Mathematics Mechanization Research Preprints
KLMM, Chinese Academy of Sciences
Vol. 29, 165–188, September 2010

Time-Optimal Interpolation of CNC Machines along Parametric Path with Chord Error and Tangential Acceleration Bounds

Chun-Ming Yuan, Xiao-Shan Gao
KLMM, Institute of Systems Science, Chinese Academy of Sciences
E-mail: (cmyuan,xgao)@mmrc.iss.ac.cn.

Abstract. Interpolation and velocity planning for parametric curves are crucial problems in CNC machining. In this paper, a time-optimal velocity planning algorithm under a given chord error bound and a tangential acceleration bound is proposed. The key idea is to reduce the chord error bound to a centripetal acceleration bound which leads to a velocity limit curve, called the chord error velocity limit curve (CEVLC). Then, the velocity planning is to find the time-optimal velocity curve governed by the tangential acceleration bound "under" the CEVLC. For two types of simple splines, explicit formulas for the optimal velocity curve are given. We implement the methods in these two cases and use curved segments from real machining parts to show the feasibility of the methods.

Keywords. Parametric curve interpolation, time-optimal velocity planning, chord error, velocity limit curve.

1 introduction

In modern CAD systems, the standard representation for free form surfaces is parametric functions. While the conventional computed numerically controlled (CNC) systems mainly use micro line segments (G01 codes) to represent the machining path. To convert the parametric curves into line segments is not only time consuming but also leads to problems such as large data storage, speed fluctuation, and poor machining accuracy. In the seminal work [5, 18, 22], Chou, Yang, and Shpitalni et al proposed to use parametric curves generated in CAD/CAM systems directly in the CNC systems to overcome these drawbacks.

An interpolation algorithm usually consists of two phases: velocity planning and parameter computation. Let $C(u)$ be the manufacturing path. The phase to determine the feed-rate $v(u)$ along $C(u)$ is called velocity planning. When the feed-rate $v(u)$ is known, the way to compute the next interpolation point at $u_{i+1} = u_i + \triangle u$ during one sampling time period is called parameter computation.

This paper focuses on velocity planning along a spatial parametric path. For the parameter computation phase, please refer to [4, 9, 21] and the literatures in them. Two types of acceleration modes can be used in the velocity planning: the tangential acceleration and the multi-axis acceleration where each axis accelerates independently.

Due to its conceptual simplicity, the tangential acceleration is widely used in CNC interpolations. Bedi et al[1] and Yang-Kong[22] used a uniform parametric feed-rate without considering the chord error. Yeh and Hsu used a chord error bound to control the feed-rate if needed and used a constant feedarte in other places [23]. However, the machine acceleration capabilities were not considered. Narayanaswami and Yong [24] gave a velocity planning method based on tangential acceleration bounds by computing maximal passing velocities at sensitive corners. Cheng and Tsai [4] gave an interpolation method based on different velocity profiles for tangential ACC/DCC feed-rate planning. Velocity planning methods with tangential acceleration and jerk bounds were considered by several research groups [7, 11, 13, 14, 12]. In particular, Emami and Arezoo [6], Lai et al[10] proposed velocity planning methods with confined acceleration, jerk, and chord error constraints by checking these values at every sampling point and using backtracking to adjust the velocity if any of the bounds is violated. In all the above methods besides [6, 10, 23], the chord error is considered only at some special points, and there is no guarantee that the chord error is satisfied at all points. Furthermore, all these methods are not proved to be time-optimal.

In the multi-axis acceleration case, Borow [3] and Shiller et al [16, 15] presented a time-optimal velocity planning method for a robot moving along a curved path with acceleration bounds for each axis. Farouki and Timar [19, 20] proposed a time-optimal velocity planning algorithm in CNC machining under the same acceleration constraints. Following the method in [19, 20], Zhang et at gave a simplified time-optimal velocity planning method for quadratic B-splines and realized real-time manufacturing on industrial CNC machines [26]. Zhang et al[25] gave a greedy algorithm for velocity planning under multi-axis jerk constraints. These optimal methods use the “Bang-Bang” control strategy, that is, at least one of the axes reaches its acceleration bound all the time. But they do not consider the chord error which is an important factor for CNC-machining.

In this paper, we give a time-optimal velocity planning algorithm along a parametric curve path $C(u)$ under a given chord error bound and a tangential acceleration bound. The main contribution of the paper is the introduction of the chord error velocity limit curve (abbr. CEVLC) and the velocity planning algorithm.

We show that the chord error bound can be approximately reduced to a centripetal acceleration bound. Furthermore, if the centripetal acceleration reaches its maximal bound, the velocity can be written as an algebraic function in the parameter u of the curve path $C(u)$. The graph of this function is called the CEVLC. The CEVLC is significant because the final velocity curve must be “under” this curve or be part of this curve, which narrows the range of velocity planning. Also, certain key points on the CEVLC, such as the discontinuous points, play an important role in the velocity planning.

Similar to the previous work on time-optimal velocity planning, our algorithm also uses the “Bang-Bang” control strategy, which is a necessary condition for time optimal in our case. Then the final velocity curve is governed either by the centripetal acceleration bound or by the tangential acceleration bound, and the later one is called the integration velocity curve. The main task of the velocity planning is to find the switching points between the integration velocity curve and the CELVC.

Two main results on the optimal velocity curve are given in this paper. Firstly, as a theoretical result, we show that the final velocity is the minimum of the CEVLC and all the integration velocity curves passing through the initial point, the termination point, and the key points of the CEVLC. We also give a practical algorithm to compute the optimal velocity curve incrementally. For quadratic splines and cubic PH curves, the integration velocity curve can be given by explicit formulas. We implement our algorithm in these two cases and conduct experiments using curved pathes from real manufacturing parts.

Moreover, we also give a simple and real-time feed-rate override algorithm based on our velocity planning curve. For multi-axis velocity planning algorithms, feed-rate override is complicated and not suitable for real time interpolation.

As a final remark, we want to mention that by combining the CEVLC proposed in this paper and the method proposed in [19, 20], it is possible to give a time optimal velocity planning method under a given chord error bound and multi-axis acceleration bounds.

The rest of the paper will be organized as follows. Section 2 gives the time-optimal velocity planning algorithm for a parametric curve. Section 3 gives the details for computing the time optimal velocity curves for quadratic B-spline and cubic PH-spline. Section 4 gives a feed-rate override algorithm for CNC-machining. Section 5 concludes the paper.

2 Time-optimal velocity planning under chord error and tangential acceleration bound

The time-optimal velocity planning algorithm will be presented in this section. We consider a spatial piecewise parametric curve $C(u), u = 0..1$ with C^1 continuity, such as splines, Nurbs, etc. We further assume that each piece of the curve is differentiable to the third order and has left and right limitations at the endpoints.

2.1 Problem

In order to control the machine tools, we need to know the velocity of the movement at each point on the curve, which is denoted as a function $v(u)$ in the parameter u and is called the *velocity curve*. The procedure to compute the velocity curve $v(u)$ is called *velocity planning*. In this subsection, we will show that the chord error bound can be reduced to the centripetal acceleration bound and formulate the velocity planning problem as an optimization problem with tangential and centripetal acceleration bounds.

