Int J Adv Manuf Technol (2007) 35:169–178
DOI 10.1007/s00170-006-0692-2

ORIGINAL ARTICLE

Design of a real-time adaptive interpolator with parameter compensation

Cunliang Yan · Daoshan Du · Congxin Li

Received: 17 November 2005 / Accepted: 9 June 2006 / Published online: 30 August 2006

Abstract The interpolator is the key part of a CNC system, which has strong effect on machining accuracy, tool motion smoothness, and machining efficiency. In this paper, a real-time adaptive interpolator is developed for non-uniform rational B-spline curves (NURBS) interpolation while considering the maximum acceleration/deceleration of the machine tool. In this proposed interpolator, both constant feedrate and high accuracy are achieved while the inconsistency of feedrate is dramatically reduced as well. In order to deal with the acceleration/deceleration around the feedrate-sensitive sharp corners, a look-ahead function is introduced to detect and adjust the feedrate adaptively. Furthermore, a parameter compensation scheme is proposed to eliminate the parametric truncation error which has been analyzed by several researchers but still not incorporated into any real-time interpolator so far. A case study was conducted to evaluate the feasibility of the developed interpolator.

Keywords NURBS · Adaptive interpolator · Look-ahead · Parameter compensation

1 Introduction

Parametric surfaces are extensively applied to a wide range of industries such as automotive, aerospace, and dies/molds. Before machining, parametric surfaces are usually discretized into a set of parametric curves through a process called tool-path planning, and then a trajectory planning process further processes these curves to prepare NC codes for the following CNC system.

In the conventional way, the CNC system only provides linear and circular interpolators which usually use a sequence of linear segments to approximate a curve. Therefore, a parametric curve after trajectory planning should be divided into a series of linear segments and sent to the CNC system. Such a method easily leads to a larger number of fluctuations in the feedrate due to the segmentation and also produces a large data file [1–4], which is obviously inefficient for general parametric surface machining.

Research has been done on developing an interpolator to deal with the parametric curves. Bedi et al. [5] set $\Delta(u_i)$ as a constant in the uniform interpolation algorithm to reduce speed fluctuation during the interpolation process. Houng and Yang [6] and Shpitalni et al. [7] developed first-order approximation interpolation algorithms using Euler and Taylor's expansions, respectively. These first-order approximation interpolation algorithms provide a uniform curve speed during the interpolation process. Then, the second-order approximation and speed-controlled interpolation algorithms were proposed by Yang and Kong [8], and Yeh and Hsu [9] yield more precise results. However, these algorithms do not consider chord error explicitly during the interpolation process. Furthermore, Yeh and Hsu developed an adaptive parametric curve interpolation algorithm [10] that eliminates the drawbacks of both the constant parameter interval and constant speed interpolation It also considers chord error. However, the acceleration/deceleration capacity of the machine is not mentioned. Although T. Yong and R. Narayanaswami adopted a method [11] to deal with the acceleration/deceleration at feedrate-sensitive corners, this method is not a real-time approach; it conducts interpolation calculation in an off-line mode.

C. Yan (✉) · D. Du · C. Li
National Die and Mould CAD Eng. Research Center,
Shanghai Jiao Tong University,
200030 Shanghai, People's Republic of China
e-mail: yancunliang@hotmail.com

Among many parametric curve representation methods, NURBS is one of the most popular industry standard tools and has been widely used in the interpolator design. The major advantages of NURBS can be summarized as follows: (1) They offer one common mathematical form for both standard analytical shapes (e.g., conics) and free-form shapes; (2) They provide the flexibility to design a large variety of shapes; (3) They can be evaluated reasonably fast by numerically stable and accurate algorithms; (4) They are invariant under affine as well as perspective transformations; (5) They are generalizations of non-rational B-splines, and non-rational and rational Bezier curves and surfaces.

Based on the present work by many researchers [5–16], in this paper a real-time adaptive interpolator with parametric compensation and real-time look-ahead function is developed for NURBS curve. The proposed algorithm keeps the feedrate constant during most of the interpolation process while the chord error is still kept within a specified tolerance range. By using the parametric compensation, the feedrate curve in the interpolation process can be improved dramatically; and with the help of real-time look-ahead, the acceleration/deceleration can be limited to the range of the machine tool capacity, thus, feedrate fluctuation is reduced.

The outline of this paper is organized as follows. In Section 2, the concept of parameter curve interpolation is introduced. In Section 3, the development of proposed real-time adaptive interpolator is discussed in detail. Section 4 discusses the parameter compensation algorithm. Section 5 shows a case study. The paper is concluded in Section 5.

2 Parameter curve interpolation

The parameter curve equation can be described as follows: (Here shows a two-dimension curve,)

$$C(u): \begin{cases} x = X(u) \\ y = Y(u) \end{cases} \tag{1}$$

Where, u is parameter variable.

When machining starts, the cutting tool moves at a certain speed F. Assuming the interpolation period is T, the step length of this period will be $\Delta L=FT$. From the aspect of machining process, u is a function of the time, t, i.e., $u=u(t)$. When $t=t_i$, u is u_i, and the position of machine tool is $C(u_i)$. In the next interpolation period, $t=t_{i+1}$,$u=u_{i+1}$, the position of machine tool, $C(u_{i+1})$ can be calculated from the step length, ΔL. According to the above explanation, the interpolation process is that if the parameter variable u is divided into $\{u_1, u_2\ldots\ldots, u_1\ldots\ldots u_n\}$ by a sequence of time $\{t_1, t_2\ldots\ldots, t_i\ldots\ldots t_n\}$, the sequence of the position of the machine tool $\{C(u_1), C(u_2)\ldots\ldots, C(u_i)\ldots\ldots C(u_n)\}$ can be determined. So, the key issue of parameter curve interpolation is to calculate u_{i+1} from u_i and therefore the corresponding interpolation points $C(u_{i+1}):\{X(u_{i+1}), Y(u_{i+1})\}$.

By using Taylor's expansion, the approximation up to the second derivative is:

$$u_{i+1} = u_i + \left.\frac{du}{dt}\right|_{t=ti} T + \frac{1}{2}\left.\frac{d^2u}{dt^2}\right|_{t=ti} T^2 + \varepsilon \tag{2}$$

Where, T is interpolation period and $T = t_{i+1} - t_i \cdot \varepsilon$ is truncation error, which is determined in requirement.

