Int J Adv Manuf Technol (2012) 63:1257–1270
DOI 10.1007/s00170-012-4167-3

ORIGINAL ARTICLE

An accurate NURBS curve interpolation algorithm with short spline interpolation capacity

Haitao Dong · Bing Chen · Youping Chen · Jingming Xie · Zude Zhou

Received: 6 December 2011 / Accepted: 10 April 2012 / Published online: 8 May 2012
© Springer-Verlag London Limit 2012

Abstract In this paper, a novel and accurate real-time non-uniform rational B-spline curve interpolation algorithm is proposed. This algorithm not only considers chord errors, feedrate fluctuations, jerk-limited, and acceleration/deceleration (Acc/Dec) capabilities of the machine, but also optimizes the look-ahead process. In the meanwhile, it improves machining efficiency by adding the circular buffer and pre-interpolation (non-off-line) and enhances the real-time performance by removing the time-consuming calculation from the interrupt service routine. Furthermore, the proposed interpolation algorithm can interpolate both the long spline and the short spline with uniform method. The advantages of the proposed method were confirmed by the simulation results.

Keywords CNC · NURBS · Interpolation algorithm · Look-ahead · S curve acceleration/deceleration

1 Introduction

The conventional computer numerical control (CNC) machines generally support only linear and circular interpolations, which lead to feedrate fluctuation, large jerk, and heavy transmission between CAM and CNC. To overcome the shortcomings of the linear/circular interpolation and achieve the high-speed and high-precision CNC machining, many scholars devote themselves to parametric interpolation, because parametric interpolation is able to achieve high-speed and high-precision performance. In the early 1990s, Shipitalni et al. proposed and realized a parametric curve interpolation in CNC machine, which received parametric curve codes directly from CAD/CAM [1]. Yang and Kong presented a comparative study of linear and parametric interpolators used for command generation during CNC machining. They testified that a parametric interpolator has advantageous in machining freeform geometry in terms of high speed and tight tolerance [2]. Among the parametric curve interpolations, the non-uniform rational B-spline (NURBS) even became the standard format of freeform curve and surface in 1990 because NURBS method offered a common mathematical form for the precise representation of standard analytical shapes as well as freeform curves and surfaces [3].

However, the early NURBS curve interpolation algorithms just maintained constant feedrate without taking into account chord errors [4]. Although these were simple solutions to parametric interpolator, the generated feedrate fluctuated severely. In literatures [1] and [5], first-order and second-order approximation interpolation algorithms were proposed using Euler and Taylor's expansions, respectively. Cheng et al. compared several numerical interpolation approaches and suggested that Taylor's second-order approximation was a good approach for real-time command generation [5]. Thus, Taylor's second-order approximation is adopted in this paper. To achieve uniform feedrate and confined chord error, Tsai and Cheng proposed a closed-loop predictor–corrector interpolator (PCI), in which the deviation between the current and desired feedrate was forced into a specified tolerance through a feedback compensation scheme [6]. By using PCI method, the

H. Dong · B. Chen (✉) · Y. Chen · J. Xie
National NC System Engineering Research Center,
Huazhong University of Science and Technology,
Wuhan 430074, China
e-mail: chenbing@mail.hust.edu.cn

H. Dong
e-mail: surge_d@163.com

Z. Zhou
Hubei Digital Manufacturing Key Laboratory,
Wuhan University of Technology,
Wuhan 430070, China

feedrate fluctuation can be controlled through setting tolerance of feedrate error for either given constant or variable feedrate command. Erkorkmaz and Altintas proposed a quintic spline interpolation method to minimize feedrate fluctuation [7]. Lei et al. proposed a novel fast real-time NURBS path interpolation method, which generated inverse length functions (ILF) for each parameter subinterval in off-line [8]. Because this method divides the NURBS curve into subintervals using of Simpson's rule to build ILF function and it needs to deal with the occasion of multiple control points, it is time consuming and may require long-time pre-processing. Zhiming et al. presented a curvature-based interpolation algorithm based on the curvature of curves [9]. But the continuity of the federate cannot be guaranteed. Yeh and Hsu proposed an adaptive-feedrate interpolator to adjust curve speed according to chord tolerance [10]. Tikhon et al. employed the adaptive feedrate algorithm and took into consideration constant material removal rate to improve machining accuracy [11]. Although some of the above researchers take into account chord errors, they do not consider feedrate fluctuation, the Acc/Dec and the jerk might be beyond the limits if the curvature of a curve changes abruptly.

To obtain a smooth feedrate profile, the Acc/Dec should be considered in the design process. A few researchers have made great progress in this field. Luo and Zhou proposed a universal velocity profiles generation approach and corresponding optimal look-ahead algorithm based on dynamic back tracking along with a doubly linked list [12]. But this method is only used in the conventional linear interpolation. Meng and Hao proposed an integrated Acc/Dec interpolation scheme by using digital convolution technique to generate velocity profiles [13]. But this digital convolution approach is unable to generate various velocity profiles which are useful for CNC machine tools. Du and Liu considered the maximum Acc/Dec of the machine tool, and proposed an adaptive parametric curve interpolator with a real-time look-ahead function [14].

However, the aforementioned interpolation algorithms [1–15] have not explicitly considered jerk level, and the effect of jerk to machining process is also ignored. In the actual machining, high jerks mean dramatic changes on Acc/Dec profile, which may be out of the machine tool drivers' ability. In order to obtain shock-free and smooth machining process, some researchers proposed jerk-limited trajectory planning [15–21]. Nam and Yang proposed a real-time interpolation scheme, which considered chord errors, feedrate fluctuations, and jerk limitation in the design [15]. Liu and Lin proposed real-time look-ahead algorithms by shifting back 400 points [17] and 800 points [18], respectively. Sekar and Narayanan developed jerk-bounded feedrate profile with ripple effect for adaptive NURBS interpolator [19]. In order to improve the data utilization and simplify the calculation process, circular buffers [20] and structure arrays [21] were designed respectively for storing a sequence of the previous interpolation points, which were similar to the work in literature [22]. All these algorithms strove to achieve uniform feedrate, confined chord errors, and kinetics control over the trajectory and minimize deflection cutting forces. Nevertheless, some issues were not discussed in these researches. In the trace-back process, most of the existing NURBS interpolation algorithms only considered long spline interpolation, in which the feedrate of the deceleration point is equal to the command feedrate. Even though a few NURBS interpolation algorithms considered the case that the feedrate of the deceleration point is not equal the command feedrate, their Acc/Dec planning is poor, and their machining process is not efficient, either.

To overcome the disadvantages mentioned above and satisfy the need for high-speed and high-precision machining of NURBS curves, a novel and accurate real-time NURBS curve interpolation algorithm is proposed in this study. This algorithm not only considers chord errors, feedrate fluctuations, jerk-limited, and Acc/Dec capabilities of the machine, but also optimizes the look-ahead process. Furthermore, the proposed method can also solve the short spline interpolation where sensitive points are close to each other.

The remainder of the paper is organized as follows. Section 2 reviews elementary knowledge of NURBS curve interpolation algorithm. In Section 3, the novel and more accurate real-time NURBS curve interpolation algorithm with short spline interpolation capacity is presented. The simulation and experimental results are provided to demonstrate the effectiveness and universality of the proposed algorithm in Section 4. Finally, Section 5 concludes with a brief summary.

2 NURBS curve interpolation algorithm

NURBS is a mathematical model commonly used for generating and representing curves and surfaces. In this study, NURBS is used to represent a parametric curve, and thus it is introduced firstly.

2.1 Introduction to NURBS curve

The general form of a NURBS curve is defined as [3]:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \quad a \le u \le b \tag{1}$$

where p_i is the control points, w_i is the weights on every control point, u is the parameter, and $N_{i,p}(u)$ is the ith B-spline basis function of the *p*th degree defined on the non-periodic knot vector

$$U=\left\{\underbrace{a,\cdots a,}_{p+1}u_{p+1},\cdots,u_{m-p-1},\underbrace{b,\cdots b}_{p+1}\right\}\quad(m=n+p+1)$$

The *p*th degree B-spline basis function is recursively defined as:

$$\begin{cases}N_{i,0}(u)=\begin{cases}1 & \text{if } u_i\le u\le u_{i+1}\\ 0 & \text{otherwise}\end{cases}\\ N_{i,p}(u)=\frac{u-u_i}{u_{i+p}-u_i}N_{i,p-1}(u)+\frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}}N_{i+1,p-1}(u)\end{cases}\tag{2}$$

The *k*th derivative of the basis function is derived from Eq. (2). It is used to calculate the curve curvature and the curve curvature radius. The general formulation is given as follows [3]:

$$N_{i,p}^{(k)}(u)=\mathrm{p}\left(\frac{N_{i,p-1}^{(k-1)}(u)}{u_{i+p}-u_i}+\frac{N_{i+1,p-1}^{(k-1)}(u)}{u_{i+p+1}-u_{i+1}}\right)\tag{3}$$