For a parametric curve $C(u)$, we denote its *parametric speed* to be:

$$\sigma(u) = \frac{ds}{du} = |C'(u)|,$$

where $'$ is the derivative w.r.t. u. The *curvature* and *radius of curvature* are defined to be:

$$k(u) = \frac{|C'(u) \times C''(u)|}{\sigma(u)^3}, \rho(u) = \frac{1}{k(u)}.$$

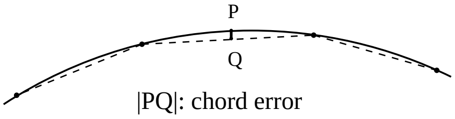


Fig. 1. The chord error

The cutter moves in a line segment between two adjacent points on the curve $C(u)$ and the distance between the line segment and the curve segment is called the *chord error* which is the main source of the manufacturing error (Fig 1). Firstly, we will show that the chord error bound can be reduced to the centripetal acceleration bound.

Let T be the sampling period, δ the error constraint, and $\pm A$ the tangential acceleration bounds. In practice, the chord error bound is much less than the radius of curvature at each point on the curve. By the chord error formula [23], at each parametric value u,

$$q(u) = v^2(u) \le \frac{8\delta\rho(u) - 4\delta^2}{T^2} \approx \frac{8\delta\rho(u)}{T^2} = \frac{8\delta}{k(u)T^2}.$$

If we denote by $d(u)$ the interpolating chord error at u with feed-rate $v(u)$, then $d(u) = \frac{T^2 v(u)^2}{8\rho(u)} \le \delta$. Let

$$a_N(u) = v(u)^2/\rho(u) = k(u)v(u)^2 = k(u)q(u) \tag{1}$$

be the centripetal acceleration and

$$B = \frac{8\delta}{T^2}. \tag{2}$$

Then, the chord error bound is transformed to the centripetal acceleration bound:

$$d(u) \le \delta \Longleftrightarrow a_N(u) \le B.$$

Since

$$\frac{d}{dt} = \frac{ds}{dt}\frac{du}{ds}\frac{d}{du} = \frac{v}{\sigma}\frac{d}{du}, \tag{3}$$

The tangential acceleration is:

$$a_T(u) = \frac{dv(u)}{dt} = \frac{v(u)v'(u)}{\sigma(u)} = \frac{q'(u)}{2\sigma(u)}. \tag{4}$$

From (3), the time optimal velocity planning problem becomes the following optimization problem: to find a velocity curve $v(u)$ such that

$$\min_{v(u)} t = \int_0^1 \frac{\sigma(u)}{v(u)} du, \tag{5}$$

with the tangential and centripetal acceleration constraints

$$|a_T(u)| \le A, a_N(u) \le B, u = 0..1, \tag{6}$$

where $(a_T(u), a_N(u))$ are the tangential and centripetal accelerations and (A, B) are their bounds respectively. Note that B can be derived from the chord error bound from (2).

2.2 The main result

In this section, we present the main result of the paper about the time optimal velocity planning problem (5). We will give a velocity planning algorithm in section 2.4 and prove the correctness of the result in section 2.5.

Before giving the main result, we need to introduce several basic concepts. We use "Bang-Bang" control, that is, at least one of "=" holds among the inequalities in (6). This will lead to two cases.

If the centripetal acceleration reaches its bound B, then from (1) we have $q(u) = \frac{B}{k(u)}$, which defines a curve

$$\begin{aligned} &q_{lim}(u) = v_{lim}^2(u) = B/k(u), u \in [0,1] \text{ or} \\ &v_{lim}(u) = \sqrt{B/k(u)}, u \in [0,1], \end{aligned} \tag{7}$$

in the u-v plane with v as the vertical axis. We call this curve the *velocity limit curve with the chord error constraint*, denoted by CEVLC. If $C(u)$ is a piecewise curve with C^1 continuity, then its CEVLC is the combination of the CEVLCs of its components. From the constraint $a_N \leq B$, the velocity curve must be "under" the CEVLC in the v direction.

Example 2.1 If $C(u) = (\frac{2u}{1+u^2}, \frac{1-u^2}{1+u^2}, 0)$ and $B = 100$, then $v_{lim}(u) = 10, u \in [0,1]$.

If $C(u) = (u^2+u+1, u^2+2u-1, 2u+1)$ and $B = 600$, then $v_{lim}(u) = 10(8u^2+12u+9)^{\frac{3}{4}}, u \in [0,1]$.

If $C(u) = (\frac{1}{3}u^3 - 4u^2 - 8u, \frac{2}{3}u^3 + 3u^2 - 8u, -\frac{2}{3}u^3 - 2u^2 - 14u)$ and $B = 1000$, then $v_{lim}(u) = 30u^2 + 40u + 180, u \in [0,1]$.

If the tangential acceleration reaches its bound, then from (4) the square of the velocity $q(u) = v(u)^2$ can be obtained by solving the differential equation $q'(u) = \pm 2A\sigma(u)$, whose solution is

$$q = \pm 2A\pi(u) + c, \tag{8}$$

where $\pi(u) = \int \sigma du$ is the primitive function of $\sigma(u)$ and c a constant which can be determined by an initial point $(u^*, q(u^*))$:

$$c = q(u^*) \mp 2A\pi(u^*).$$

The curve (8) is called the *integration curve* or the *integration trajectory.*

Let $v_P(u)$ be the integration trajectory with tangential acceleration A starting from an initial point $P = (u_P, v_{lim}(u_P))$ both for the forward (the $+u$) and the backward (the $-u$) directions. Let $v_0(u)$ be the forward integration trajectory with tangential acceleration A from the initial point $P_0 = (0,0)$ and $v_1(u)$ the backward integration trajectory with tangential acceleration A from the initial point $P_1 = (1,0)$. Of curse, all the curves mentioned above are defined in $[0,1]$.

Then, the solution to the optimization problem (5) is given below.

Theorem 2.2 *Let $\mathcal{K}$ be the finite set of key points of the CEVLC to be defined in the next section. Then*

$$v(u) = \min_{P \in \mathcal{K}}(v_{lim}(u), v_P(u), v_0(u), v_1(u)). \tag{9}$$

Then, $v(u)$ is the solution to the optimal problem (5).

Obviously, $v(u)$ is a piecewise continuous curve for $u \in [0,1]$. We will prove the theorem in Section 2.5.

2.3 Key points of the CEVLC

There exist three types of special points on the CEVLC, which are called *switching points* or *key points*.

The first type switching points are the discontinuous points of the CEVLC. These points correspond to the curvature discontinuous points of the original curve. They must be the singular points of the parametric curve or the connection points of two curve segments. If we denote by $C(u)$ the curve segment, then the singular points of $C(u)$ can be computed by solving the equations $C'(u) \times C''(u) = 0$. From (7), the limit velocity v_{lim} is ∞ at these points and the real velocity curve can not reach it, so we can omit the singular points from the key points. Thus, only the connection points need to be considered.

At a first type key point, the velocity is not continuous and we call the smaller one of the left and right side velocities as the *minimal velocity* or simply the *velocity* of this point.

The second type switching points are the continuous but non-differentiable points of the CEVLC (slope discontinuity). These points correspond to the curvature continuous but non-differentiable points of the original curve. Hence, they must be the connection points of two curve segments.

For a differentiable segment of the CEVLC divided by the above two types of key points, we can divide it according to whether the tangential acceleration of the CEVLC is $\pm A$, where A is from (6). A point on the CEVLC is called a key point of the third type if the tangential acceleration along the CEVLC at this point is $\pm A$. We can find the third type key points by solving the following algebraic equation in u

$$a_T(u) = \frac{q_{lim}(u)}{2\sigma(u)} = \frac{B\sigma(u)^2}{2|C'(u) \times C''(u)|} = \pm A.$$

With these switching points, the CEVLC is divided into two types of segments:

1. A curve segment is called *feasible* if the absolution values of tangential acceleration at all points are bounded by A. A feasible CEVLC segment can be a part of the final velocity curve.
2. A curve segment is called *unfeasible* if the absolution values of tangential acceleration at all points are larger than A. An unfeasible CEVLC segment cannot be a part of the final velocity curve. In other words, the final velocity curve must be strictly under it.