Equation (2) can be changed as [12]:

$$u_{i+1} = u_i + \frac{F(u_i)T}{\sqrt{\left(\frac{dX(u_i)}{(du)}\right)^2 + \left(\frac{dY(u_i)}{du}\right)^2}} + \varepsilon \tag{3}$$

During interpolation calculating, the conventional method uses one-degree or two-degree Taylor's expansion according to the accuracy requirement, and the parametric truncation error, ε, is completely ignored. The result without parameter compensation is analyzed in Fig. 1. Although the next step interpolation is still on the curve, its position has been changed from $C(u_{i+1})$ in theory to $C(u'_{i+1})$ in fact. The change of position results in the difference between the actual feedrate and the desired feedrate; chord error is also changed to δ' from δ. Thus, when machining at sharp corners, the chord error is probably over the allowable tolerance range.

Obviously, it is impossible to calculate the parametric truncation error from the Taylor's expansion. This value can only be gained by other means; in [9], Yeh and Hsu analyzed the compensatory value, ε, from the principle of interpolation. Section 4 of this paper gives a method of how to determine this value approximately.

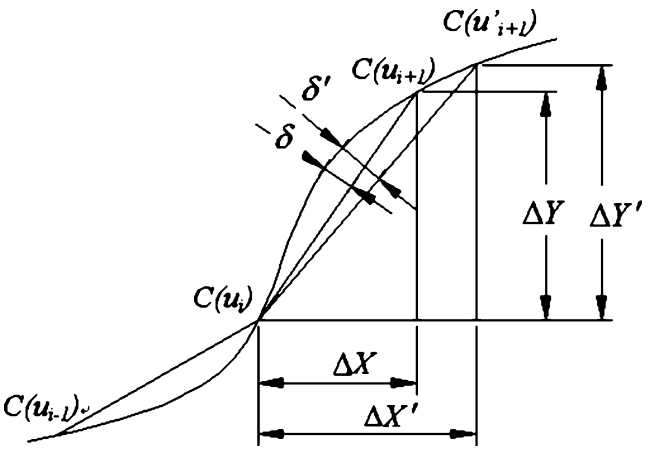

Fig. 1 Chord error and interpolation

3 Adaptive interpolator design

3.1 Interpolator architecture

The architecture of the proposed interpolator is shown in Fig. 2, which mainly includes three function modules: the parameter compensation module, the adaptive feedrate adjustment module, and real-time look-ahead module.

The parameter compensation module is used to calculate u_{i+1} from u_1 and F with parametric truncation error compensation, which can gain more accurate parameter variable than that of the first-order approximation interpolation or the second-order approximation interpolation. Inputs of the truncation error compensation module are curve property (control point, knot vector, weight, command federate F_c) and the parameter variable u_i. Output is the next step parameter variable u_{i+1}. The purpose of the adaptive feedrate adjustment module is to adjust the feedrate to reduce chord error into the set tolerance. Its inputs include the parameter variable u_i and the maximal allowable chord error while the output is adjusted feedrate. The real-time look-ahead module is to detect the feedrate-sensitive position by comparing the feedrate at point u_i to the feedrate point u_{i-1}, and then to deal with the feedrate to confine acceleration/deceleration within the capacity of the machine tool.

As seen in Fig. 2, the inputs to the proposed NURBS interpolator include both geometric information and cutting conditions. The geometric information involves knot vectors, weights, and control points while the cutting conditions mostly mean the feedrate. The feedrate is determined in three steps. Firstly, the feedrate is given as command with the desired chord error setting. Secondly, this feedrate is adjusted adaptively when the actual chord error is larger than the set tolerance. Finally, the feedrate is processed by the look-ahead function according to the acceleration/deceleration capacity of the machine tool. After the feedrate processing, the final outputs will be the interpolation point, which will be sent to the following servo controller.

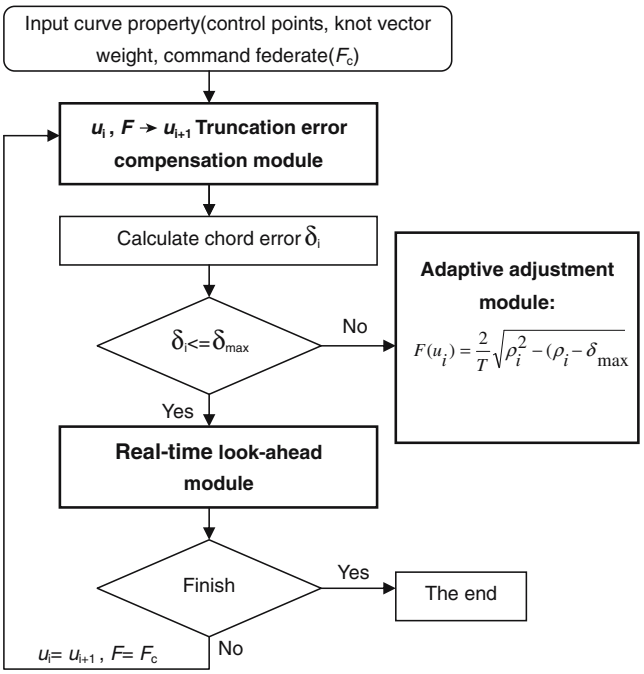

Fig. 2 The flow of the interpolator

3.2 Truncation error compensation module and chord error calculation

In terms of the discussion in Section 2 of this paper, ignoring the parametric truncation error during the parametric curve interpolating may change the actual feedrate, which may further increase the chord error to over the desired tolerance. In order to overcome this problem, a practical method is applied here to approximate the parametric truncation error and consider it into the parametric curve interpolation.