2.2 Principle of NURBS curve interpolation

To implement NURBS interpolation, a second-order approximation interpolation algorithm is utilized. By using the Taylor series expansion method, the approximation up to the second derivative is given as follows:

$$u_{i+1}=u_i+\left.\frac{du}{dt}\right|_{t=t_i}T_s+\frac{1}{2}\left.\frac{d^2u}{dt^2}\right|_{t=t_i}T_s^2+\text{H.O.T}\tag{4}$$

where T_s is the interpolation period (sampling period), and H.O.T. stands for the high-order term in Taylor expansion, which is neglected. The $V(u_i)$ along with the curve $C(u)$ is defined as:

$$V_i=V(u_i)=\left\|\frac{dC(u)}{dt}\right\|_{u=u_i}=\left\|\frac{dC(u)}{du}\right\|_{u=u_i}\cdot\left.\frac{du}{dt}\right|_{t=t_i}\tag{5}$$

Therefore, the first-order derivative of *u* to the time *t* is derived from the Eq. (5) as:

$$\left.\frac{du}{dt}\right|_{t=t_i}=\frac{V_i}{\left\|\frac{dC(u)}{du}\right\|_{u=u_i}}\tag{6}$$

From the result of the first-order derivative of *u* to the time *t*, the second derivative is given as:

$$\left.\frac{d^2u}{dt^2}\right|_{t=t_i}=-\frac{V_i^2\cdot\left.\left(\frac{dC(u)}{du}\cdot\frac{d^2C(u)}{du^2}\right)\right|_{u=u_i}}{\left\|\frac{dC(u)}{du}\right\|_{u=u_i}^4}\tag{7}$$

By neglecting the high-order term from Eq. (4), the second-order approximation on the parameter $u=u_i$ becomes:

$$u_{i+1}=u_i+\frac{V_i\cdot T_s}{\left\|\frac{dC(u)}{du}\right\|_{u=u_i}}-\frac{V_i^2\cdot T_s^2\cdot\left.\left(\frac{dC(u)}{du}\cdot\frac{d^2C(u)}{du^2}\right)\right|_{u=u_i}}{2\cdot\left\|\frac{dC(u)}{du}\right\|_{u=u_i}^4}\tag{8}$$

where V_i (i.e., $V(u_i)$) can be either the command feedrate or the planning feedrate according to the S curve Acc/Dec profile.

3 The NURBS interpolation algorithm

The system architecture of a multi-axis motion controller is given in Fig. 1 and it is utilized to implement CNC control tasks. The controller consists of three main programs: CNC interpreter, NURBS interpolator, and motion controller. The CNC interpreter reads NC strings from NC files to generate and store NC blocks in the buffer. The NURBS interpolator mainly includes two function modules: NURBS parametric interpolator and NURBS position interpolator. The motion controller receives outputs from the interpolation module, and achieves closed-loop motion control according to the feedback information from the encoder of each motion axis. Here, position interpolator and motion controller are positioned in the interrupt service routine (ISR).

The NURBS parametric interpolator is the main content of this research. It consists of three parts: look-ahead, trace-back, and motion planning. Since tracing back and re-interpolation need to modify interpolated data, a large circular buffer is established to store interpolated parameters from the NURBS parametric interpolator.

3.1 Chord error

In order to achieve high-precision machining, the chord error must be controlled under the prescribed tolerance during interpolation. As shown in Fig. 2, an arc approximation is adopted to calculate the chord error δ_i.

$$\delta_i=\rho_i-\sqrt{\rho_i^2-\left(\frac{L_i}{2}\right)^2}=\rho_i-\sqrt{\rho_i^2-\left(\frac{v_i\cdot T_s}{2}\right)^2}\tag{9}$$

Rearranging Eq. 9,

$$v_i=\frac{2}{T_s}\sqrt{\rho_i^2-(\rho_i-\delta_i)^2}\tag{10}$$

where $L_i=\|C(u_{i+1})-C(u_i)\|=v_i\cdot T_s$. It represents the distance of an interpolation interval between interpolation point $C(u_i)$ and $C(u_{i+1})$; $\rho_i(\rho_i=1/K_i)$ and K_i are the curvature radius and the curvature at $C(u_i)$ of the curve $C(u)$,

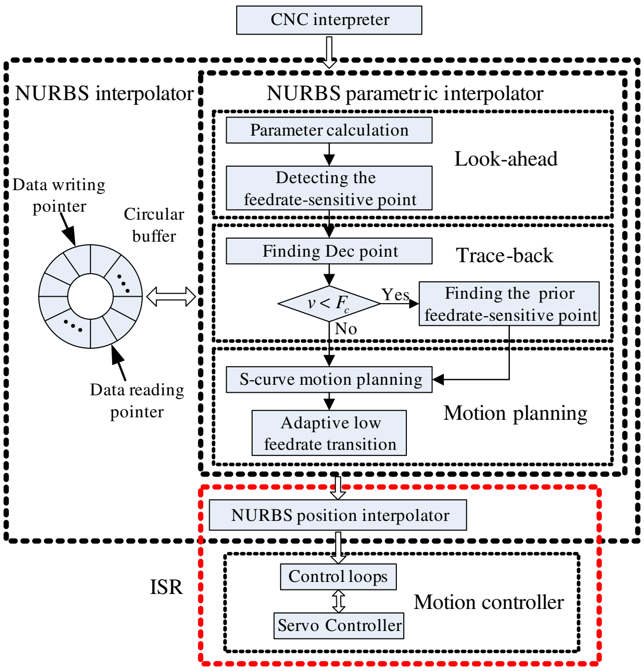

Fig. 1 System architecture of a multi-axis motion controller

respectively. The curvature radius ρ_i is given by the following equation.

$$K_i = K(u_i) = \left. \frac{\left\| \frac{dC(u)}{du} \times \frac{dC^2(u)}{du^2} \right\|}{\left\| \frac{dC(u)}{du} \right\|^3} \right|_{u=u_i} \tag{11}$$

3.2 Look-ahead

According to Eq. (9), the chord error is heavily influenced by the feedrate. Basically, the higher the feedrate is, the larger the chord error would be. During machining, it is desirable to keep feedrate constant as much as possible in order to achieve high surface machining quality. However, this may lead to undesired chord errors. The proposed adaptive feedrate algorithm can ensure that the chord error does not exceed the tolerance. From Eq. (10), if the allowed

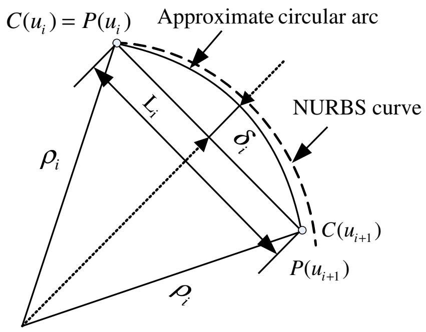

Fig. 2 The chord error

chord error is $\delta_{\max}$, the algorithm of determining the federate v_i adaptively is given as:

$$v_i = \begin{cases} F_c & \text{if } \frac{2}{T_s}\sqrt{\rho_i^2 - (\rho_i - \delta_{\max})^2} F_c \\ \frac{2}{T_s}\sqrt{\rho_i^2 - (\rho_i - \delta_{\max})^2} & \text{if } \frac{2}{T_s}\sqrt{\rho_i^2 - (\rho_i - \delta_{\max})^2} \le F_c \end{cases} \tag{12}$$

where F_c represents the command feedrate.

In the look-ahead stage, a series of interpolation parameters (including u parameters, feedrate, and cumulative chord length, etc.) will be produced step by step, and be written to the circular buffer in turn. The look-ahead algorithm is given as follows:

Step 1: Calculate the feedrate v_i based on S curve acceleration profile. Let the feedrate v_i equal to command feedrate F_c when it peaks.
Step 2: Calculate u_{i+1} from v_i and u_i in terms of Eq. (8).
Step 3: Calculate chord error δ_i from the parameter u_i using Eq. (9).
Step 4: If $\delta_i > \delta_{\max}$, then adjust the feedrate according to Eq. (12), and enter the detecting the feedrate-sensitive point module. Otherwise, let $i=i+1$ and jump to the Step 1.

The detecting the feedrate-sensitive point algorithm is given as follows:

Step 1: let $v_i = v_{i-1}$, and calculate u_i from v_{i-1} and u_{i-1} in terms of Eq. (8).
Step 2: Calculate chord error δ_i from the parameter u_i using Eq. (9).
Step 3: If $\delta_i > \delta_{\max}$, adjust the feedrate according to Eq. (12), then let $i=i+1$, and jump to the step 1. Otherwise, the feedrate-sensitive point has been found, and exit the detecting the feedrate-sensitive point module.

3.3 Trace-back

The purpose of tracing back is to find the deceleration point. A brief introduction is given as follows.

Step 1: Calculate the travel length S of deceleration zone based on the cumulative chord length of the current feedrate-sensitive point and the prior feedrate-sensitive point.
Step 2: Compute the theoretical decelerating zone length $S_{\min}$ according to the feedrate of the current feedrate-sensitive point.
Step 3: If $S > S_{\min}$, then let the cumulative chord length equal to $S_E - S_{\min}$, and use a bisection searching method in the circular buffer to determine the deceleration point. Otherwise, the prior feedrate-

sensitive point is the deceleration point. Here, S_E is the cumulative chord length of the current feedrate-sensitive point.

