortened type transfer, to prevent the over-cut of the workpiece at the transition, it is necessary to remove the invalid equidistant NURBS track. As shown in Figure 18, the intersection corresponds to the point $C(u_m)$ on the original NURBS curve, which divides the original NURBS curve to two sub-curves, respectively $C_1(u)$ and $C_2(u)$. We need to remove the equidistant curve segment corresponding to $C_2(u)$, and then fit the equidistant curve segment corresponding to $C_1(u)$ with NURBS curve.

In this paper, we use the approximation of two sets of NURBS equidistant points to illustrate the validity of the curve approximation algorithm based on feature point extraction. The first set of NURBS equidistant points does not have sharp corners, so just one curve is needed to approximate the equidistant points. As shown in Figure 19, there are 2997 equidistant points, and the number of characteristic seed points is 63. In the approximation algorithm, the number of characteristic seed points is the same as the number of control points

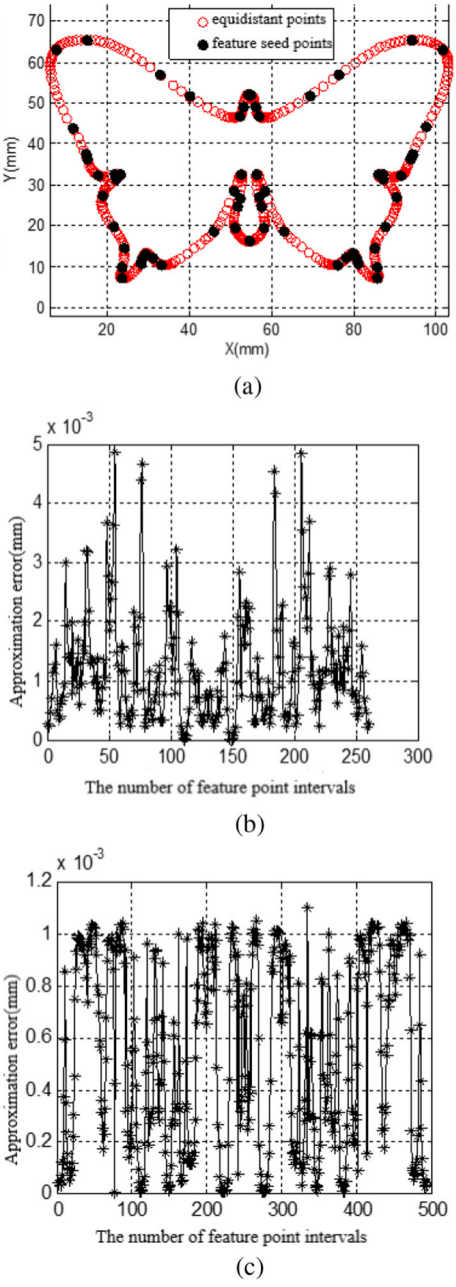


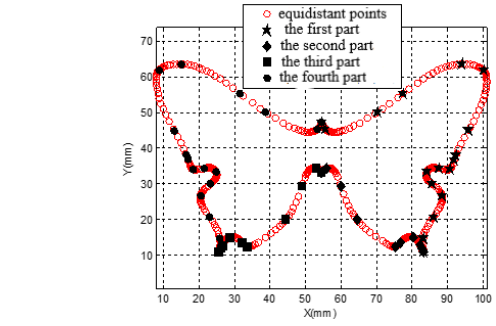
Fig. 19. Distribution of the feature seed points in the equidistant point (a), $\delta = 0.005$ mm actual approximation error distribution (b), $\delta = 0.001$ mm actual approximation error distribution (c).

of the approximation NURBS curve. Figure 19a shows the distribution of the selected feature seed points in the equidistant point. When the approximation error limit is given as $\delta = 0.005$ mm, the number of control points after approximation is 261, as shown in Figure 19b, and the actual biggest approximation error is $\epsilon = 0.004856$ mm; when we set the approximation error limit as $\epsilon = 0.001$ mm, the number of control points after approximation is 495, as shown in Figure 19c, and the actual biggest approximation error is 0.001102 mm.