Referring to Eq. (3), assume:

$$u'_{i+1} = u_i + \frac{F(u_i)T}{\sqrt{\left(\frac{dX(u_i)}{du}\right)^2 + \left(\frac{dY(u_i)}{du}\right)^2}}$$

Equation (3) can be described as:

$$u_{i+1} = u'_{i+1} + \varepsilon \tag{4}$$

Because

$$\begin{aligned} F(u_i) &= \sqrt{F_x^2(u_i) + F_y^2(u_i)} \\ &= \frac{1}{T}\sqrt{[X(u_{i+1}) - X(u_i)]^2 + [Y(u_{i+1}) - Y(u_i)]^2} \end{aligned} \tag{5}$$

$X(u_{i+1})$, $Y(u_{i+1})$ can be calculated using Taylor's expansion at the point u'_{i+1} as:

$$\begin{aligned} X(u_{i+1}) &= X\left(u'_{i+1} + \varepsilon\right) = X\left(u'_{i+1}\right) + \frac{dX\left(u'_{i+1}\right)}{du}\varepsilon \\ Y(u_{i+1}) &= Y\left(u'_{i+1} + \varepsilon\right) = Y\left(u'_{i+1}\right) + \frac{dY\left(u'_{i+1}\right)}{du}\varepsilon \end{aligned} \tag{6}$$

Use Eq. (6) to substitute the corresponding variable in Eq. (5), and multiply T on both sides of Eq. (5), at the same time. Assuming

$$\Delta X'(u_i) = X\left(u'_{i+1}\right) - X(u_i),\ \Delta Y'(u_i) = Y\left(u'_{i+1}\right) - Y(u_i),$$

then, Eq. (5) can be simply described as follows:

$$F(u_i)T = \sqrt{\left[\Delta X'(u_i) + \frac{dX(u'_{i+1})}{du}\varepsilon\right]^2 + \left[\Delta Y'(u_i) + \frac{dY(u'_{i+1})}{du}\varepsilon\right]^2}$$

Thus, the equation about ε is:

$$\underbrace{\left[\left(\frac{dX(u'_{i+1})}{du}\right)^2 + \left(\frac{dY(u'_{i+1})}{du}\right)^2\right]}_{A}\varepsilon^2 + 2\underbrace{\left[\Delta X'(u_i)\frac{dX(u'_{i+1})}{du} + \Delta Y'(u_i)\frac{dY(u'_{i+1})}{du}\right]}_{B}\varepsilon + \underbrace{\Delta X'^2(u_i) + \Delta Y'^2(u_i) - F^2(u_i)T^2}_{C} = 0 \quad (7)$$

Assume:

$$A = \left(\frac{dX(u'_{i+1})}{du}\right)^2 + \left(\frac{dY(u'_{i+1})}{du}\right)^2$$

$$B = 2\left[\Delta X'(u_i)\frac{dX(u'_{i+1})}{du} + \Delta Y'(u_i)\frac{dY(u'_{i+1})}{du}\right.$$

$$C = \Delta X'^2(u_i) + \Delta Y'^2(u_i) - F^2(u_i)T^2$$

Equation (7) is simplified:

$$A\varepsilon^2 + B\varepsilon + C = 0 \quad (8)$$

By solving this equation, the parametric truncation error can be gained. If $B^2 - 4AC \le 0$, the equation has complex roots or has only one root. This is the case that $F(u_i)$ is too big and chord error probably exceeds the maximal allowable tolerance. The command feedrate should be reduced. If $B^2 - 4AC > 0$, the equation has two real roots:

$$\varepsilon_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A},\ \varepsilon_2 = \frac{-B - \sqrt{B^2 - 4AC}}{2A} \quad (9)$$

As ε_1 is small, the first root is near zero and the other root is negative and relatively large. To achieve reliable compensation and forward motion during the interpolation process, the small compensatory parameter is preferable [9].

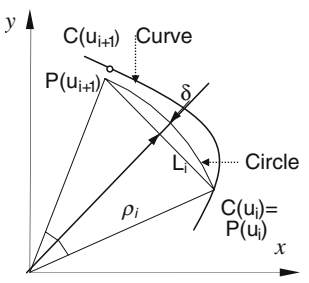

Fig. 3 Chord error

Chord error calculation is very important for the interpolator design. Figure 3 shows a graphical representation of chord error. To determine the relationship between the chord error and the feedrate, the arc approximation is thus applied. Suppose the object arc of $u\in[u_i, u_{i+1})$ has radius ρ_i at parameter $u=u_i$, where $\rho_i = 1/K_i$, is the radius of curvature at $u=u_i$. The curvature K_i will be:

$$K_i = \frac{\frac{dC_X(u)}{du}\frac{d^2C_Y(u)}{du^2} - \frac{dC_Y(u)}{du}\frac{d^2C_X(u)}{du^2}\Big|_{u=u_i}}{\left|\frac{dC(u)}{du}\right|^3\Big|_{u=u_i}} \quad (10)$$

Where: $P(u_i)$ is the interpolated point on the arc at $u=u_i$, $P(u_{i+1})$ is the estimated interpolated point at $u=u_{i+1}$ $C(u_i)$ and $C(u_{i+1})$ are the interpolated points at $u=u_i$ and $u=u_{i+1}$ respectively. For $C(u_i)=P(u_i)$, the chord error can be derived [10]:

$$\delta_i = \rho_i - \sqrt{\rho_i^2 - \left(\frac{L_i}{2}\right)^2} \quad (11)$$

3.3 Adaptive adjustment module

Usually the feedrate is a quite important factor that influences the chord error. The higher the feedrate is, the larger the chord error will be. While at the same time, it is desirable to keep the feedrate constant as much as possible in order to achieve high surface machining quality, though this may lead to the undesired chord error. In order to overcome this drawback, an adaptive method was proposed where the feedrate is kept constant at most of the time, and it will be adaptively adjusted depending on the curvature radius of the interpolation points when the chord error exceeds the tolerance. From Eq. (11), assuming the allowed chord error is $\delta_{\max}$, the feedrate could change adaptively along the curve subjecting to the following law [10]:

$$F(u_i) = F_c, \qquad If\ \rho_i - \sqrt{\rho_i^2 - \left(\frac{L_i}{2}\right)^2} \le \delta_{max},$$

$$F(u_i) = \frac{2}{T}\sqrt{\rho_i^2 - (\rho_i - \delta_{max})^2},\ If\ \rho_i - \sqrt{\rho_i^2 - \left(\frac{L_i}{2}\right)^2} > \delta_{max}, \quad (12)$$

Where, F_c is a command feedrate.