Note: when $S>S_{\min}$, the found deceleration point is not always the actual deceleration point, and it may be a pseudo deceleration point. The process of bisection searching for determining deceleration point is carried out in the circular buffer and does not produce new interpolation parameters. In addition, adaptive beginning point is the first interpolation point of adaptive feedrate algorithm to find the local lowest feedrate point, and feedrate-sensitive point is those points which are on the curve and whose feedrate need to be adjusted according to the given chord error.

3.3.1 The first case of the deceleration point: $v = F_c$

In this case, interpolation curve is either the long spline or its feedrate-sensitive point is far away each other. As shown in Fig. 3, the feedrate of the found deceleration point is equal to the maximum command feedrate ($v=F_c$). Then the found deceleration point is the actual deceleration point.

For the case shown in Fig. 3a, deceleration point is point A (actual Dec point), and the prior feedrate-sensitive point is located on the left side of the deceleration point A, then the prior feedrate-sensitive point information does not need to be modified. In Fig. 3b, the found deceleration point is also point A (actual Dec point), but the prior feedrate-sensitive point is located on the right side of the deceleration point A, then the information of saved feedrate at sensitive point D (the prior feedrate-sensitive point) needs to be deleted.

3.3.2 The second case of the deceleration point: $v < F_c$ and $v_e > v_s$

In this case, the interpolation curve is either the shorter spline or its feedrate-sensitive point is closer each other. As shown in Fig. 4, the feedrate of the found deceleration point is less than the maximum command feedrate ($v<F_c$), and the feedrate of the end point (v_e) is greater than that of the starting point (v_s), i.e., $v_e>v_s$.

If $S>S_{\min}$, shown in Fig. 4a, the found deceleration point De is a pseudo deceleration point. The actual deceleration

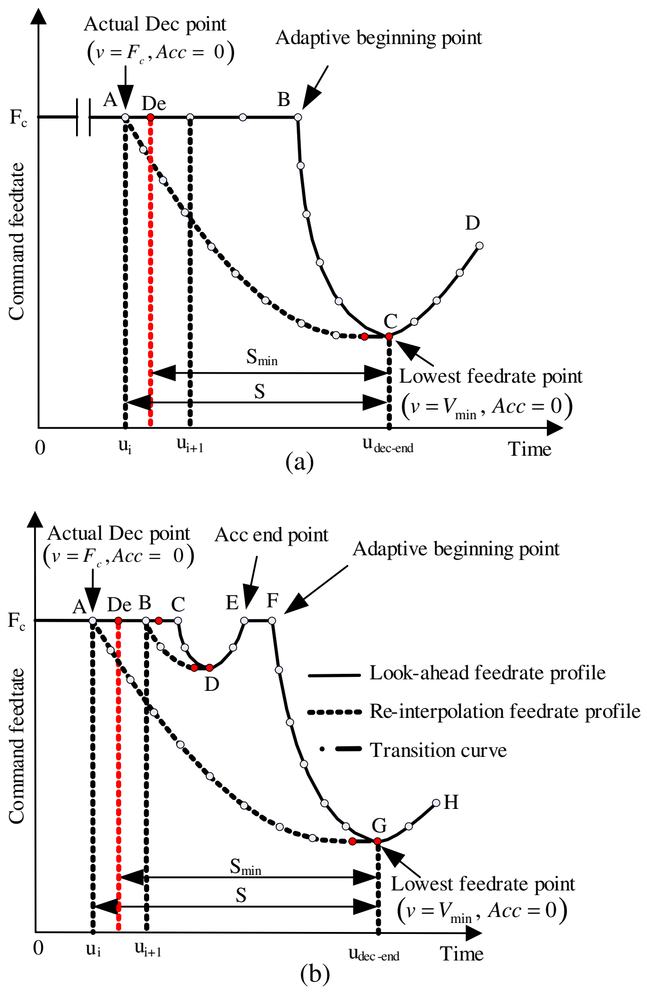

Fig. 3 The first case ($v=F_c$)

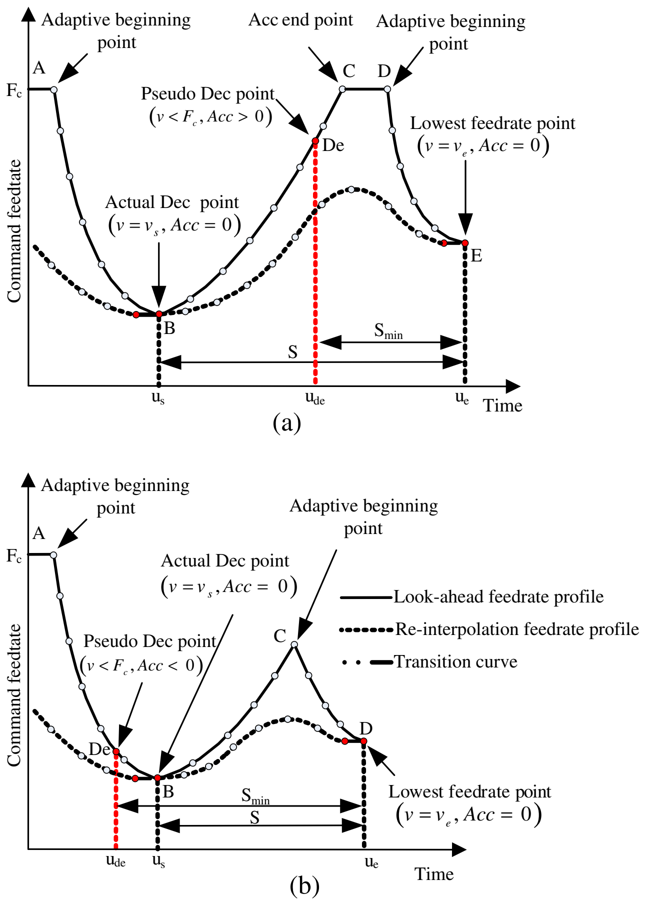

Fig. 4 The second case ($v<F_c$ and $v_e>v_s$)

point is the prior feedrate-sensitive point B, just in front of the current feedrate-sensitive point. If $S<S_{\min}$, shown in Fig. 4b, the found deceleration point De is also a pseudo deceleration point. After the pseudo deceleration point, the local feedrate-sensitive point B is the actual deceleration point.

3.3.3 The third case of the deceleration point: $v < F_c$ and $v_e < v_s$

As shown in Fig. 5, this case is similar to the second case. But the feedrate of the end point (v_e) is less than that of the starting point (v_s), i.e., $v_e<v_s$.

If $S>S_{\min}$, shown in Fig. 5a, the found deceleration point De is a pseudo deceleration point. The actual deceleration point is the prior feedrate-sensitive point B, just in front of the current feedrate-sensitive point. If $S<S_{\min}$, shown in Fig. 5b, the found deceleration point De is also a pseudo deceleration point. After the pseudo deceleration point, the local feedrate-sensitive point B is the actual deceleration point.

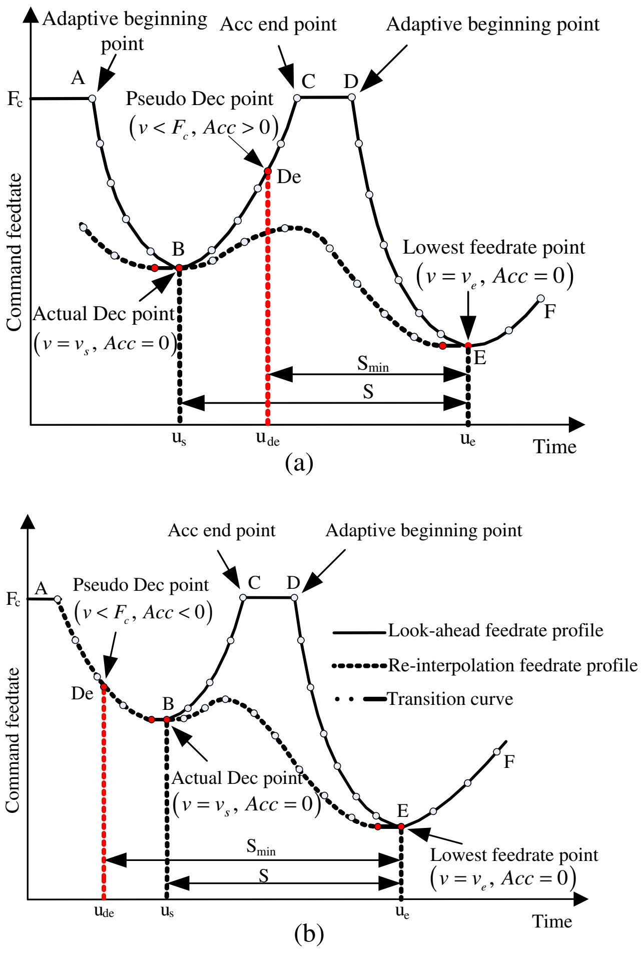

Fig. 5 The third case ($v<F_c$ and $v_e<v_s$)

3.4 Motion planning (feedrate planning)

After the deceleration point is obtained, the motion profile between the present position (prior feedrate-sensitive point) and the target position (current feedrate-sensitive point) needs to be re-interpolated and planned. The motion planning process will also produce a series of interpolation parameters step by step, including the u parameters, feedrate, and the cumulative chord length, etc. They will be written to the circular buffer in turn, and replace the previous old parameters in it. When the circular buffer is full, the parametric interpolator process will be suspended, and it will restart when the circular buffer is not full.