If the curve segments on the left and right sides of a second type key point are both feasible, we delete this key point.

2.4 The time optimal velocity planning algorithm

We use the "Bang-Bang" control strategy. Then the real velocity curve is governed either by the maximal centripetal acceleration or by the maximal tangential acceleration. Since the velocity curve must be under or be a part of the CEVLC, we need to find the proper integration trajectory under the CEVLC. We first give the main idea of the velocity planning algorithm.

Firstly, we compute the CEVLC, find the key points and their speeds. Compute the forward integration trajectory v_s from the starting point $(0,0)$ with tangential acceleration A. Find the intersection point $(u_l, v_s(u_l))$ of v_s and the CEVLC. Compute the backward integration trajectory v_e from the ending point $(1,0)$ with tangential acceleration A. Find the intersection point $(u_r, v_e(u_r))$ of v_e and the CEVLC.

Secondly, if $(u_l, v_s(u_l)) = (u_r, v_e(u_r))$, then return the combination of v_s and v_e as the final velocity curve. If $u_l > u_r$, find the intersection point of v_s and v_e and return the combination of v_s and v_e as the final velocity curve. Otherwise, set $P_c = (u_l, v_s(u_l))$ to be the current point and consider the following three cases.

If the next segment of CEVLC from point P_c in the forward direction (the $+u$ direction) is feasible, then we merge this feasible segment into v_s and set the current point $(u_l, v_s(u_l))$ to be the end point of this segment.

If the next segment of CEVLC from point P_c in the forward direction is not feasible and the forward integration trajectory v_f with initial point P_c is under the CEVLC, then let $(u_i, v_f(u_i))$ be the intersection point of v_f and the CEVLC. Merge $v_f(u), u \in [u_r, u_i]$ into v_s. Let $u_l = u_i$ and $P_c = (u_l, v_f(u_l))$.

If the next segment of CEVLC from point P_c in the forward direction is not feasible and the integration trajectory v_f with initial point P_c is above the CEVLC, then we find the next key point P_n on the right hand side of P_c. From P_n, compute the backward integration trajectory v_b, find the intersection point of v_b with v_s, and merge v_b and v_s as the new v_s. Set P_n as the new current point.

Finally, with the new current point, we can repeat the procedure from the second step until the velocity curve is found.

Here by saying a forward curve $v_1(u)$ is under or above another curve $v_2(u)$ from a point $P = v_1(u*)$, we mean that $v_1(u)$ is under or above $v_2(u)$ in a neighborhood of $u*$ in the forward direction.

To describe our algorithm precisely, we need the following notations. For a discontinuous curve $f_1(x)$, we denote by $f_1^+(x^*)$ and $f_1^-(x^*)$ the limitations of $f_1(x)$ at x^* from the left and right hand sides respectively, and define $f_1(x^*) = \min(f_1^+(x^*), f_1^-(x^*))$. Let $f_2(x)$ be a curve with C^0 continuity. If $f_2(x^*) = f_1(x^*)$ or $f_2(x^*)$ is between the left and right limitations of $f_1(x)$ at x^*, then define $(x^*, f_2(x^*))$ to be the intersect point of the curves $(x, f_2(x))$ and $(x, f_1(x))$.

We now give the velocity planning algorithm.

Algorithm VP_CETA. The input of the algorithm is the curve $C(u), u \in [0,1]$, a chord error bound d, and a tangential acceleration bound A. The output is the velocity curve $v_c(u), u \in [0,1]$ which is the solution to the optimization problem (5).

1 Compute the CEVLC in (7) and its key points as shown in section 2.3. Denote by U_f the parametric intervals where the corresponding segments of the CEVLC are feasible.

2 From the starting point $(0,0)$, compute the forward integration trajectory v_s using the tangential acceleration A. Compute the first intersection point $(u_l, v_s(u_l))$ of v_s and the CEVLC. If there exists no intersection, denote $u_l = 1$.

3 From the ending point $(1, 0)$, compute the backward integration trajectory v_e using the tangential acceleration A. Compute the first intersection point $(u_r, v_e(u_r))$ of v_e and the CEVLC. If there exists no intersection, denote $u_r = 0$.

4 If $(u_l, v_s(u_l)) = (u_r, v_e(u_r))$, then return the combination of v_s and v_e as the final velocity curve. If $u_l > u_r$, find the intersection point $(u_i, v_s(u_i))$ of v_s and v_e, return $v(u)$, where

$$v(u) = \begin{cases} v_s, 0 \le u \le u_i \\ v_e, u_i < u \le 1. \end{cases} \tag{10}$$

5 If $(u_l, v_{lim}(u_l))$ is a discontinuous point on the CEVLC, there are three possibilities.

(a1) If $v^+_{lim}(u_l) > v_s(u_l) = v^-_{lim}(u_l)$, goto step 6.

(a2) If $v^+_{lim}(u_l) = v_s(u_l) < v^-_{lim}(u_l)$, goto step 8.

(a3) If $v^+_{lim}(u_l) \ge v_s(u_l) > v^-_{lim}(u_l)$, let $u_n = u_l$ and goto step 9.

6 Now, $(u_l, v_s(u_l))$ is the starting point of the next segment of CEVLC. There exist three cases.

(b1) If the next segment of the CEVLC is feasible, goto step 7

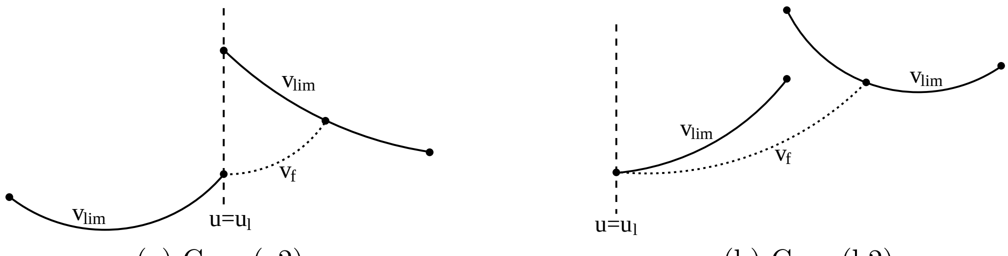
(b2) If $a^-_T(u_l) \ge A$, goto step 8.

(b3) If $a^-_T(u_l) \le -A$, then find the next key point $(u_n, v_{lim}(u_n))$ along the $+u$ direction and goto step 9.

7 Let $u_n > u_l$ be the parameter of the next key point. Then, $(u_l, u_n) \subset U_f$. Update v_s to be

$$v_s(u) = \begin{cases} v_s(u), & 0 \le u < u_l \\ v_{lim}(u), & u_l \le u \le u_n. \end{cases}$$

Let $u_l = u_n$, goto step 4.