The process of adaptive feedrate adjustment is explained as follows:

1) Let the feedrate variable F be equal to command feedrate F_c;
2) Calculate u_{i+1} from F and u_i (consider the parameter compensation);

3) Calculate chord error $\delta(u_i)$ at the point u_i, and compare $\delta(u_i)$ and $\delta_{\max}$ If $\delta(u_i)<\delta_{\max}$, jump to step 5;
4) In step 3, if $\delta(u_i)>\delta_{\max}$, the feedrate at the point u_i is too large and needs adaptive adjustment according to Eq. (12). Then, let the feedrate variable F be equal to the calculated feedrate $F(u_i)$, jump to step 2;
5) Exit adaptive feedrate adjustment function and enter look-ahead function with the feedrate F at point u_i and the next step parameter variable u_{i+1}.

3.4 Real-time look-ahead module

The adaptive feedrate adjustment can assure that the chord error is within an allowable range and keep the feedrate along the curve constant most of the time. But, if the first derivative of the curve is not continuous at some points, the abrupt change of feedrate will appear and probably exceed the acceleration/deceleration capacity of the machine tool. Here, a real-time look-ahead function is introduced to deal with the abrupt change of the feedrate. The purpose of look-ahead has two aspects: one is to detect the feedrate-sensitive position where the feedrate may change abruptly; the other is to determine the start point of acceleration/decelerate and further smooth the feedrate within the sensitive zone.

The detection algorithm for the feedrate-sensitive position is to compare the difference of the feedrate between the current interpolation point $C(u_{i+1})$ and the last interpolation point $C(u_i)$, to figure out whether or not the change exceeds the maximal acceleration/deceleration capacity ($A_{\max}$). Usually the acceleration/deceleration adjustment is necessary when encountering a little radius of curvature along the curve. Two situations may appear: one of increasing feedrate or one of decreasing feedrate.

Assume that the command feedrate is F_c and the maximal acceleration/deceleration capacity is $A_{\max}$. First considering the deceleration situation, in the worst case, the feedrate may be reduced to zero from the current feedrate. With the maximal deceleration capacity, the minimal distance needed to finish the deceleration will be $S_{min} = F_c^2/2A_{max}$.

A structure array is designed to store a sequence of the former interpolation points. The form of this structure array is shown as *point[i]{u,F, distance_step,distance_entire}*. Where, "*u*" is variable parameter of the *i*-th interpolation point in the structure array; "*F*" is its feedrate; "*distance_step*"is the distance between this point and the next interpolation point, also *distance_step=F×T*, *T* is the interpolation sample time; "*distance_entire*" is the distance between the current point and the first interpolation point in the structure array *point[0]{u,F, distance_step, 0 }*. In terms of the above definition, the following equations are satisfied:

$$
\begin{aligned}
&point[i].distance\_entire \\
&= point[i-1].distance\_entire + point[i-1].distance\_step \\
&= point[0].distance\_step + point[1].distance\_step \ + \ldots\ldots + \ point[i-1].distance\_step
\end{aligned}
$$

The structure array does not need to store all of the interpolation points according to two limitations. One is that the distance from the first point to the last point is not smaller than $S_{\min}$, i.e., *point[i].distance_entire*$\geq S_{\min}$; the other is that the distance from the second point to the last point is smaller than $S_{\min}$, i.e., (*point[i].distance_entire-point[0].distance_step*)$<S_{\min}$.

Such a structure array is very useful: when a feedrate-sensitive point is located, the distance needed to finish the deceleration can be calculated according to the maximal deceleration capacity and the maximal allowable feedrate from Eq. (12). In the structure array, the point from where the deceleration starts can be found by using a bisection method. The searching condition is that the distance between this point and the feedrate-sensitive point is not smaller than the distance calculated above. Then, re-interpolation can be done by tracing back to this point from the current interpolation point, i.e., the feedrate-sensitive point. During re-interpolating, the actual deceleration is calculated based on the following information: the feedrate at this point, the feedrate at the feedrate-sensitive point and the distance between the two points. Thus the feedrate is adjusted in terms of the actual deceleration before arriving the feedrate-sensitive point.

In the deceleration zone, after the look-ahead function, the feedrate is smaller than the result of adaptive adjustment, thus, the chord error can be assured. But in the case of acceleration, the feedrate is possibly larger than the result of adaptive adjustment with the same method as the deceleration treatment. When encountering a feedrate-