3.4.1 The first case: S curve deceleration motion planning ($v_s = F_c$)

In this paper, an S curve Acc/Dec control before interpolation is adopted. Typically, the increasing feedrate stage includes three sub-stages: increasing acceleration stage, constant acceleration stage, and decreasing acceleration stage; the decreasing feedrate stage includes: increasing deceleration stage, constant deceleration stage, and decreasing deceleration stage. The acceleration and deceleration stage are jointed by constant feedrate stage.

When the found deceleration point is in the position shown in Fig. 3, the re-interpolation profile is only the S curve deceleration motion planning process, and according to the difference of F_c and $V_{\min}$ (the local minimum feedrate), motion planning process is divided into two cases: Trapezoidal decreasing feedrate motion planning and triangular decreasing feedrate motion planning. Its profiles, including the distance, feedrate, Dec, and jerk profiles are shown in Fig. 6. Here, a_m and j_m represent maximum acceleration and jerk, respectively.

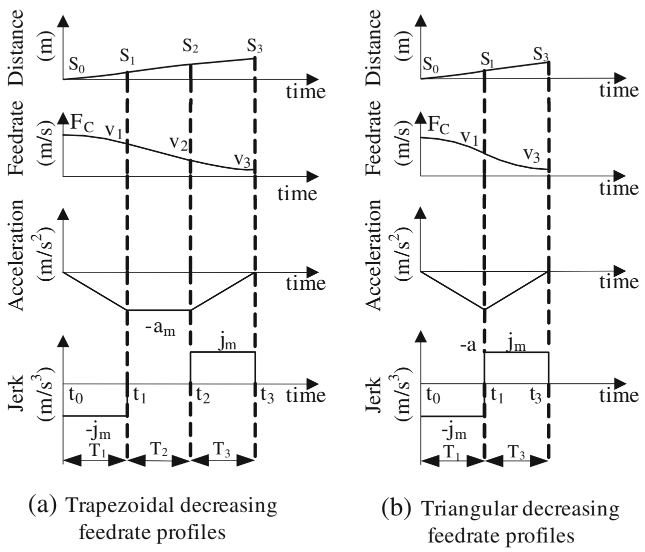

Fig. 6 S curve Dec motion planning

1. Trapezoidal decreasing feedrate profiles: $F_c - V_{\min}{}^{3}\frac{a_m^2}{j_m}$
   The planning profile is shown in Fig. 6a, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 = \frac{a_m}{j_m} \\ T_2 = \frac{F_c - V_{\min}}{a_m} - \frac{a_m}{j_m} \end{cases} \tag{13}$$

$$\begin{cases} v = F_c - \frac{1}{2}\cdot j_m \cdot \tau^2 & t_0 \le t < t_1(\tau = t - t_0) \\ v = v_1 - a_m \cdot \tau & t_1 \le t < t_2(\tau = t - t_1) \\ v = v_2 - a_m \cdot \tau + \frac{1}{2}\cdot j_m \cdot \tau^2 & t_2 \le t < t_3(\tau = t - t_2) \end{cases} \tag{14}$$

where

$$\begin{cases} v_1 = F_c - \frac{1}{2}\cdot j_m \cdot T_1^2 \\ v_2 = v_1 - a_m \cdot T_2 \end{cases}$$

For the trapezoidal profile, the total deceleration distance is derived as:

$$S_{\min} = \frac{1}{2}\cdot (F_c + V_{\min}) \cdot \left(\frac{a_m}{j_m} + \frac{F_c - V_{\min}}{a_m}\right) \tag{15}$$

2. Triangular decreasing feedrate profiles: $F_c - V_{\min}\frac{a_m^2}{j_m}$
   The planning profile is shown in Fig. 6b, and each parameter is calculated as follows:

$$\begin{cases} a = \sqrt{(F_c - V_{\min})\cdot j_{\mathrm{m}}} \\ T_1 = T_2 = \frac{a}{j_{\mathrm{m}}} = \sqrt{\frac{(F_c - V_{\min})}{j_{\mathrm{m}}}} \end{cases} \tag{16}$$

$$\begin{cases} v = F_c - \frac{1}{2}\cdot j_m \cdot \tau^2 & t_0 \le t < t_1(\tau = t - t_0) \\ v = v_1 - a \cdot \tau + \frac{1}{2}\cdot j_m \cdot \tau^2 & t_1 \le t < t_2(\tau = t - t_1) \end{cases} \tag{17}$$

where

$$v_1 = F_c - \frac{1}{2}\cdot j_m \cdot T_1^2$$

For the triangular profile, the total deceleration distance is derived as:

$$S_{\min} = (F_c + V_{\min}) \cdot \sqrt{\frac{F_c - V_{\min}}{j_m}} \tag{18}$$

3.4.2 The second case: S curve Acc/Dec motion planning ($v_e > v_s$)

When the found deceleration point is in the position shown in Figs. 4 and 5, the re-interpolation profile is the Acc/Dec motion planning process. In other words, it may also include the process of acceleration and deceleration stages at the same time. According to the given starting point feedrate v_s, command feedrate F_c, end point feedrate v_e and to be re-interpolated curve length S, the S curve motion planning may have a lot of cases. In some cases, the deceleration distance may be too short, and the feedrate of the sensitive point needs to be adjusted locally.

When $v_e > v_s$, in accordance with the proposed algorithm of finding the deceleration point, the maximum number of segments of the motion planning is six. Because the motion planning of the seven segments indicate that the maximum command feedrate can be achieved, this case will only have the motion planning of decreasing feedrate. In fact, the S curve motion planning of the seven segments does not appear. The motion planning is classified into eight types, and all of the profiles are shown in Fig. 7.

Firstly, algorithms determine the critical interval (distance and feedrate), and the interval is divided according to it. Then determine which interval the distance is in by calculating the deceleration distance. Finally, the motion is

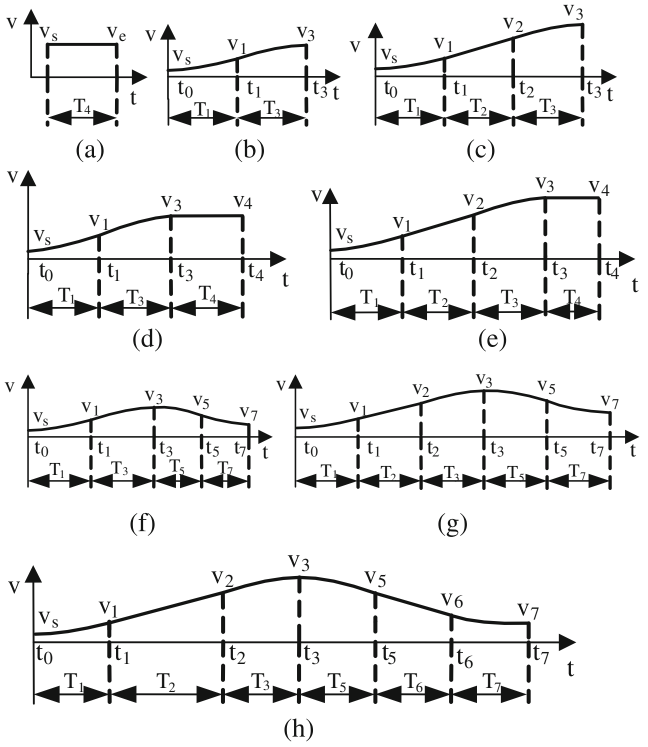

Fig. 7 Eight profiles (**a**–**h**) of motion planning ($v_e > v_s$)

planned according to the corresponding interval. The interval divisions are shown in Fig. 8.

The specific parameters are calculated as follows:

$$\begin{cases} S_5 = \frac{2\cdot a_m{}^3}{j_m{}^2} + \frac{1}{2}\left[\frac{a_m(7v_e+v_s)}{j_m} + \frac{v_e(v_e-v_s)}{a_m}\right] \\ v_5 = v_e + \frac{a_m{}^2}{j_m} \end{cases} \tag{19}$$

$$\begin{cases} S_4 = \frac{a_m{}^3}{j_m{}^2} + \frac{2a_m\cdot v_s}{j_m} + \left(v_e + v_s + \frac{a_m{}^2}{j_m}\right)\frac{\sqrt{a_m{}^2 - j_m(v_e-v_s)}}{j_m} \\ v_4 = v_s + \frac{a_m{}^2}{j_m} \end{cases} \tag{20}$$

$$\begin{cases} S_3 = \frac{v_e^2-v_s^2}{2\cdot a_m} + \frac{a_m(v_e+v_s)}{2\cdot j_m} \\ v_3 = v_e \end{cases} \tag{21}$$

$$\begin{cases} S_2 = (v_e+v_s)\cdot\sqrt{\frac{v_e-v_s}{j_m}} \\ v_2 = v_e \end{cases} \tag{22}$$

1. One segment motion planning ($S<S_2$)

   As shown in Fig. 9, interpolation curve is either the very short spline or its feedrate-sensitive point is close to each other. When $S<S_2$, the feedrate of starting point B is less than that of the end point D. In this case, the feedrate of point D needs to be adjusted to the feedrate of the starting point B. In other words, let $v_e=v_s$, and then the motion is planned according to the adjusted feedrate.