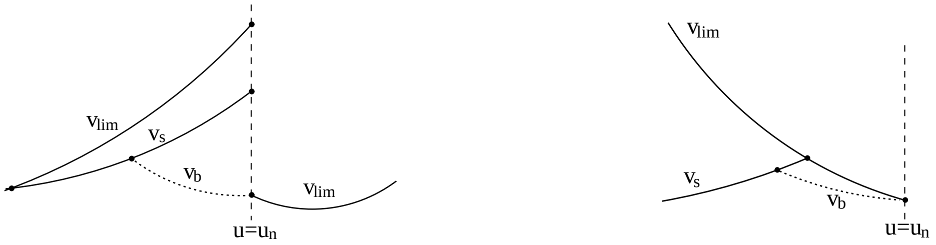
(a) Case (a2)

(b) Case (b2)

Fig. 2. Two cases of step 8: computation of forward integration curve

8 Starting from $(u_l, v_s(u_l))$, compute the forward integration trajectory v_f with tangential acceleration A. Find the first intersection point $(u_i, v_f(u_i))$ of v_f and the CEVLC (Fig. 2). If there exists no intersection, denote $u_l = 1$. Update v_s to be

$$v_s(u) = \begin{cases} v_s(u), & 0 \le u < u_l \\ v_f(u), & u_l \le u \le u_i. \end{cases}$$



(a) Case (a3)

(b) Case (b3)

Fig. 3. Two cases of step 9: computation of backward integration curve

Let $u_l = u_i$, goto step 4.

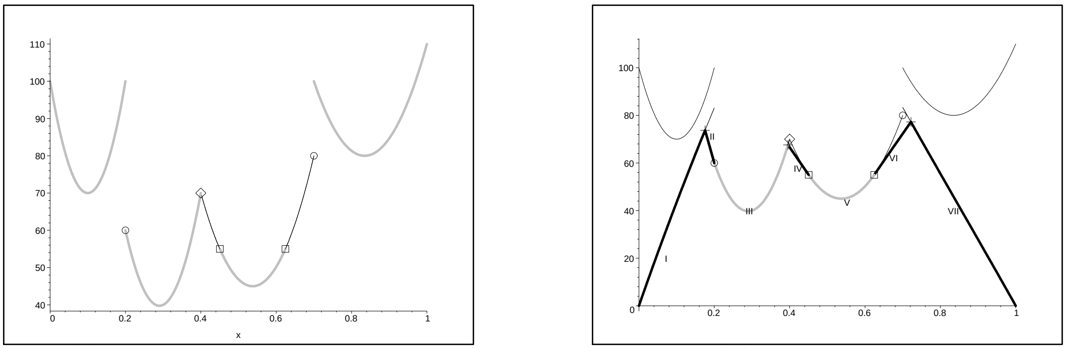
9 Starting from point $(u_n, v_{lim}(u_n))$, compute the backward integration trajectory v_b in the $-u$ direction with tangential acceleration A. Find the intersection point[1] $(u_i, v_b(u_i))$ of v_b and v_s. See Fig. 3 for an illustration. Update v_s to be

$$v_s(u) = \begin{cases} v_s(u), & 0 \leq u < u_i \\ v_b(u), & u_i \leq u \leq u_n. \end{cases}$$

Let $u_l = u_n$, goto step 4.

The correctness proof of the algorithm is given in Section 2.5. Further improvements of the algorithm are given in Section 2.6.

The flow chart of the above algorithm is given in Fig. 5. The details of the algorithm is omitted in the figure.



(a) CEVLC and key points

(b) velocity curve

Fig. 4. An illustrative example of velocity planning

Figure 4 is an illustrative example of the algorithm.

Compute the CEVLC v_{lim} and the key points as shown in Figure 4(a), where $\circ$ represents the key points of first type and the corresponding parameters are $0.2, 0.7$; $\diamond$ represents the key point of second type, and the corresponding parameter is 0.4; and $\square$ represents the key points of third type. The gray parts are feasible segments.

[1] We will show that there exists a unique intersection point.

Starting from point $(u, v) = (0, 0)$, compute the forward integration trajectory v_s which intersects the CEVLC at $u = 0.2$. Starting from $(u, v) = (1, 0)$, compute the backward integration trajectory v_e which intersects the CEVLC at $u = 0.7$.

In step 5, the current point P_c is a discontinues point and case (a3) is executed. In step 9, starting from the first $\circ$ point, compute the backward integration trajectory v_b which intersects v_s at the first point marked by $+$. Update v_s to be the piecewise curve marked by $\mathbf{I}, \mathbf{II}$ in Figure 4(b).

From the first point marked by $\circ$, the CEVLC is feasible. Hence, update v_s to be the piecewise curve marked by $\mathbf{I}, \mathbf{II}$ and the first gray part of the CEVLC($\mathbf{III}$).

Let the key point marked by $\diamond$ be the current point. From the current point, the CEVLC is not feasible and case (b3) is executed. In step 9, we select the next key point which is the first point marked by $\square$. Starting from this point, compute the backward integration trajectory v_b which intersects v_s at the second point marked by $+$. Update v_s to be the piecewise curve marked by $\mathbf{I}, \mathbf{II}, \mathbf{III}, \mathbf{IV}$ in the figure.

Starting from the first point marked by $\square$, the CEVLC is feasible. Hence, update v_s to be the piecewise curve marked by $\mathbf{I}, \mathbf{II}, \mathbf{III}, \mathbf{IV}, \mathbf{V}$ in the figure.

Let the second point marked by $\square$ to be the current point. Starting from this point, the CEVLC is not feasible and case (b2) is executed. In step 8, compute the forward integration trajectory v_f which intersects v_e at the third point marked by $+$. The final velocity curve consists of seven pieces marked by $\mathbf{I}, \mathbf{II}, \mathbf{III}, \mathbf{IV}, \mathbf{V}, \mathbf{VI}$, and $\mathbf{VII}$ in the figure.

Note that the CEVLC can be parts of the final velocity curve quite often, while in [19, 20], the VLC is rarely a part of the final velocity curve.

Remark 2.3 In the above algorithm, we need to compute the CEVLC and its key points, the integration trajectory, the intersection points of the integration trajectory and the CEVLC, and the intersection points of two integration trajectories. In principle, these computations can be reduced to computing integrations and solving algebraic equations. In Section 3, we will show how to give explicit formulas for the integration curve for two types of special types of curves.

2.5 The correctness of the algorithm

In this section, we will show that the velocity curve given by the algorithm in the preceding subsection is the only solution to the optimization problem (5). The proof is divided into two parts. We will first show that the velocity curve computed by the algorithm is the curve defined in (9) and then prove that this velocity curve is the solution to the problem (5). In order to prove this, we need the following result.

Theorem 2.4 *[2][p.24-26] Let y, z be solutions of the following differential equations*

$$y' = F(x, y), z' = G(x, z),$$

respectively, where $F(x, y) \leq G(x, y), a \leq x \leq b$, and F or G satisfies Lipschitz's condition. If $y(a) = z(a)$, then $y(x) \leq z(x)$ for any $x \in [a, b]$.

In our case, the above theorem implies the following result.