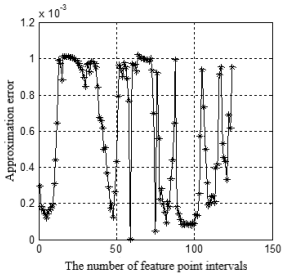
The second set of NURBS equidistant points contains sharp corners, so the pre-processing is required before

the curve approximation, and then the characteristic seed points in each set of the equidistant points corresponding to each trajectory segment are extracted. As shown in Figure 20, the number of the equidistant points is 2204, and the equidistant curve can be divided to four parts: the number of extracted characteristic seed points for the first part is 17, for the second part is 10, for the third part is 10, and for the fourth part is 17. Figure 20a shows the distribution of the selected feature seed points in the equidistant point. When the approximation error limit is given as 0.001 mm, there are 125 control points for the first part, and the actual biggest approximation error is 0.001021 mm, as shown in Figure 20b, there are 64 control points for the second part, and the actual biggest approximation error is 0.001166 mm, as shown in Figure 20c, there are 64 control points for the third part, and the actual biggest approximation error is 0.001224 mm, as shown in Figure 20d, there are 127 control points for the fourth part, and the actual biggest approximation error is 0.001069 mm, as shown in Figure 20d.

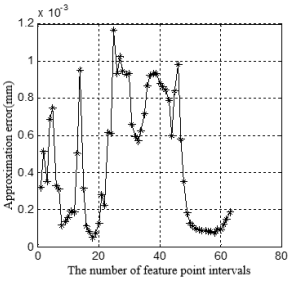
Comparing the errors of the two sets of equidistant point approximation, we can see that the approximation algorithm based on feature point extraction needs less control points while satisfying the machining precision compared with the previous approximation algorithm based on node configuration. And the fitting efficiency is also higher for the reason that the previous approximation algorithm has a certain degree of blindness when updating the node vector which will lead to low efficiency.



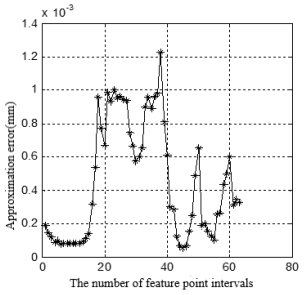
(a) Distribution of the feature seed points in the equidistant point



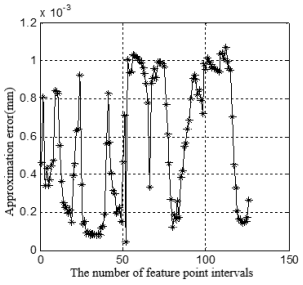
(b) Actual approximation error distribution for the first part



(c) Actual approximation error distribution for the second part



(d) Actual approximation error distribution for the third part



(e) Actual approximation error distribution for the fourth part

Fig. 20. Approximation error(mm) for NURBS 2206 equidistant points does not have sharp corners.

5 Simulation and experiment of NURBS tool radius compensation

In this paper, the simulation of the algorithm is based on the VERICUT simulation platform and actual machining experiment is based on three-axis CNC machine tools. The feasibility and effectiveness of the NURBS tool radius compensation algorithm introduced in this paper can be proved by the results of simulation and experiment. The VERICUT emulation platform can handle G code in NURBS form, so the input of simulation is G code in the form of NURBS corresponding to the tool center trajectory. Due to the limitations of the experimental conditions, we can not find the CNC machine tools supporting NURBS interpolator, so the tool center trajectory of NURBS is discrete according to the adaptive discrete algorithm in this paper, so that a series of linear segments are obtained for actual machining experiment after discretization.

5.1 VERICUT simulation process and result

The trajectory of the simulation consists of straight segments, arcs and NURBS multi-segment trajectories. The processing trajectory in the VERICUT simulation is shown in Figure 22.

We perform the left and right tool radius compensation, respectively as shown in Figure 22a and b.

From the simulation results in Figure 22, we can conclude that: no matter whether it is left-handed or right-handed, when dealing with a multi-segment trajectory, the appropriate transition method can be adopted for the elongation type transfer. For example, when the adjacent track deflection angle is relatively large, the arc transition method is adopted, and when the angle is relatively small, the tool center track extending method is adopted. For shortened type transitions, the algorithm can calculate the exact intersection position regardless of how many intersections of the NURBS tool center trajectory and the adjacent trajectory, thus avoiding the over-cut phenomenon at the transition.

Table 1. Parameters about experiment.