   The planning profile is shown in Fig.7a, and the parameter is calculated as follows:

$$T_4 = \frac{S}{v_s} \tag{23}$$

2. Two segments motion planning ($S=S_2$)

   The planning profile is shown in Fig. 7b, and each parameter is calculated as follows:

$$T_1 = T_3 = \sqrt{\frac{v_e - v_s}{j_m}} \tag{24}$$

$$\begin{array}{ll} v = v_s + \frac{1}{2}\cdot j_m\cdot\tau^2 & t_0 \le t < t_1(\tau = t - t_0) \\ v = v_1 + j_m\cdot T_1^2 - \frac{1}{2}\cdot j_m\cdot\tau^2 & t_1 \le t < t_3(\tau = t - t_1) \end{array} \tag{25}$$

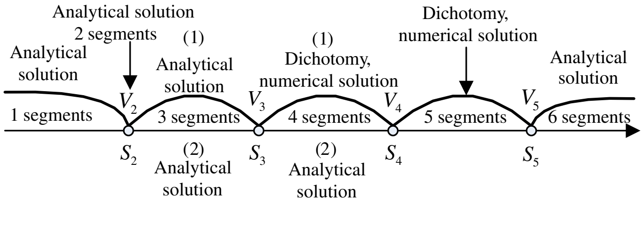

Fig. 8 The interval division ($v_e>v_s$)

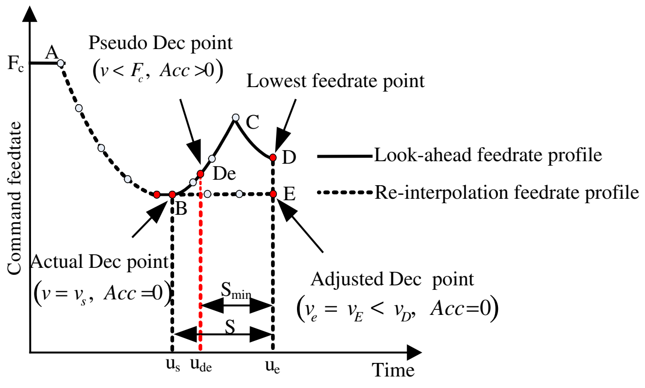

Fig. 9 Motion planning of short spline ($v_e>v_s$)

where

$$v_1 = v_s + \frac{1}{2}\cdot j_{\mathrm{m}}\cdot T_1^2$$

3. Three segments motion planning ($S_3 \ge S > S_2$)

   (a) $v_e v_s, v_e - v_s \frac{a_m^2}{j_m}$

   The planning profile is shown in Fig. 7c, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 = \frac{a_m}{j_m} \\ T_2 = \frac{v_e - v_s}{a_m} - T_1 \end{cases} \tag{26}$$

$$\begin{cases} v = v_s + \frac{1}{2}\cdot j_{\mathrm{m}}\cdot\tau^2 & t_0 \le t < t_1(\tau = t - t_0) \\ v = v_1 + a_{\mathrm{m}}\cdot\tau & t_1 \le t < t_2(\tau = t - t_1) \\ v = v_2 + a_{\mathrm{m}}\cdot\tau - \frac{1}{2}\cdot j_{\mathrm{m}}\cdot\tau^2 & t_2 \le t < t_3(\tau = t - t_2) \end{cases} \tag{27}$$

where

$$\begin{cases} v_1 = v_s + \frac{1}{2}\cdot j_m\cdot T_1^2 \\ v_2 = v_1 + a_{\mathrm{m}}\cdot T_2 \end{cases}$$

   (b) $v_e > v_s, v_e - v_s < \frac{a_m^2}{j_m}$

   The planning profile is shown in Fig. 7d, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 = \sqrt{\frac{v_e - v_s}{j_m}} \\ T_4 = \frac{S - S_3}{v_e} \end{cases} \tag{28}$$

   where

$$S_3 = \frac{v_e^2 - v_s^2}{2\cdot a_m} + \frac{a_m(v_e+v_s)}{2\cdot j_m}$$

$$\begin{cases} v = v_s + \frac{1}{2}\cdot j_{\mathrm{m}}\cdot\tau^2 & t_0 \le t < t_1(\tau = t - t_0) \\ v = v_1 + j_{\mathrm{m}}\cdot T_1^2 - \frac{1}{2}\cdot j_{\mathrm{m}}\cdot\tau^2 & t_1 \le t < t_3(\tau = t - t_1) \\ v = v_e & t_3 \le t < t_4(\tau = t - t_3) \end{cases} \tag{29}$$

where

$$v_1 = v_s + \frac{1}{2} \cdot j_m \cdot T_1^2$$

4. Four segments motion planning ($S_4 \geq S > S_3$)

   (a) $v_e > v_s, v_e - v_s > \frac{a_m^2}{j_m}$

   The planning profile is shown in Fig. 7e, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 = \frac{a_m}{j_m} \\ T_2 = \frac{v_e - v_s}{a_m} - T_1 \\ T_4 = \frac{S - S_4}{v_e} \end{cases} \tag{30}$$

   where

$$S_4 = \frac{{a_m}^3}{{j_m}^2} + \frac{2a_m \cdot v_s}{j_m} + \left(v_e + v_s + \frac{{a_m}^2}{j_m}\right) \frac{\sqrt{{a_m}^2 - j_m(v_e - v_s)}}{j_m}$$

$$\begin{cases} v = v_s + \frac{1}{2} \cdot j_m \cdot \tau^2 & t_0 \leq t < t_1(\tau = t - t_0) \\ v = v_1 + a_m \cdot \tau & t_1 \leq t < t_2(\tau = t - t_1) \\ v = v_2 + a_m \cdot \tau - \frac{1}{2} \cdot j_m \cdot \tau^2 & t_2 \leq t < t_3(\tau = t - t_2) \\ v = v_e & t_3 \leq t < t_4(\tau = t - t_3) \end{cases} \tag{31}$$

where

$$\begin{cases} v_1 = v_s + \frac{1}{2} \cdot j_m \cdot T_1^2 \\ v_2 = v_1 + a_m \cdot T_2 \end{cases}$$

   (b) $v_e > v_s, v_e - v_s < \frac{a_m^2}{j_m}$

   The planning profile is shown in Fig. 7f, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 \\ T_5 = T_7 \end{cases} \tag{32}$$

$$\begin{cases} v_{\max} = v_3 = v_s + j_m \cdot T_1^2 \\ 2 \cdot (v_s \cdot T_1 + v_e \cdot T_5) + j_m \cdot \left(T_1^3 + T_5^3\right) = S \\ v_e - v_s = j_m \cdot \left(T_1^2 - T_5^2\right) \end{cases} \tag{33}$$

Here, the $v_{\max}$ is in the interval $\left(v_e, v_s + \frac{{a_m}^2}{j_m}\right)$, let the initial value $v_{\max}^1 = \left(v_s + v_e + {a_m}^2/j_m\right)/2$, and use a bisection searching method to calculate the value of $v_{\max}$. After $v_{\max}$ is obtained, T_1 and T_5 can be figured out from Eqs. (32) and (33), and then the motion can be planned.

$$\begin{cases} v = v_s + \frac{1}{2} \cdot j_m \cdot \tau^2 & t_0 \leq t < t_1(\tau = t - t_0) \\ v = v_1 + j_m \cdot T_1^2 - \frac{1}{2} \cdot j_m \cdot \tau^2 & t_1 \leq t < t_3(\tau = t - t_1) \\ v = v_3 - \frac{1}{2} \cdot j_m \cdot \tau^2 & t_3 \leq t < t_5(\tau = t - t_3) \\ v = v_5 - j_m \cdot T_5^2 + \frac{1}{2} \cdot j_m \cdot \tau^2 & t_5 \leq t < t_7(\tau = t - t_5) \end{cases} \tag{34}$$

where

$$\begin{cases} v_1 = v_s + \frac{1}{2} \cdot j_m \cdot T_1^2 \\ v_2 = v_1 + a_m \cdot T_2 \end{cases}$$

5. Five segments motion planning ($S_5 \geq S > S_4$)

   The planning profile is shown in Fig. 7g, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 = \frac{a_m}{j_m} \\ T_5 = T_7 \end{cases} \tag{35}$$

$$\begin{cases} v_{\max} = v_3 = v_s + a_m \cdot T_2 + \frac{{a_m}^2}{j_m} \\ T_5 = \frac{\sqrt{a_m^2 + j_m(a_m \cdot T_2 - v_e + v_s)}}{j_m} \end{cases} \tag{36}$$

Here, the $v_{\max}$ is in the interval $\left(v_s + \frac{{a_m}^2}{j_m}, v_e + \frac{{a_m}^2}{j_m}\right)$, let the initial value $v_{\max}^1 = \left[v_s + v_e + \left(2 \cdot {a_m}^2\right)/j_m\right]/2$, and use a bisection searching method to calculate the value of $v_{\max}$. After $v_{\max}$ is obtained, T_2 and T_5 can be figured out from Eqs. (35) and (36), and then the motion can be planned.