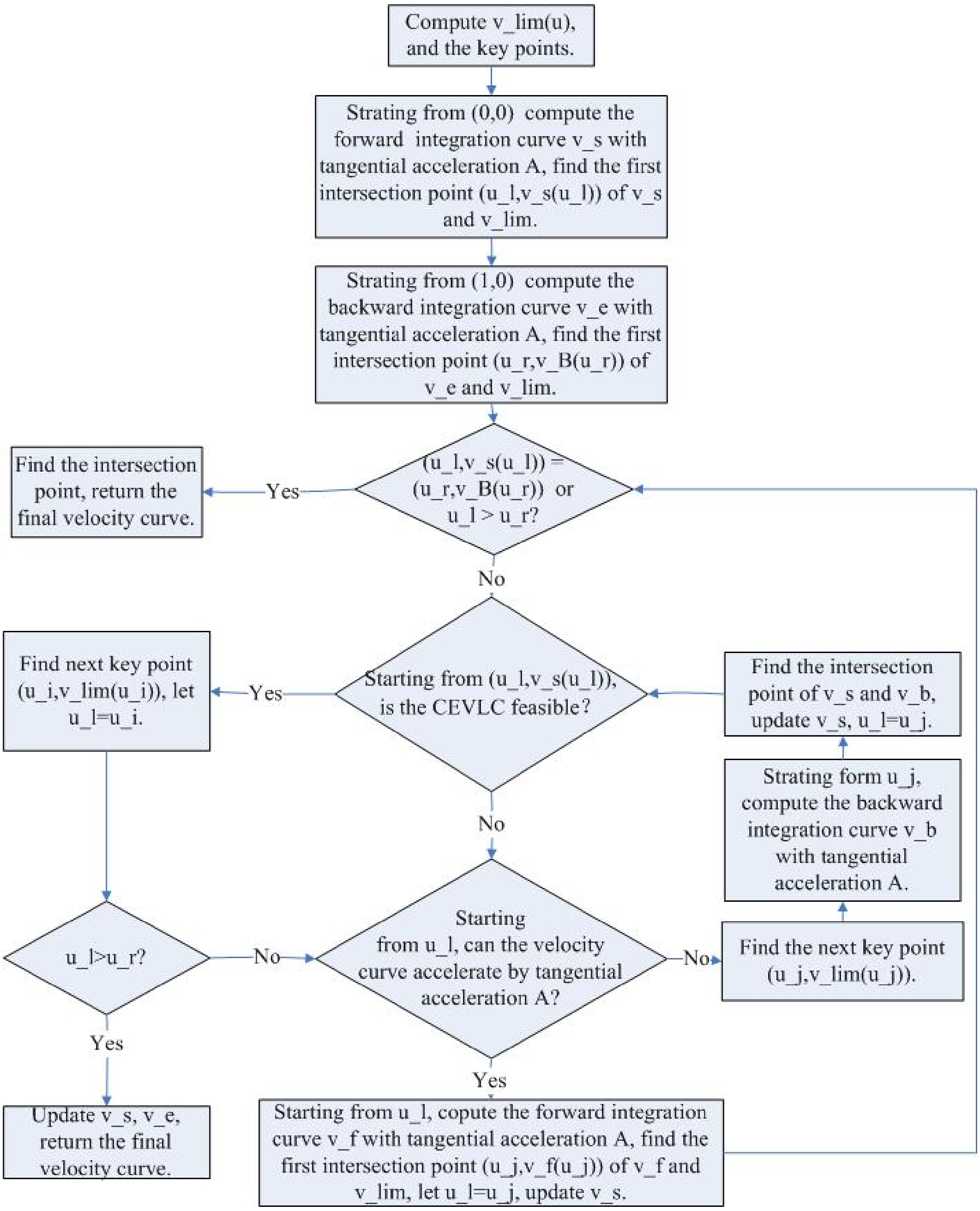

Compute v_lim(u), and the key points.
Strating from (0,0) compute the forward integration curve v_s with tangential acceleration A, find the first intersection point (u_l,v_s(u_l)) of v_s and v_lim.
Strating from (1,0) compute the backward integration curve v_e with tangential acceleration A, find the first intersection point (u_r,v_B(u_r)) of v_e and v_lim.
(u_l,v_s(u_l)) = (u_r,v_B(u_r)) or u_l > u_r?
Yes
Find the intersection point, return the final velocity curve.
No
Starting from (u_l,v_s(u_l)), is the CEVLC feasible?
Yes
Find next key point (u_i,v_lim(u_i)), let u_l=u_i.
Find the intersection point of v_s and v_b, update v_s, u_l=u_j.
Strating form u_j, compute the backward integration curve v_b with tangential acceleration A.
No
Starting from u_l, can the velocity curve accelerate by tangential acceleration A?
No
Find the next key point (u_j,v_lim(u_j)).
u_l>u_r?
No
Yes
Update v_s, v_e, return the final velocity curve.
Yes
Starting from u_l, copute the forward integration curve v_f with tangential acceleration A, find the first intersection point (u_j,v_f(u_j)) of v_f and v_lim, let u_l=u_j, update v_s.

Fig. 5. Flow chart of the velocity planning

Lemma 2.5 *Let $v_1(u)$ and $v_2(u)$ be two velocity curves for the curve $C(u)$ and $[u_1, u_2] \subset [0, 1]$. If $v_1(u_1) \leq v_2(u_1)$ and $a_{T1}(u) \leq a_{T2}(u)$ for $u \in [u_1, u_2]$, then $v_1(u) \leq v_2(u)$ for $u \in [u_1, u_2]$. Furthermore, if $v_1(u_1) < v_2(u_1)$, then $v_1(u) < v_2(u)$ for $u \in [u_1, u_2]$.*

Proof. We may assume that $v_1(u_1) = v_2(u_2)$, since if $v_1(u_1) < v_2(u_1)$, we may consider $\bar{v}_2(u) = v_2(u) - v_2(u_1) + v_1(u_1)$ which still satisfies the conditions in the lemma. Since $C(u)$ is differentiable to the order of three, $\sigma(u)$ and $a_{T1}(u)$ must be bounded in $[u_1, u_2]$. From (4), $q_1'(u) = 2\sigma(u)a_{T2}(u)$ and $q_2'(u) = 2\sigma(u)a_{T2}(u)$. Then, we have $q_1(u_1) = v_1(u_1)^2 = q_2(u_1) = v_2(u_1)^2$, $2\sigma(u)a_{T2}(u) \leq 2\sigma(u)a_{T2}(u)$ for $u \in [u_1, u_2]$, and $2\sigma(u)a_{T1}(u)$ satisfies the Lipschitz's condition. Using Theorem 2.4, we have $v_1(u) \leq v_2(u)$ for $u \in [u_1, u_2]$. The second part of the lemma can be proved similarly. ▮

Before giving the correctness proof, we first explain the key steps of the algorithm. Besides the in initial steps, new velocity trajectories are generated in Steps 7, 8, 9. We will show that these steps are correct in the sense that they will really generate new velocity trajectories. Step 7 is obvious, since we will use the next feasible CEVLC segment as the velocity trajectory. Two cases lead to step 8: cases (a2) and (b2). In case (a2), we have $v_{lim}^+(u_l) = v_s(u_l) < v_{lim}^-(u_l)$, which means there exists a parameter $u_n > u_l$ such that $v_{lim}(u) > v_s(u_l)$ for $u \in (u_l, u_n)$. As a consequence, starting at $(u_l, v_s(u_l))$, we can use the tangential acceleration A to generate a piece of integration trajectory without violate the CEVLC (See Fig. 2(a)). In case (b2), there exists a parameter $u_n > u_l$ such that the tangential acceleration of the CEVLC must be strictly larger than A for $u \in (u_l, u_n)$. By Lemma 2.5, starting from point $(u_l, v_s(u_l))$, we can use the tangential acceleration A without violate the CEVLC (See Fig. 2(b)). We thus prove the correctness of step 8. The correctness step 9 can be proved in a similar way. Furthermore, we have

Lemma 2.6 *In step 9 of the algorithm, the backward integration trajectory v_b intersects v_s only once if there exist no overlap curve segments.*

Proof. Two situations lead to the execution of step 9: step (a3) and step (b3). In case (a3), the key point at u_l is of the first type and $v_{lim}^+(u_l) \geq v_s(u_l) > v_{lim}^-(u_l)$. Then v_s and v_b must intersect because v_b decreasing with the acceleration $-A$ and v_s is bounded by ±A. By Lemma 2.5, v_s and v_c can intersects only once if there exist no overlap curve segments. Moreover, if there exists an overlap curve segment of v_b and v_s, then there exists no intersection point of v_b and v_s except the overlap curve segment by Lemma 2.5. Case (b3) can be proved similarly. ▮

Similarly, we can show that v_s and v_e in step 4 of the algorithm only intersect once if there exist no overlap curve segments. And we can use the bisection method to compute the intersection point.