Max acceleration	200 mm/s^2
Turning time	0.02 s
Tool diameter	1 mm
Tool compensation radius	2 mm

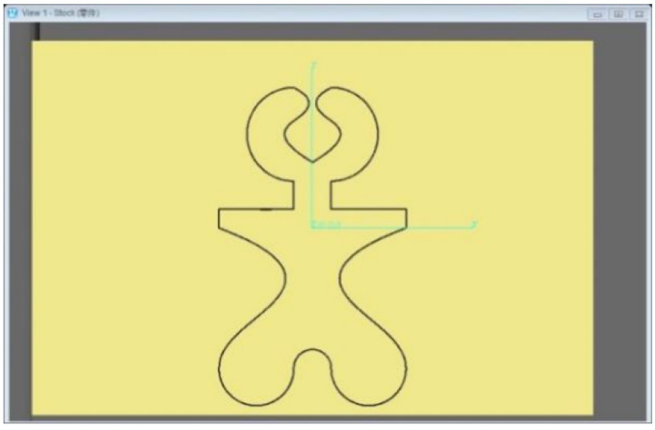
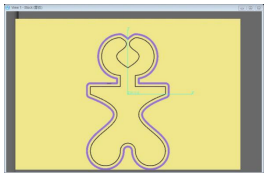
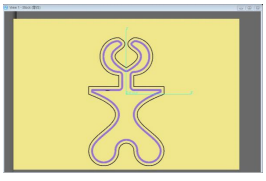


Fig. 21. Trajectory of the simulation consists of multi-segment trajectories.



(a) Left tool radius compensation for multi-segment trajectories



(b) Right tool radius compensation for multi-segment trajectories

Fig. 22. Tool radius compensation for multi-segment trajectories.

The tool radius compensation algorithm proposed in this paper has been proved to be effective and accurate in dealing with multi-segment trajectories by simulation . In order to verify the validity of the algorithm when dealing with self-intersection phenomenon for single-segment NURBS trajectories, a 3-order butterfly NURBS trajectory is simulated, and the trajectory of the track in VERICUT is shown in Figure 23. Two groups of comparative experiments were performed: the first sets the tool diameter as 1 mm and the tool radius as 1 mm, and the second sets the tool diameter as 4 mm and the tool radius as 2.5 mm. We perform the left and right two-way tool radius compensation for both sets of experiments, by which we can observe the band changes between the left and right tool center track to analyze the overcut situation. Figure 24a shows the two tool center tracks after machining in accordance with the data provided in the first set of experiment, and Figure 24b shows the result of the second set of experiment.

5.2 Actual machining processing examples and results

For the butterfly type NURBS, the rough length, width, and height of the machined parts used in this experiment are respectively 120 mm, 80 mm and 20 mm. Like the previous simulation, we also perform two sets of experiments. We firstly calculate the tool center trajectory for both right and left tool radius compensation according to the tool radius compensation algorithm proposed in this paper, and then load the trajectories into the machine

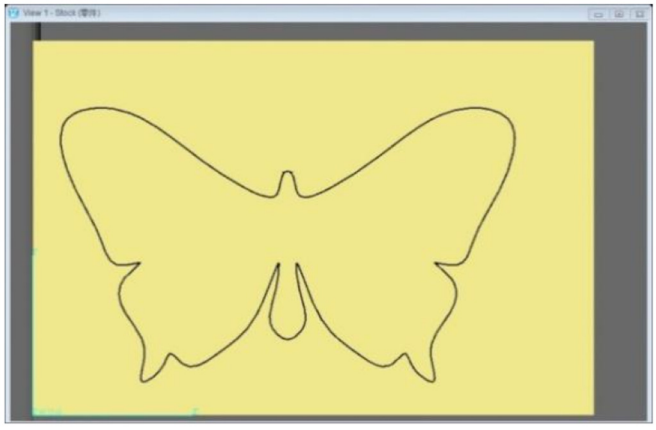
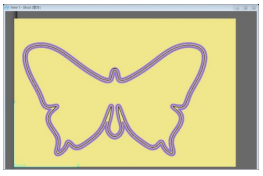
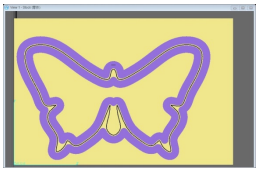


Fig. 23. Tool center trajectory simulation for single segment trajectory.



(a) Simulation result of the first set of experiment



(b) Simulation result of the second set of experiment

Fig. 24. Simulation result of single segment trajectory.

(a) Experiment result of the first set of experiment

(b) Experiment result of the second set of experiment

Fig. 25. Experiment results for single segment trajectory.

numerical control system. The track after machining is shown in Figure 25.

It can be seen from Figures 24 and 25 that the butterfly-type blade will be formed between the tool center trajectories when the tool cuts the blanks according to the trajectories obtained by the left and right tool radius compensation. For the first set of experiment, the thickness of butterfly-type blade is relatively , reached 1 mm. For the second set of experiment, the thickness of butterfly-type blade is relatively in some locations, this is due to the protection mechanism of the tool radius compensation algorithm to prevent over-cut of the parts and it also illustrates the effectiveness of the tool radius compensation algorithm proposed in this paper.