6. Six segments motion planning ($S > S_5$)

   The planning profile is shown in Fig. 7h, and each parameter is calculated as follows:

$$\begin{cases} T_1 = T_3 = T_5 = T_7 = \frac{a_m}{j_m} \\ T_2 = \frac{-\left(2 \cdot v_s \cdot j_m + 3 \cdot {a_m}^2\right)}{2 \cdot j_m \cdot a_m} + \frac{\sqrt{{a_m}^4 - 2 \cdot j_m \cdot \left[a_m^2 (v_s + v_e) - j_m\left(v_s^2 + v_e^2\right) - 2 \cdot a_m \cdot j_m \cdot S\right]}}{2 \cdot j_m \cdot a_m} \\ T_6 = T_2 - \frac{v_e - v_s}{a_m} \end{cases} \tag{37}$$

$$\begin{cases} v = v_s + \frac{1}{2} \cdot j_m \cdot \tau^2 & t_0 \leq t < t_1(\tau = t - t_0) \\ v = v_1 + a_m \cdot \tau & t_1 \leq t < t_2(\tau = t - t_1) \\ v = v_2 + a_m \cdot \tau - \frac{1}{2} \cdot j_m \cdot \tau^2 & t_2 \leq t < t_3(\tau = t - t_2) \\ v = v_3 - \frac{1}{2} \cdot j_m \cdot \tau^2 & t_3 \leq t < t_5(\tau = t - t_3) \\ v = v_5 - a_m \cdot \tau & t_5 \leq t < t_6(\tau = t - t_5) \\ v = v_6 - a_m \cdot \tau + \frac{1}{2} \cdot j_m \cdot \tau^2 & t_6 \leq t < t_7(\tau = t - t_6) \end{cases} \tag{38}$$

Where

$$\begin{cases} v_1 = v_s + \frac{1}{2} \cdot j_m \cdot T_1^2 \\ v_2 = v_1 + a_m \cdot T_2 \\ v_3 = v_2 + a_m \cdot T_3 - \frac{1}{2} \cdot j_m \cdot T_3^2 \\ v_5 = v_3 - \frac{1}{2} \cdot j_m \cdot T_5^2 \\ v_6 = v_5 - a_m \cdot T_6 \\ v_7 = v_6 - a_m \cdot T_7 + \frac{1}{2} \cdot j_{\max} \cdot T_7^2 \end{cases}$$

3.4.3 The third case: S curve Acc/Dec motion planning ($v_e < v_s$)

As shown in Fig. 5, when $v_e < v_s$, based on the feedrate and distance between the starting point and end point, the

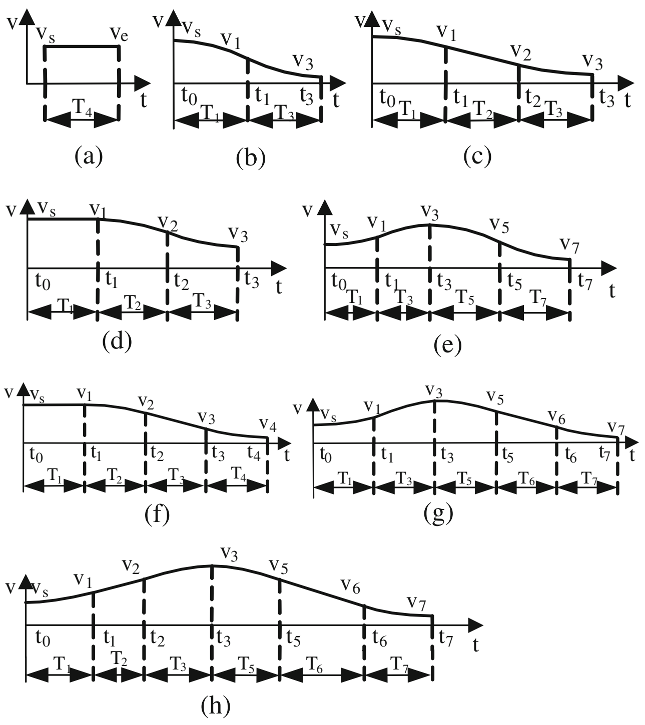

Fig. 10 Eight profiles (**a**–**h**) of motion planning ($v_e < v_s$)

motion planning has also eight types, which are shown in Fig. 10.

As shown in Fig. 11, it is the very short spline or the sensitive point close to each other. When $S < S2$, the feedrate of the starting point B is higher than that of the end point D. In this case, the feedrate of point B needs to be adjusted according to the feedrate of the end point D, that is, let $v_s = v_e$. After the feedrate of starting point is adjusted, it needs to continue tracing back, and then the motion is planned according to the adjusted feedrate. The motion planning of

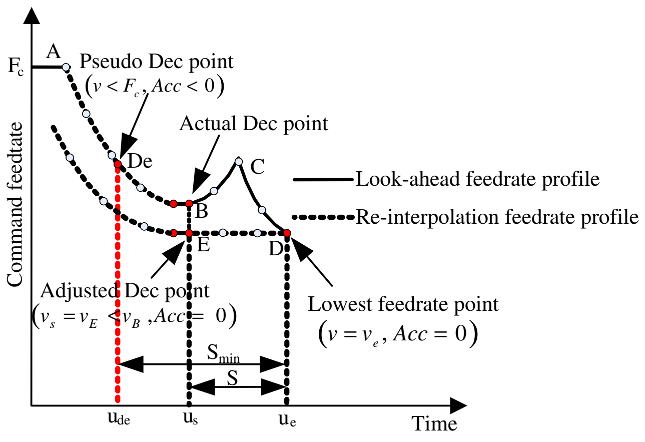

Fig. 11 Motion planning of short spline ($v_e < v_s$)

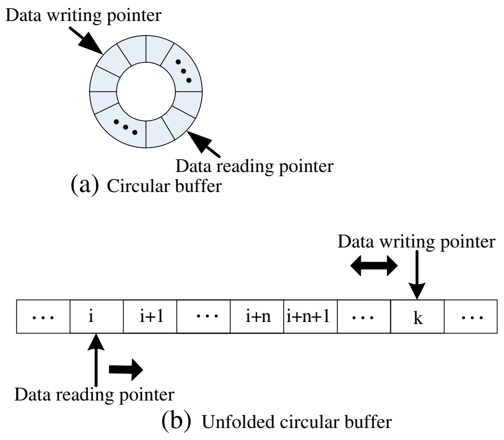

Fig. 12 The circular buffer

other cases is similar to that of the $v_e > v_s$, and it is not illustrated in detail again.

3.4.4 Adaptive low feedrate transition

In general, the theoretical deceleration point cannot overlap with the actual deceleration point, whose position will be slightly ahead (see Fig. 3a, b). Therefore, the destination will not be reached when the feedrate decelerates to the $V_{\min}$ because of the deceleration point error. In other words, the parameter value on the curve will be different than that of the original saved feedrate-sensitive point, and it may be slightly smaller than the original parameter value.

In order to control the interpolation accuracy and ensure that the chord error is constrained under the

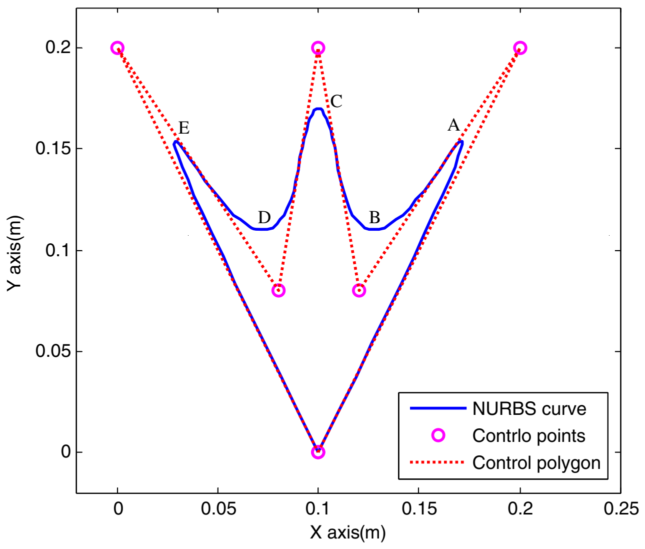

Fig. 13 NURBS curve

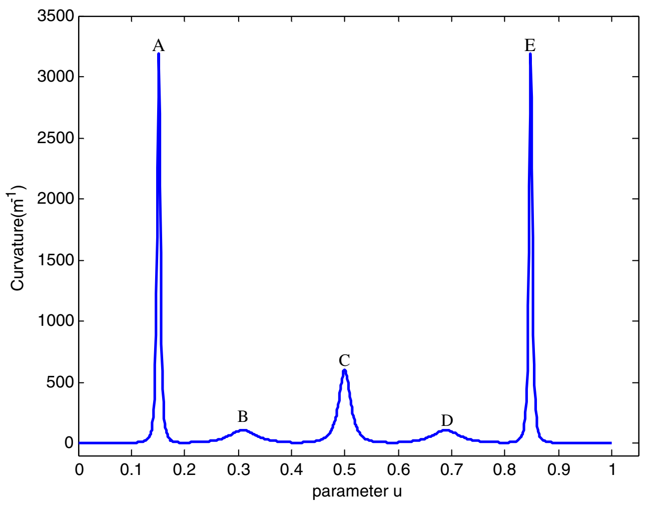

Fig. 14 NURBS curve curvature

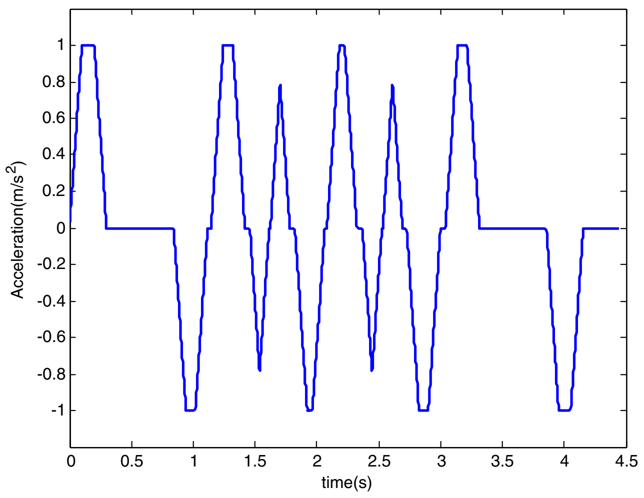

Fig. 16 The acceleration profile

prescribed tolerance, a low adaptive feedrate transition curve is added to the algorithm. When the feedrate decelerates to the $V_{\min}$, the current parameter value and that of the original saved feedrate-sensitive point are compared to determine whether this low feedrate transition curve is required or not.