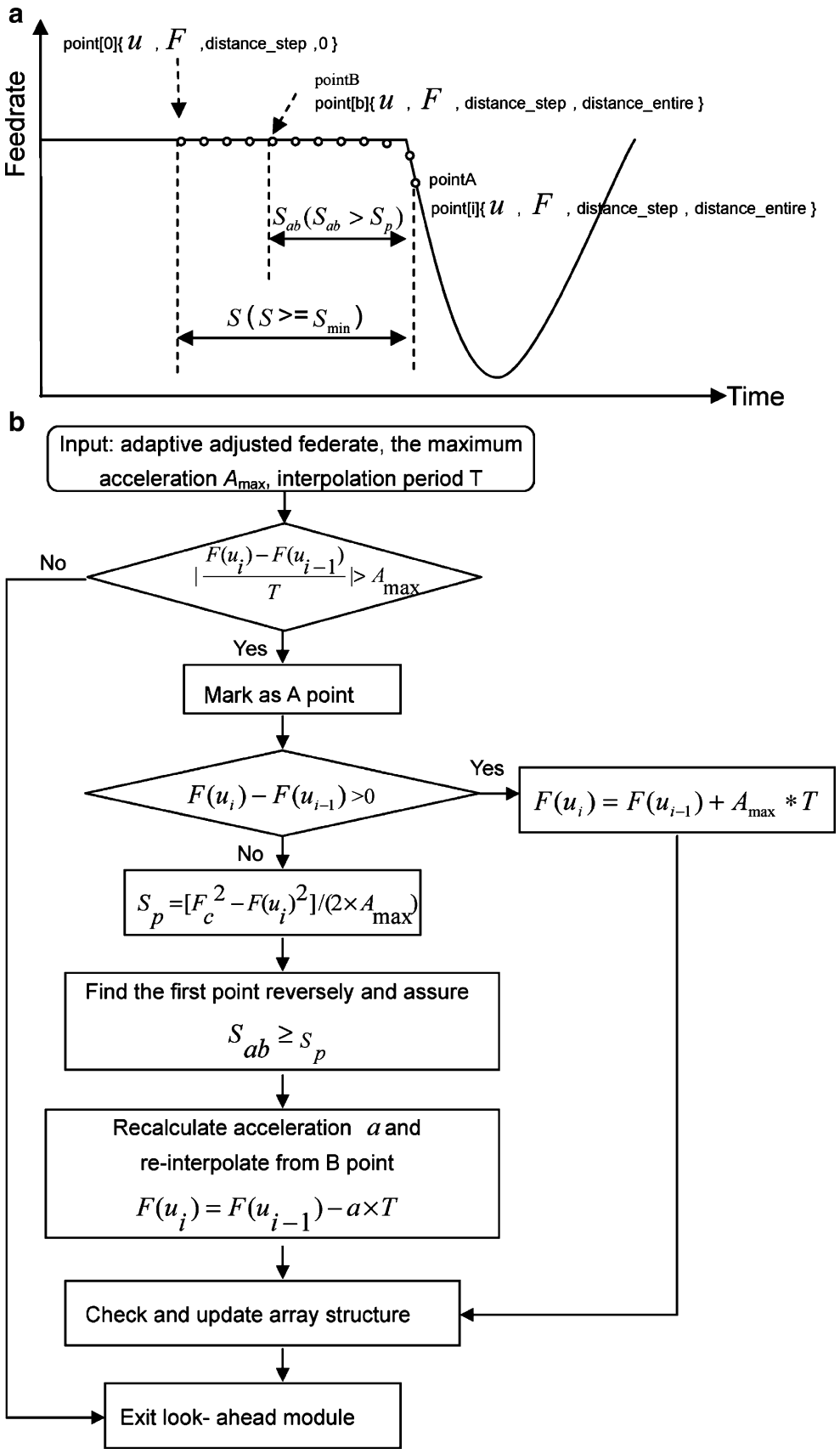

Fig. 4 **a** Feedrate profile of a feedrate-sensitive corner; **b** Flow of look-ahead module

sensitive point in the acceleration zone, in order to keep the chord error within the tolerance range, the feedrate calculation is subject to $F(u_i) = F(u_{i-1}) + A_{max} \times T$.

In summary of the above analysis, the complete acceleration/deceleration adjustment process of look-ahead function at the feedrate-sensitive position as follows (shown in Fig. 4a and b):

1) Detect the feedrate-sensitive point. If $\left|\frac{F(u_i)-F(u_{i-1})}{T}\right| > A_{max}$, the point u_i on curve is a feedrate-sensitive point and look-ahead function is therefore necessary. Mark this point as A. Else, jump to step 10 and exit look-ahead function;
2) Determine deceleration or acceleration adjustment for point A. If $F(u_i) - F(u_{i-1}) > 0$, it is an acceleration-sensitive point. The feedrate is calculated by $F(u_i) = F(u_{i-1}) + A_{max} * T$. Then, jump to step 8. If $F(u_i) - F(u_{i-1}) < 0$, it is a deceleration-sensitive point, go to step 3;
3) Calculate the distance needed to decrease the feedrate F_c to $F(u_i)$ with the deceleration $A_{\max}$, mark this distance as S_p. Obviously, $S_p \leq S_{\min}$;
4) Find a point B by using the bisection method in the structure array to assure that the distance S_{ab} between point A and point B is not smaller than S_p, i.e., $S_{ab} \geq S_p$;
5) Go back to point B and re-interpolate. The actual deceleration, a, is calculated based on the parameters: the feedrate at this point stored in the structure array, the feedrate at the feedrate-sensitive point, $F(u_i)$, and the distance between two points, S_{ab};
6) Re-interpolate from point B. The feedrate is adjusted depending on the actual deceleration, a, $F(u_i) = F(u_{i-1}) - a * T$;
7) Re-interpolate until achieving point A;
8) After each interpolation, update the structure array. A new component, point[i]{u, F, distance_step, distance_entire }, is added into the structure array;
9) Check the structure array, and assure: (1) the distance from the first point to the last point is not smaller than $S_{\min}$, i.e., point[i].distance_entire$\geq S_{\min}$; (2) the distance from the second point to the last point is smaller than $S_{\min}$, i.e., (point[i].distance_entire - point[0].distance_step)$< S_{\min}$;
10) Exit look-ahead function.

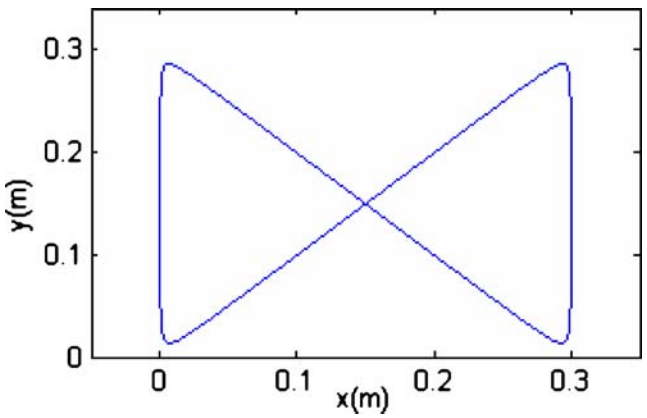

Fig. 5 An example of NURBS curve

4 Case study

4.1 NURBS curve example

To evaluate the designed interpolator performance, a NURBS curve with two degrees is chosen, which is shown in Fig. 5. Interpolator prototype is programmed by using VC++.

The control points, weight vector, and knot vector of this NURBS curve are:

- The ordinal control points are: {0.15,0.15,0}, {0,0,0}, {0,0.3,0}, {0.15,0.15,0}, {0.3,0,0}, {0.3,0.3,0}, {0.15,0.15,0}(m);
- The weight vector is: W={1,100,100,1,100,100,1};
- The knot vector is: U= {0,0,0,0.25,0.5,0.5,0.75,1,1,1};
- The sampling time in interpolation is: T=2 ms;
- The feedrate command is: F=0.2 m/s;
- The maximal allowable chord error is : Emax=1e-7 m;
- The maximal allowable acceleration/deceleration is: Amax=0.5 m/s^2.