We now prove the main result of the paper.

Theorem 2.7 *The velocity curve computed with Algorithm* **VP_CTEA** *is the velocity curve defined in equation (9) and is the only solution to the optimization problem (5).*

Proof. Let $v(u)$ be the velocity curve computed with the algorithm. It is clear that $v(u)$ is below the CEVLC and the tangential acceleration $a_T(u)$ of $v(u)$ satisfies $|a_T(u)| \leq A$.

Also, if $|a_T(u)| \neq A$, the corresponding $v(u)$ must be a segment of feasible CEVLC. As a consequence, $v(u)$ satisfies conditions (6) and is bang-bang.

From the algorithm, it is clear that $v(u)$ consists of pieces of $v_P(u)$ for all key points P of the CEVLC including the start point $(0, 0)$ and the ending point $(1, 0)$. To prove (9), it suffices to show that for each key point P, if $v_P(u^*) \neq v(u^*)$ for a parametric value u^*, then $v_P(u^*) > v(u^*)$. From the algorithm, it is clear that all key points including the starting and ending points are on or above $v(u)$. Let $P = (u_0, v_0)$ be a key point. Then $v_0 \geq v(u_0)$. In $[u_0, 1]$, the tangential acceleration of $v_P(u)$ is A. Then by lemma 2.5, $v_P(u) \geq v(u)$ for $u \in [u_0, 1]$. In $[0, u_0]$, if we consider the movement from u_0 to 0, then the acceleration of $v_P(u)$ is also A, and hence $v_P(u) \geq v(u)$ for $u \in [0, u_0]$. As a consequence, $v_P(u)$ cannot be strictly under $v(u)$ at any u. We thus prove that $v(s)$ is the curve in (9).

We now prove that $v(s)$ is an optimal solution. We will prove a stronger result, that is, the velocity curve $q(u) = v^2(u)$ obtained by the algorithm is the maximally possible velocity at each parametric value u. To prove this, we assume that there exists another velocity curve $v_*(u)$ satisfying the constraints listed in (6), and there exists a $u_* \in [0, 1]$ such that $v_*(u_*) > v(u_*)$. Let $q_*(u) = v_*^2(u)$.

The parametric interval $[0, 1]$ is divided into sub-intervals by the key points of CEVLC on the final velocity curve and intersection points in steps 4, 8, 9 of the algorithm. From the algorithm, we can see that on each of these intervals, $v(u)$ could be a segment of the CEVLC, an integration trajectory with tangential acceleration A in the $+u$ direction, which is called an *increasing interval*, or an integration trajectory with tangential acceleration $-A$ in the $+u$ direction, which is called a *decreasing interval*. Furthermore, if $[u_1, u_2]$ is an increasing interval, the starting point $(u_1, v(u_1))$ must be a key point of the CEVLC; if $[u_1, u_2]$ is a decreasing interval, the end point $(u_2, v(u_2))$ must be a key point of the CEVLC.

According to the definition of the CEVLC, u_* cannot be on the CEVLC and thus must be in an increasing or decreasing interval. Firstly, let u_* be in an increasing interval $[u_1, u_2]$. Since $(u_1, v(u_1))$ is a key point on the CEVLC, we have $v_*(u_1) \leq v(u_1)$. Since $v_*(u_*) > v(u_*)$ and v_*, v are continuous curves, there exists a $u_0 \in [u_1, u_*]$ such that $v_*(u_0) = v(u_0)$. On $[u_0, u_*]$, using Lemma 2.5, we obtain a contradiction. Secondly, let $u_* \in [u_1, u_2]$ and $[u_1, u_2]$ be a decreasing interval. We can consider the movement from u_2 to u_1 and the acceleration of v becomes A and the theorem can be proved similarly to the case of increasing intervals. ▮

2.6 Further improvements of the algorithm

In this section, we present modifications to **Algorithm VP_CETA** to improve its efficiency by getting rid of some unnecessary computations.

We first prove a lemma.

Lemma 2.8 *Let $(u_l, v_{lim}(u_l))$ and $(u_n, v_{lim}(u_n))$ be two adjacent key points of the CEVLC. Then an integration trajectory intersects the curve segment $v_{lim}(u), u \in (u_l, u_n)$ once at most.*

Proof. We denote by $i(u)$ an integration trajectory. Without loss of generality, we assume that the tangential acceleration of $i(u)$ is A; otherwise, consider the $-u$ direction. Since the two key points are adjacent, there are three cases: (a): $a_T(u) > A$, (b): $a_T(u) < -A$, or (c):

$-A < a_T(u) < A$[2)] for $u \in (u_l, u_n)$.

Case (a). Since $i(u_l) \le v_{lim}(u_l)$, we have $i(u) < v_{lim}(u)$ for $u \in (u_l, u_n)$ by Lemma 2.5. That is, if they interest, they must intersect at $u = u_l$.

Case (b). There are two cases. If $v^+_{lim}(u_n) < i(u_n)$, then they must intersect once since $i(u_l) \le v_{lim}(u_l)$ and the two curve segments are continuous. Due to Lemma 2.5, they cannot intersect more than one time. If $v^+_{lim}(u_n) > i(u_n)$, then follow the $-u$ direction, $a_T(u) > A$, by Lemma 2.5 we have $i(u) < v_{lim}(u)$ for $u \in (u_l, u_n)$, thus they do not intersect.

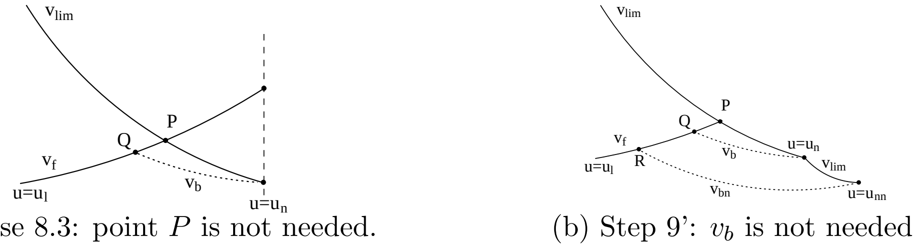
Case (c). There are two cases. If $v^+_{lim}(u_n) < i(u_n)$ then $v_{lim}(u)$ and $i(u)$ must intersect. Assuming that $(u_*, i(u_*))$ is an intersection point. Then, from this point in the $+u$ direction, we have $a_T(u) < A$, hence, $v_{lim}(u) < i(u)$ for $u \in (u_*, u_n)$ by Lemma 2.5. From this point in the $-u$ direction, we have $v_{lim}(u) > i(u)$ for $u \in (u_l, u_*)$. So $i(u)$ intersects $v_{lim}(u), u \in (u_l, u_n)$ only once. If $v^+_{lim}(u_n) > i(u_n)$, we have $a_T(u) > -A$ in the $-u$ direction. By Lemma 2.5, we have $i(u) < v_{lim}(u)$ for $u \in (u_l, u_n)$ and they do not intersect. ▮

Remark 2.9 Step 8 of **Algorithm VP_CETA** can be modified as follows. Let u_l be the current parametric value, u_n the parametric value for the next key point of the CEVLC, and $v_f(u)$ the forward integration trajectory starting from point $(u_l, v_s(u_l))$. Then the tangential acceleration of $v_f(u)$ is A in the $+u$ direction. We can modify step 8 as follows:

8.1 If $v_f(u_n) < v_{lim}(u_n)$, by Lemma 2.8, v_f does not meet the CEVLC in $(u_l, u_n]$ and we can repeat this step for the next segment of CEVLC until either $u_n = 1$ or $v_f(u_n) \ge v_{lim}(u_n)$.