3.5 The circular buffer

Based on the above reasons, a large memory region is opened up as the circular buffer for storing the parameters of interpolation. The circular buffer's structure is shown in Fig. 12a and its unfolded structure is shown in Fig. 12b.

The circular buffer has two pointers, the data writing pointer and the data reading pointer. The data writing pointer is used to write data obtained from the parametric interpolator into the circular buffer, and it is a two-way pointer, which means that the pointer can be either upwards or downwards. The data reading pointer is used to continuously read data from the circular buffer from the position interpolator, and the reading pointer is a single direction pointer which can only increase. The data writing pointer may need to be traced back from time to time. Because reading and writing pointers are far from each other, and the speed of writing data to the circular buffer is faster than the speed of reading data from it, the reading pointer is not affected when the writing pointer tracing back to update the interpolation parameter.

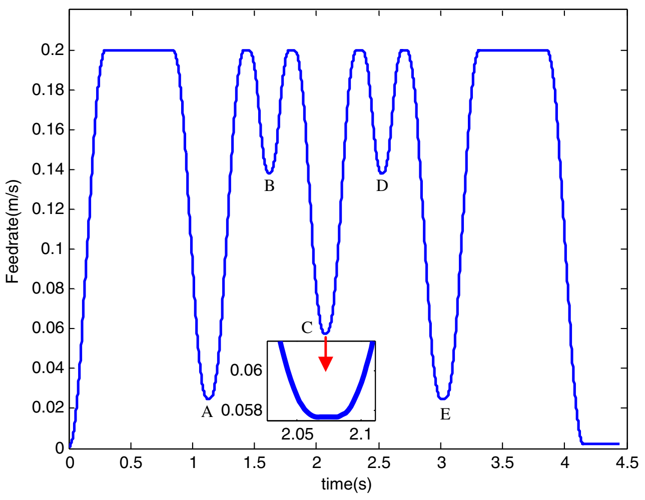

Fig. 15 The feedrate profile

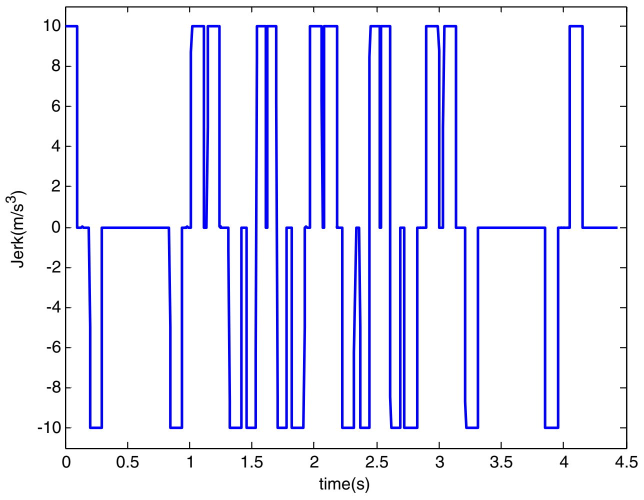

Fig. 17 The jerk profile

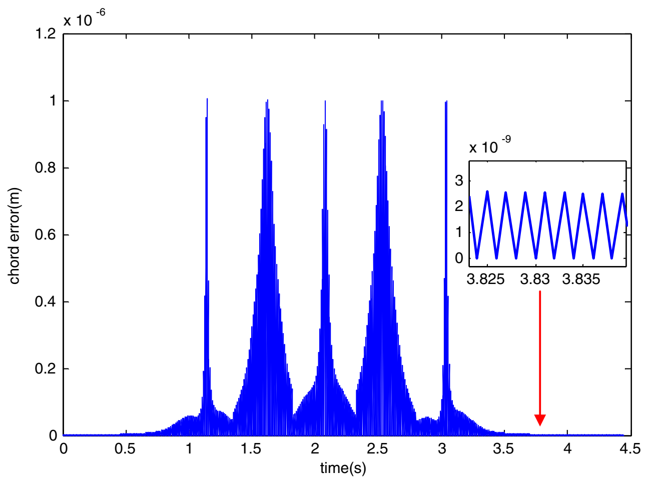

Fig. 18 Chord error profile

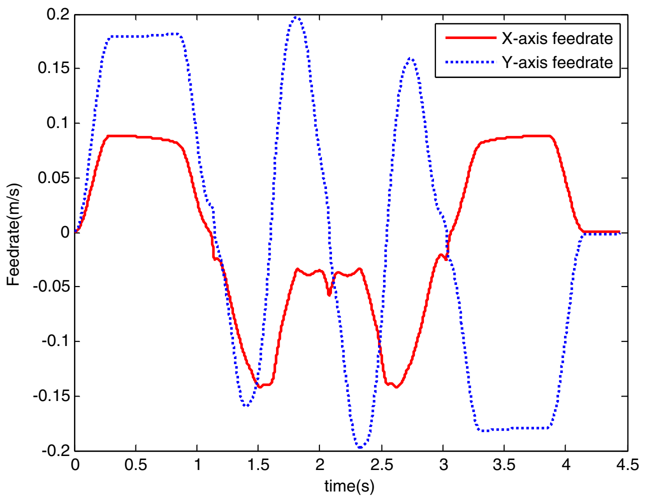

Fig. 20 The *X*- and *Y*-axis feedrate

3.6 NURBS position interpolator

The NURBS position interpolator is the interpolation process immediately after the NURBS parametric interpolator. The input of the position interpolator is the u parameters obtained from the parametric interpolator, and the output of the position interpolator is the position command of the position loop of the motion control module. The parametric interpolator and position interpolator are bridged through the circular buffer. The algorithm schedules it in the ISR, so it takes less time in the position interpolator. The position interpolation algorithm is given as follows:

Step 1: The ISR will not start until half of the data has been written into the circular buffer, or until the parametric interpolator process has been finished.

Step 2: The parametric interpolator and the position interpolator are processed at the same time. In the course of the parametric interpolator process, if the circular buffer is full, the parametric interpolator process will be suspended. The parametric interpolator process will start again once the circular buffer is not full.

Step 3: Whenever an ISR interrupt takes place, a parameter u_i in the circular buffer will be read. Then the reading pointer is advanced by one.

Step 4: According to the read parameters u_i, the position command of each axis is obtained by substituting u_i into the Eq. (1). This process will not stop until the circular buffer is empty, and then the position interpolator is finished.