4.2 Interpolation result analysis

Upon the above NURBS curve, there are 4,037 interpolation steps with only adaptive feedrate adjustment and the consumed time is 8.074 s. The minimal feedrate occurs at the position (time, sec): 1.174, 2.882, 5.210 and 6.918, respectively. With the real-time look-ahead and parameter compensation, the number of interpolation steps is 4,242 and the time is 8.484 s. The minimal feedrate occurs at the position (time, sec): 1.276, 3.188, 5.72 and 7.632, respectively. Two feedrate profiles are shown in Fig. 6. As seen in the figure, it is not easy to analyze the influence of look-ahead function and parameter compensation. To ease study, the feedrate profile with look-ahead function and parameter compensation is divided into four segments, each part is shifted left: the profile in (I), (II), (III), (IV) is shifted left by 102 ms (1.276–1.174 s), 306 ms (3.188–2.882 s), 510 ms (5.72–5.210 s), 714 ms (7.632–6.918 s), respectively. The similar shift is applied to the acceleration/deceleration profile and the chord error profile. The results after shifting are shown in Figs. 7, 8, 9.

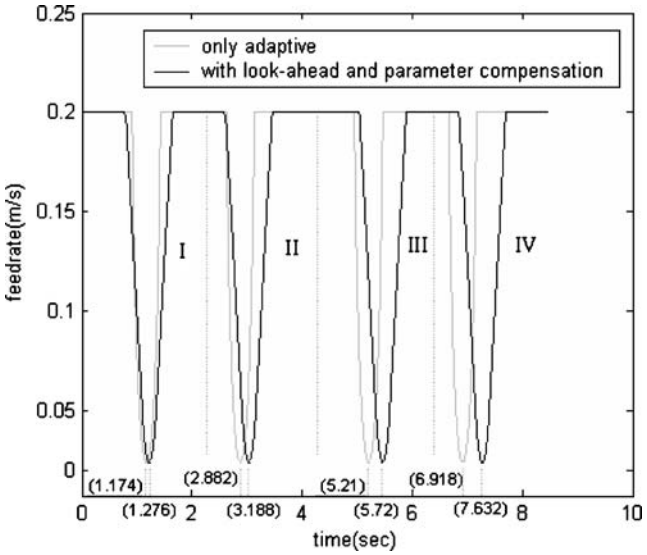

Fig. 6 Feedrate profile of adaptive interpolator (before shift)

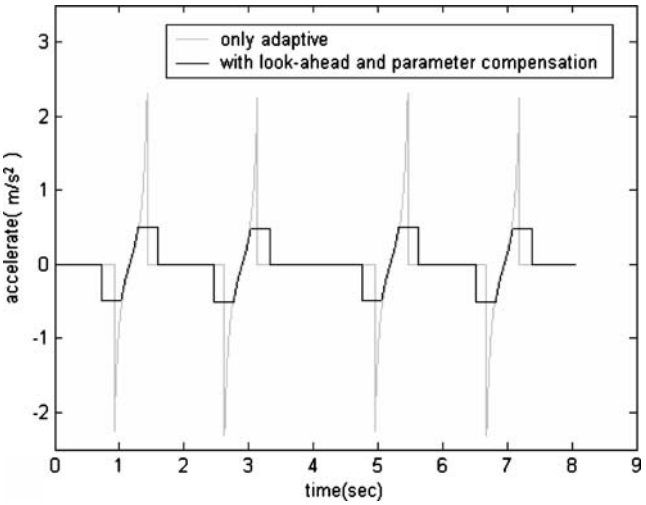

Fig. 8 Acceleration/deceleration profile of adaptive interpolator (after shift)

Compare the feedrate profile in Fig. 7 and the chord error profile in Fig. 9; it can be seen that there is a very close correlation between the chord error and feedrate. When the chord error reaches the maximum permissible value, the feedrate is adapted and slowed down so that the chord error never exceeds the maximum value (1e-7 m).

Also, there is a very close correlation between the feedrate and the acceleration/deceleration (Fig. 8). From Fig. 8, the maximal acceleration/deceleration can reach 2.2 m/s^2 at the feedrate-sensitive corner before look-ahead function. After the acceleration/deceleration adjustment in look-ahead function, the acceleration/deceleration zone becomes wider. The acceleration/deceleration profile is improved and confined within the desired range. At the same time, the feedrate within the area in which acceleration/deceleration exceed the limitation (0.5 m/s^2) is smoothed (Fig. 7).

In addition to the close correlation between the chord error and the feedrate, the difference of chord error before using look-ahead and that after is also indicated. The time keeps the maximal allowable chord error become shorter after look-ahead function and parameter compensation. Though actual chord error at some zone is bigger than that before acceleration/deceleration treatment, the value of chord error does not exceed the maximal allowable range.

4.3 Parameter compensation analysis

By applying the proposed algorithm, the roots of Eq. (11) for the compensatory value are also shown in Fig. 10. Figure 10 indicates that one root is near zero, which is adopted here while the other is negative and relatively large.

S. Yeh, P. Hsu [5] offered a speed-controlled algorithm with parameter compensation, which obviously obtained

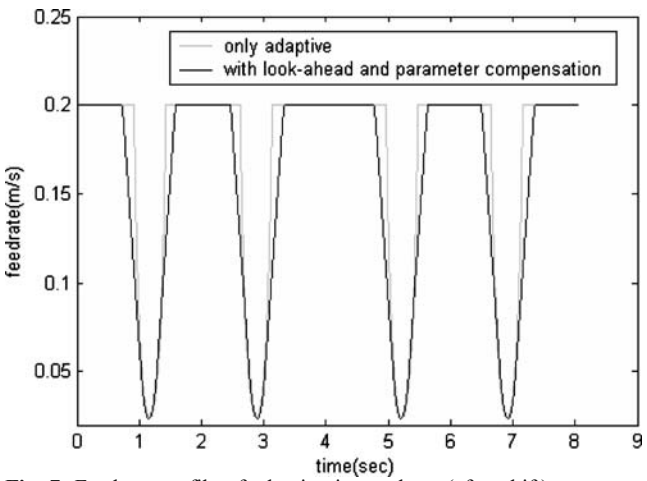

Fig. 7 Feedrate profile of adaptive interpolator (after shift)