8.2 If $v^+_{lim}(u_n) \ge v_f(u_n) \ge v^-_{lim}(u_n)$ or $v^+_{lim}(u_n) = v_f(u_n) \le v^-_{lim}(u_n)$, then v_f meets the CEVLC at $(u_n, v_f(u_n))$.

8.3 Otherwise, we have $v_f(u_n) > v^+_{lim}(u_n)$ and v_f meets the CEVLC in (u_l, u_n) at a unique point P by Lemma 2.8. Furthermore, if the current CEVLC segment is not feasible, we need not to compute this intersection point. Because, in the next step, we will execute step 9 by computing the backward integration curve v_b from $u = u_n$ and compute the intersection point Q of v_f and v_b. Point P is above v_b and will not be a part of the final velocity curve (Fig. 6(a)).



(a) Case 8.3: point P is not needed.

(b) Step 9': v_b is not needed.

Fig. 6. Modifications of steps 8 and 9

In step 8.3, we need to compute the intersection point between an integration curve and a feasible CEVLC or between a backward integration curve and a forward integration curve.

[2)] In this case, the curve segment is feasible.

We can use numerical method to compute it. A simple but useful method to compute these points is the bisection method, since the intersection point is unique.

Steps 2 and 3 of the algorithm can be modified similarly as step 8.

Remark 2.10 Step 9 can be simplified as follows. We call a parameter u_n *useless* if $v_{lim}^{+}(u_n) \geq v_{lim}^{-}(u_n)$, $a_T^{-}(u_n) \leq -A$, and the next CEVLC segment is not feasible. Note that the second and third conditions mentioned above are equivalent to the following condition: $a_T(u) < -A, u \in (u_n, u_{nn})$, where u_{nn} is the parametric value for the next key point after u_n. If u_n is useless, then we need not to compute the backward integration trajectory v_b from point $(u_n, v_{lim}(u_n))$. Because the backward integration trajectory v_{bn} starting from u_{nn} will be strictly under v_b due to Lemma 2.5 (Fig. 6(b)), and as a consequence v_b will not be a part of the final velocity curve due to Theorem 2.2. So, step 9 can be modified as follows.

Step 9'. If u_n is useless, we will choose the next key point as u_n and repeat this step until either $u_n = 1$ or u_n is not useless. Use this u_n to compute the backward integration trajectory and update v_s.

Remark 2.11 Due to Theorem 2.2 and Lemma 2.5, we can give the following simpler and more efficient algorithm. The input and output of the algorithm are the same as that of **Algorithm VP_CETA**. After computing the CEVLC and its key points, we compute the velocity curve as follows.

1 Let $\mathcal{P}$ be the set of key points of the CEVLC and the starting and ending points $(0, 0)$ and $(1, 0)$. Set the velocity curve to be the empty set.

2 Repeat the following steps until $\mathcal{P} = \emptyset$.

3 Let $P = (u, v_{lim}(u)) \in \mathcal{P}$ be a point with the smallest velocity $v_{lim}(u)$ and remove P from $\mathcal{P}$.

4 Let V_P^f and V_P^b be the forward and backward integration trajectories starting from point P respectively, which can be computed with the methods in Remark 2.9. If, starting from point P, the left (right) CEVLC segment is feasible, V_P^b (V_P^f) is set to be this segment. Find the intersection points of V_P^b (V_P^f) and the existence velocity curve if needed. Update the velocity curve using V_P^f and V_P^b.

5 Remove the points in $\mathcal{P}$, which are above the curve V_P^f or V_P^b. This step is correct due to Theorem 2.2 and Lemma 2.5.

The main advantage of the above algorithm is that many key points are above these integration trajectories and we do not need to compute the integration trajectories starting from these points. Also, all the integration trajectories computed in this new algorithm will be part of the output velocity curve, because each of them starts from the key point which are not processed and has the smallest velocity.

3 Time optimal velocity planning of quadratic B-splines and cubic PH-splines

In Algorithm **VP_CETA**, we assume that $\int \sigma du$ is computable. In this section, we will show that for quadratic B-splines and cubic B-splines, we can have a complete and efficient time optimal velocity planning algorithm by giving explicit formulas for $\int \sigma du$. We implemented our algorithm in these two cases in Maple and used examples from real manufacturing parts to show the feasibility of our algorithm.

3.1 Velocity planning for quadratic B-splines

Let $C(u), u = 0..1$ be a quadratic B-spline. Since $C(u)$ has only C^1 continuity and a quadratic curve has no singular points, the connection points of the spline are all the key points of first or second type. To compute the key points of third type, consider a piece of the spline:

$$\begin{aligned}&C(u) = (x(u), y(u), z(u))\\&= (a_0 + a_1u + a_2u^2, b_0 + b_1u + b_2u^2, c_0 + c_1u + c_2u^2),\end{aligned}$$

where $a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2$ are constants. The curvature of $C(u)$ is $k(u) = \frac{|C' \times C''|}{\sigma^3}$, where $\sigma = |C'| = \sqrt{x'^2 + y'^2 + z'^2}$. For quadratic curves,

$$|C' \times C''| = \sqrt{(b_1c_2 - c_1b_2)^2 + (c_1a_2 - a_1c_2)^2 + (a_1b_2 - b_1 * a_2)^2}$$