6 Conclusion

The NURBS tool radius compensation algorithm introduced in this paper fills the gap of the tool radius compensation for CNC systems which can support NURBS interpolation. The algorithm can directly deal with the contour trajectory containing NURBS, and can output the tool center trajectory in a straight line interpolation G code format or NURBS interpolation G code format based on specific needs.

However, if the contour trajectory contains too many NURBS, the calculation speed of the algorithm will be relatively slow, and research is needed to improve the real-time performance of the algorithm.

References

[1] Q.G. Zhang, R.B. Greenway, Development and implementation of a nurbs curve motion interpolator, Robot. Comput. Integr. Manufactur. **14**, 27–36 (1998)

[2] J. Li, Y. Liu, Y. Li, G. Zhong, S-model speed planning of nurbs curve based on uniaxial performance limitation, *IEEE Access* **7**, 60837–60849 (2019)

[3] M.-Y. Cheng, M.-C. Tsai, J.-C. Kuo, Real-time nurbs command generators for cnc servo controllers, Int. J. Mach. Tools Manufact. **42**, 801–813 (2002)

[4] L. Zhang, H. Wang, Y. Li, J. Tan, A progressive iterative approximation method in offset approximation, J. Comput. Aided Des. Comput. Graph. **26**, 1646–1653 (2014)

[5] W. Tiller, E. Hanson, Offsets of two-dimensional profiles, IEEE Comput. Graph. Appl. **4**, 36–46 (2007)

[6] S. Coquillart, Computing offsets of b-spline curves, Comput. Aided Des. **19**, 305–309 (1987)

[7] J. Hoschek, Spline approximation of offset curves. Springer, Netherlands (1990)

[8] R. Klass, An offset spline approximation for plane cubic splines, Comput. Aided Des. **15**, 297–299 (1983)

[9] B. Pham, Offset approximation of uniform b-splines, Compu. Aided Des. **20**, 471–474 (1988)

[10] I.K. Lee, M.S. Kim, G. Elber, Planar curve offset based on circle approximation, Comput. Aided Des. **28**, 617–630 (1996)

[11] L.G. Liu, G.J. Wang, Optimal approximation to curve offset based on shifting control points, J. Softw. **13**, 398–403 (2002)

[12] J. Tiffin, The applications of weighted dual functions of bernstein basis in curve offsetting, J. Compu. Aided Des. Comput. Graph. **23**, 1987–1993 (2011)

[13] Z. Jiang, X. Feng, X. Feng, Y. Liu, Contour-parallel tool-path planning of free surface using voronoi diagram approach, in *International Conference on Advanced Computer Theory and Engineering* (2010), pp. V1–302–V1–305.

[14] Z. Bo, J. Zhao, L. Lun, R. Xia, Nurbs curve interpolation algorithm based on tool radius compensation method, Int. J. Product. Res. **54**, 1–27 (2016)

[15] A. Kout, H. Müller, Tool-adaptive offset paths on triangular mesh workpiece surfaces, Comput. Aided Des. **50**, 61–73 (2014)

[16] T.C. Cai, Y.G. Zhao, Z.J. Wang, X.Y. Liu, The generation algorithm of planar nurbs curve and its offset, Mach. Des. Manuf. **7**, 224–227 (2014)

[17] T.W. Sederberg, S.R. Parry, Comparison of three curve intersection algorithms, Comput. Aided Des. **18**, 58–63 (1986)

[18] R.E. Barnhill, G. Farin, M. Jordan, B.R. Piper, Surface/surface intersection, Comput. Aided Geometr. Des. **4**, 3–16 (1987)

[19] L.I. Xu-Yu, W.U. Yu-Xiang, Research and implementation of a novel algorithm for tool compensation, Modul. Mach. Tool Autom. Manufactur. Tech. **11**, 21–24 (2009)

[20] H. Sun, D. Fan, A novel algorithm for arc transition tool compensation, China Mech. Eng. (2007)

[21] H. Park, J.H. Lee, Error-bounded b-spline curve approximation based on dominant point selection, in *International Conference on Computer Graphics* (2005)

[22] X. Cheng, W. Liu, M. Zhang, Approximation of b-spline curve of feature points, J. Comput. Aided Des. Comput. Graph. **23**, 1714–1718 (2011)

Cite this article as: J. Li, Q. Wang, G. Zhong, Planar tool radius compensation for CNC systems based on NURBS interpolation, Mechanics & Industry **21**, 107 (2020)