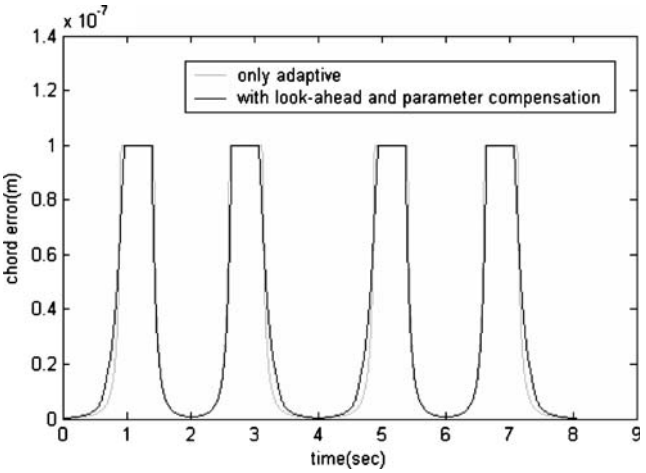

Fig. 9 Chord error profile of adaptive interpolator (after shift)

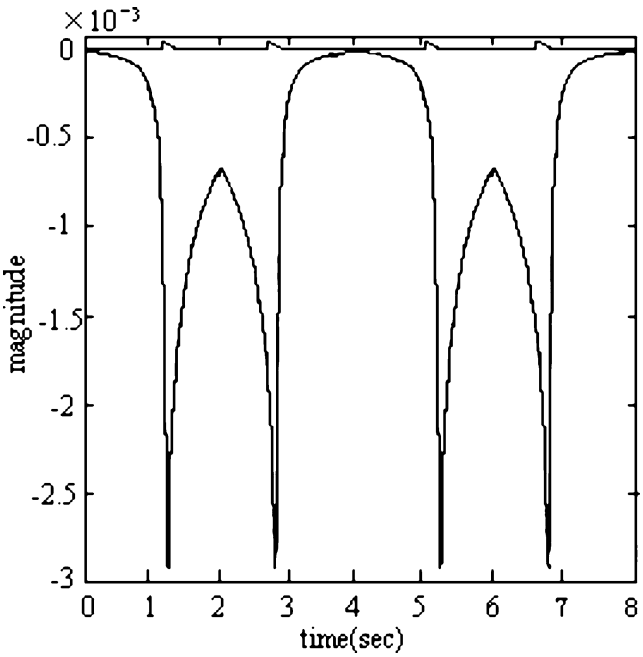

Fig. 10 Roots of Eq. (11)

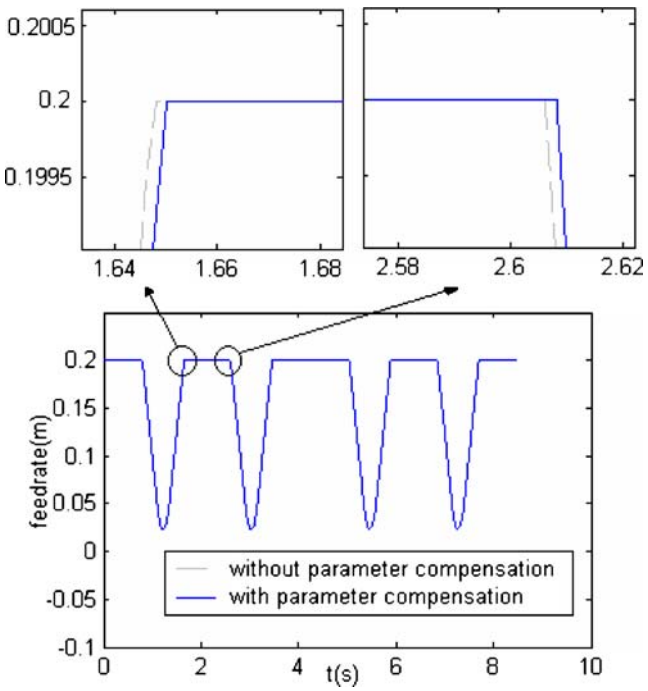

Fig. 12 Influence on feedrate of parameter compensation

smaller feedrate deviation than the second-order Taylor's approximation, shown in Fig. 11a and b. However, this algorithm did not consider the chord error and the acceleration/deceleration capacity. In this paper, both of these factors are considered (the maximal chord error is 1e-7 m, the maximal acceleration/deceleration is 0.5 m/s^2). In addition, the influence of parameter compensation on feedrate and curve contour error are analyzed. The results are shown in Figs. 12 and 13, respectively.

Since the adjusting magnitude of the feedrate in adaptive feedrate adjustment function and look-ahead function is very big while the parameter compensatory value is very small, Figs. 12 and 13 are zoomed in at specified zone in order to clearly observe the influence of parameter compensation on the feedrate and curve contour. As seen in Fig. 12, the feedrate profile is shifted after parameter compensation, which directly influences the curve error in Fig. 13. Obviously, the interpolation curve after parameter compensation is closer to the actual curve.

5 Conclusion

Parametric curve interpolation is preferred over linear interpolation because it improves machining performance. With parametric interpolation, the feedrate could be improved dramatically while the chord error can also be controlled within a desired range. In this paper, a real-time adaptive interpolator was developed for NURBS curve interpolation. By real-time detection of the feedrate-sensitive positions and applying the parametric truncation error compensation algorithm, the proposed interpolator acquires