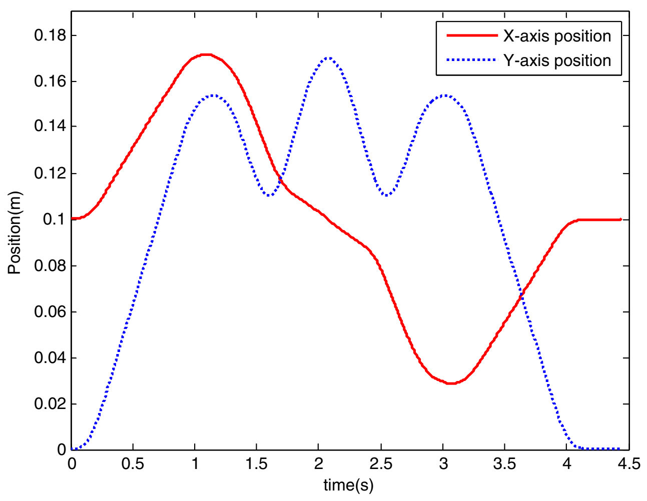

Fig. 19 The *X*- and *Y*-axis position

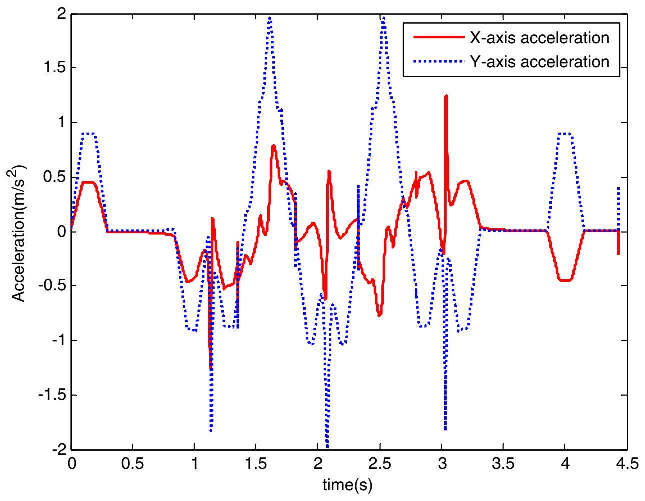

Fig. 21 The *X*- and *Y*-axis acceleration

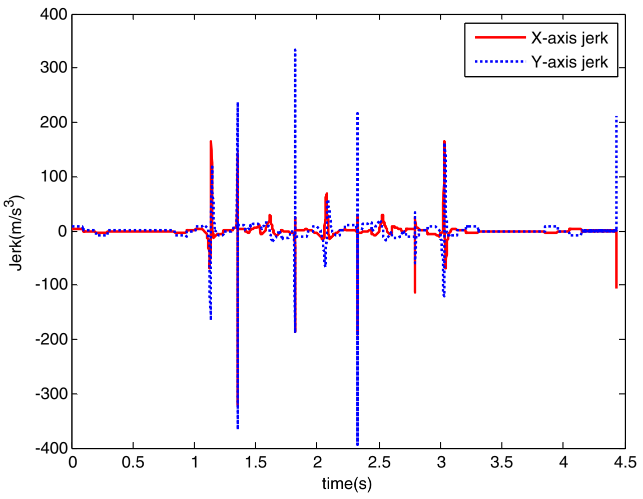

Fig. 22 The *X*- and *Y*-axis jerk

4 Case study and experiment

To evaluate the proposed interpolation algorithm, one trident NURBS curve is chosen for case study. The interpolation algorithm is simulated on a desktop computer with Intel Pentium CPU using MATLAB software.

4.1 NURBS curve

The trident curve (degree 2) is shown in Fig. 13.

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

The control points:{0.1,0,0}, {0.2,0.2,0}, {0.12,0.08,0}, {0.1,0.2,0}, {0.08,0.08,0}, {0,0.2,0}, {0.1,0,0} (meters);

The knot vector is U={0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1};

The weight vector is W={1, 1, 1, 1, 1, 1, 1};

The sampling time is T_s=0.002 s;

The command feedrate is F_c=0.2 m/s;

The maximal allowable chord error is $\delta_{\max}=1\times10^{-6}$ m;

The maximal allowable Acc/Dec is a_m=1 m/s^2;

The maximal allowable jerk is j_m=10 m/s^3;

The curvature of the trident curve is shown in Fig. 14, and the English alphabet marks the feedrate-sensitive point in sequence.

4.2 Simulation result of the trident curve

In this section, numerical simulations are performed using MATLAB software. There are 2,220 interpolation steps in the machining process, and the consumed time is 4.44 s. In addition, the parametric interpolator time taken is about 2.53 s, and position interpolator is about 31.06 ms. It can be seen that the position interpolator is much faster than the parametric interpolator and the average time of the position interpolator is less than 15 μs. Because the interpolation (sampling) period is 2 ms, it is appropriate that the position interpolator is scheduled in ISR.

The resultant feedrates, acceleration, and jerk are shown in Figs. 15, 16, and 17, respectively. From Figs. 14 and 15, it can be seen that the proposed interpolation algorithm does well on those high curvature regions; that is, it decreases feedrate to improve contour precision. By magnifying the local portion of feedrate profile in Fig. 15, it can be seen clearly that the feedrate profile is very smooth near the feedrate-sensitive point and there is a low feedrate transition curve. From Fig. 16, it can be seen that the resultant acceleration profile is trapezoidal or triangular profile. Figure 17 also shows that the variation of jerk falls in the range of CNC system parameters. As shown in Fig. 18, the chord error is constrained under the prescribed 1-μm tolerance at every interpolation point. Finally, the *X*- and *Y*-axis position, feedrate, acceleration, and jerk profile are shown in Figs. 19, 20, 21, and 22, respectively.

From Figs. 16 and 17, it can be seen that the magnitudes of the acceleration and jerk have been reduced significantly after the proposed interpolation algorithm is adopted. In addition, it is also not necessary that the proposed interpolation algorithm is done off-line. Meanwhile, the proposed interpolation algorithm is fit not only for the long spline interpolation, but also for the short spline interpolation. The simulation results show that the proposed NURBS algorithm is feasible for NURBS interpolation in terms of high-precision machining.

5 Conclusions

The key issue of high-speed and high-precision CNC machining of NURBS curve interpolation is the turning velocity improvement at the feedrate-sensitive point, i.e., high curvature point. The interpolation algorithm needs to have a good Acc/Dec control scheme and can output the feedrate profile as smooth as possible while keeping the computational time down.

In this paper, the proposed interpolation algorithm not only considers chord errors, feedrate fluctuations, jerk-limited, and Acc/Dec capabilities of the machine, but also optimizes the look-ahead process. In the meanwhile, it improves machining efficiency by adding the circular buffer and pre-interpolation (non-off-line), and enhances the real-time performance by removing the time-consuming calculation from the ISR. Furthermore, the proposed interpolation algorithm can interpolate both the long spline and the short spline with uniform method. Finally, a case study has been conducted to evaluate the proposed interpolation algorithm, and the advantages of the method are confirmed by simulation results.

Acknowledgments Financial supports from the National Nature Science Foundation of China (no. 51105154) are gratefully acknowledged.

References

1. Shipitalni M, Koren Y, Lo CC (1994) Real-time curve interpolators. Computer-Aided Design 26(11):832–838
2. Yang DCH, Kong T (1994) Parametric interpolator versus linear interpolator for precision CNC machining. Computer-Aided Design 26(3):225–234
3. Piegl L, Tiller W (1997) The NURBS Books, 2nd edn. Springer, Berlin
4. Jung HB, Kim K (2000) A new parameterisation method for NURBS surface interpolation. Int J Adv Manuf Technol 16:784–790
5. Cheng MY, Tsai MC, Kuo JC (2002) Real-time NURBS command generators for CNC servo controllers. Int J Mach Tool Manuf 42:801–813
6. Tsai MC, Cheng CW (2003) A real-time predictor–corrector interpolator for CNC machining. ASME Trans J Manuf Sci Eng 125 (3):449–60
7. Erkorkmaz K, Altintas Y (2005) Quintic spline interpolation with minimal feed fluctuation. ASME Trans J Manuf Sci Eng 127 (2):339–49
8. Lei WT, Sung MP, Lin LY, Huang JJ (2007) Fast real-time NURBS path interpolation for CNC machine tools. Int J Mach Tool Manuf 47:1530–1541
9. Zhiming X, Jincheng C, Zhengjin F (2002) Performance evaluation of a real-time interpolation algorithm for NURBS curves. Int J Adv Manuf Technol 20:270–276
10. Yeh SS, Hsu PL (2002) Adaptive-feedrate interpolation for parametric curves with a confined chord error. Comput-Aided Des 34:229–237
11. Tikhon M, Ko Tae Jo, Lee SH, Kim HS (2004) NURBS interpolator for constant material removal rate in open NC machine tools. Int J Mach Tool Manuf 44:237–245
12. Luo FY, Zhou YF, Yin J (2007) A universal velocity profile generation approach for high-speed machining of small line segments with look-ahead. Int J Adv Manuf Technol 35:505–518
13. Tsai MS, Nien HW, Yau HT (2011) Development of integrated acceleration/deceleration look-ahead interpolation technique for multi-blocks NURBS curves. Int J Adv Manuf Technol 56:601–618
14. Du DS, Liu YD, Yan CL, Li CX (2007) An accurate adaptive parametric curve interpolator for NURBS curve interpolation. Int J Adv Manuf Technol 32:999–1008
15. Nam SH, Yang MY (2004) A study on a generalized parametric interpolator with real-time jerk-limited acceleration. Comput-Aided Des 36:27–36
16. Park JH, Nam SH, Yang MY (2005) Development of a real-time trajectory generator for NURBS interpolation based on the two-stage interpolation method. Int J Adv Manuf Technol 26:359–365
17. Liu XB, Ahmad F, Yamazaki K, Mori M (2005) Adaptive interpolation scheme for NURBS curves with the integration of machining dynamics. Int J Mach Tool Manuf 45:433–444
18. Lin MT, Tsai MS, Yau HT (2007) Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm. Int J Mach Tool Manuf 47:2246–2262
19. Sekar M, Narayanan VN, Yang SH (2008) Design of jerk bounded feedrate with ripple effect for adaptive NURBS interpolator. Int J Adv Manuf Technol 37:545–552
20. Shen HY, Fu JZ, Fan YQ (2011) A new adaptive interpolation scheme of NURBS based on axis dynamics. Int J Adv Manuf Technol 56:215–221
21. Du DS, Liu YD, Guo XG, Yamazaki K, Fujishima M (2010) An accurate adaptive NURBS curve interpolator with real-time flexible acceleration/deceleration control. Robot Comput-Integrated Manuf 26:273–281
22. Yong T, Narayanaswami R (2003) A parametric interpolator with confined chord errors, acceleration and deceleration for NC machining. Comput-Aided Des 35:1249–1259