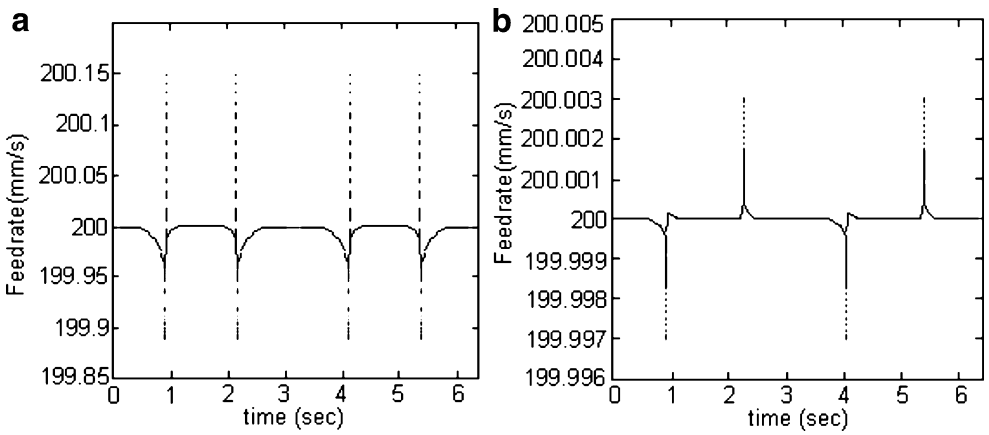

Fig. 11 **a** Simulation results of the second-order approximation [9]; **b** Simulation results of the controlled constant-speed interpolation [9]

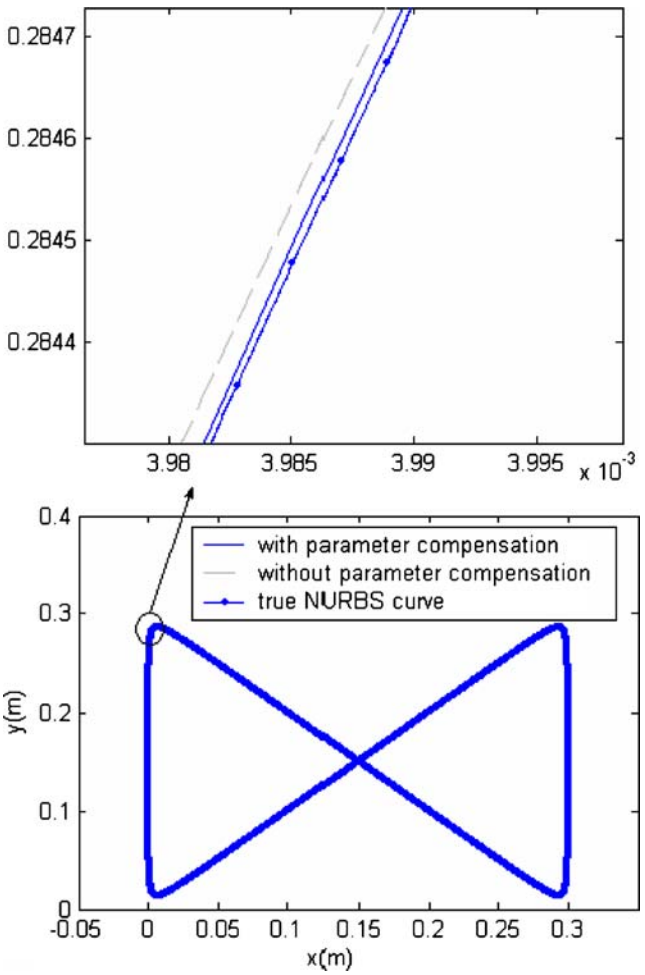

Fig. 13 Influence on curve contour of parameter compensation

the constant feedrate at the most time while the chord error is controlled within the desired value at each interpolation point. At the same time, the acceleration/deceleration capacity of machine tool is met.

In the developed real-time adaptive interpolator, the feedrate along the curve is determined through three steps. First, the feedrate is given as command. Second, the feedrate is adjusted adaptively. Third, the feedrate is calculated by a look-ahead function in terms of the acceleration/deceleration capacity of the machine tool. A case study shows that a quite good performance is achieved by using the proposed interpolator. In addition, a desired chord error is obtained while the feedrate profile is significantly smoothed as well.

References

1. Koren Y (1997) Control of machine tools. ASME Trans J Manuf Sci Eng 119:749–755
2. Koren Y, Lo CC, Shpitalni M (1993) CNC interpolators: algorithm and analysis. ASME Winter Annual Meeting PED 64:83–92
3. Hwang J (1992) Interference-free tool-path generation in the NC machining of parametric compound surfaces. CAD 24:667–677
4. Kawabe S, Kimura F, Sata T (1981) Generation of NC commands for sculptured surface machining from three-coordinate measuring data. CIRP Ann 30:369–372
5. Bedi S, Ali I, Quan N (1993) Advanced interpolation techniques for CNC machines. ASME J Eng Ind 115:329–336
6. Huang JT, Yang DCH (1992) A generalized interpolator for command generation of parametric curves in computer controlled machines. Japan/USA Symposium on Flexible Automation 1:393–399
7. Shpitalni M, Koren Y, Lo CC (1994) Real-time curve interpolators. CAD 26(11):832–838
8. Yang DCH, Kong T (1994) Parametric interpolator versus linear interpolator for precision surface machining. CAD 26(3):225–234
9. Yeh S, Hsu P (1999) The speed-controlled interpolator for machining parametric curves. CAD 31:349–357
10. Yeh S, Hsu P (2002) Adaptive-feedrate interpolation for parametric curves with a confined chord error. CAD 34:229–237
11. Yong T, Narayanaswami R (2003) A parametric interpolator with confined chord errors, acceleration and deceleration for NC machining. CAD 35:1249–1259
12. Zhang Qiyi G, Greenway RB (1998) Development and implementation of a NURBS curve motion interpolator. Robot Comput-Integr Manuf 14:27–36
13. Erkorkmaz K, Altintas Y (2001) High-speed CNC system design. Part I.Jerk-limited trajectory generation and quintic spline interpolation. Int J Adv Manuf Technol 41:1323–1345
14. Nam S, Yang M (2004) A study on a generalized parametric interpolator with real-time jerk-limited acceleration. CAD 36: 27–36
15. Tsai YF, Farouki RT, Feldman B (2001) Performance analysis of CNC interpolators for time-dependent feedrates along PH curves. Comput-Aid Geom Des 18:245–265
16. Zhiming X, Jincheng C, Zhengjin F (2002) Performance evaluation of a realtime interpolation algorithm for NURBS curves. Int J Adv Manuf Technol 20:270–276