is a constant. Hence the CEVLC is

$$q = v^2 = \frac{B}{|k(u)|} = \frac{B\sigma^3}{|C' \times C''|} = D\sigma^3,$$

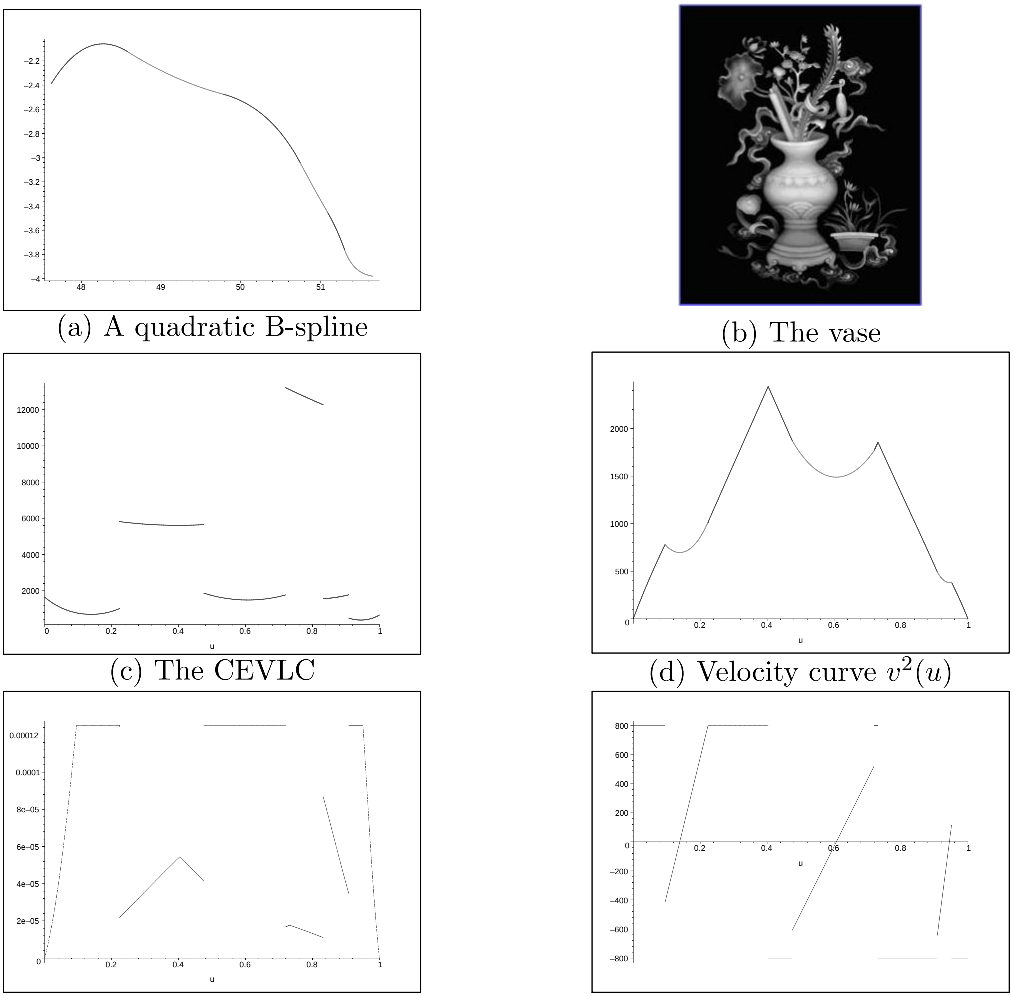
where D is a constant. The tangential acceleration along the CEVLC is $a_T = \frac{(D\sigma^3)'}{2\sigma} = \frac{3}{2}D\sigma\sigma' = \frac{3}{4}D(\sigma^2)'$. Hence, $a_T(u)$ is a linear function in the parameter u. Then, the key points of third type can be computed by solving linear equations $a_T = \pm A$. From the equations, we can see that for each piece of the quadratic B-splines, there are two key points of third type at most.

Now, we show how to compute the integration trajectory. When the tangential acceleration reaches its bounds $\pm A$, we need to compute the solution of the differential equation:

$$q' = \pm 2A\sigma, \tag{11}$$

where $\sigma = \sqrt{au^2 + bu + c}$. Let $i(u) = \pm 2A[1/4\,\frac{(2\,au+b)\sqrt{au^2+bu+c}}{a} + 1/2\,\ln(\frac{1/2\,b+au}{\sqrt{a}} + \sqrt{au^2 + bu + c})$ $c\frac{1}{\sqrt{a}}] - 1/8\,\ln(\frac{1/2\,b+au}{\sqrt{a}} + \sqrt{au^2 + bu + c})b^2a^{-3/2}$. Then, $q(u) = i(u) - i(u^*) + q(u^*)$ is the solution of the differential equation (11) with initial value $(u^*, q(u^*))$.

We use an example to illustrate the algorithm. The curve in Figure 7(a) is a quadratic B-spline consisting of six pieces of quadratic curve segments, which is from the tool path of the vase in Figure 7(b). Details of the G-codes and splines generated from the G-codes for the vase can be found in [26]. We set the tangential and centripetal acceleration bounds to be $A = 800\ mm/s^2$ and $B = 1000\ mm/s^2$ respectively.



(a) A quadratic B-spline

(b) The vase

(c) The CEVLC

(d) Velocity curve $v^2(u)$

(e) Chord error $\delta(u)$

(f) Tangential acceleration

Fig. 7. Optimal velocity planing for a quadratic B-spline

According to the algorithm, we first compute the CEVLC with the maximal centripetal acceleration, which is shown in Figure 7(c). Some parts of the $1, 2, 3, 5, 6$-th pieces of the CEVLC are feasible. Secondly, we compute the integration trajectory v_s from the starting point $(u_0, v_0) = (0, 0)$ by tangential acceleration A, which intersects the feasible part of the first CEVLC segment. Find the intersection point of v_s and the feasible CEVLC segment. Then, compute the backward integration trajectory v_e from $(1, 0)$, similarly as above, it intersects the sixth CEVLC segment. Then, execute step 7 and merge the feasible part of the first CEVLC segment into v_s. Let the end point of the feasible CEVLC (next key point) be the current point with parameter u_0. Now, since $a_T^-(u_0) \geq A$, we execute step 8 by computing the integration trajectory v_f with tangential acceleration A. The segment of v_f between the current point and the first connection point of the spline is under the CEVLC. Merge v_f into v_s. Let u_1 be the parametric value of the first connection point. Since $v_{lim}^+(u_1) < v_{lim}^-(u_1)$, we execute step 8 again by computing a new v_f starting from $(u_1, v_{lim}^+(u_1))$ with tangential acceleration A. v_f intersects the CEVLC at the second connection point of the spline. Merge v_f into v_s. Let u_2 be the parametric value of the second connection point. Since $v_s(u_2) > v_{lim}^-(u_2)$, according to the algorithm, we execute step 9 of by computing the backward integration trajectory v_b from $(u_2, v_{lim}^-(u_2))$, which intersects v_s only once. Update v_s accordingly. In this way, we can obtain the final velocity curve shown in Figure 7(d), which consists of eleven pieces, where the $1, 3, 4, 5, 7, 8, 9, 11$-th pieces are controlled by the tangential acceleration, and the $2, 6, 10$-th pieces are feasible CEVLC segments. Figure 7(e) is the chord error of the optimal velocity curve with a sampling period $T = 1ms$, which means $d = 1.25\mu m$. Figure 7(f) is its tangential acceleration. From these two figures, we can see that the control is "Bang-Bang."

Now, we give a more complicated example shown in Figure 8(a), which is a complete tool path segment $C(u) = (x(u), y(u))$ of the vase in Figure 7(b) from top to bottom and consists of 57 quadratic B-spline curve segments and 5 long straight line segments. Figure 8(b) is its velocity limited curve with centripetal acceleration $B = 1000\ mm/s^2$. Figure 8(c) is the optimal velocity curve computed with our algorithm with tangential acceleration $A = 800\ mm/s^2$ and centripetal acceleration $B = 1000\ mm/s^2$, which consists of 127 curve segments. Figure 8(d) is the chord error of the optimal velocity curve with a sampling period $T = 1ms$, which means $d = 1.25\mu m$. Figure 8(e) is its tangential acceleration. Note that in the connection point of a straight line segment and a quadratic B-spline, the velocity decreases to zero.

3.2 Velocity planning for cubic PH-splines

Let $C(u), u = 0..1$ be a cubic PH-spline. Since a cubic PH-spline only has C^1 continuity, the connection points of the PH-spline are all the key points of first or second type of the CEVLC. Let $r(u)$ be a piece of cubic PH-curve of $C(u)$. Then, $r'(u)$ has the following representation [8]:

$$\begin{aligned} r'(u) =&(f(u)^2 + g(u)^2 - m(u)^2 - n(u)^2,\\ &2(f(u)n(u) + g(u)m(u)),\\ &2(g(u)n(u) - f(u)m(u))). \end{aligned} \tag{12}$$

where $f(u), g(u), m(u), n(u)$ are linear functions in u.

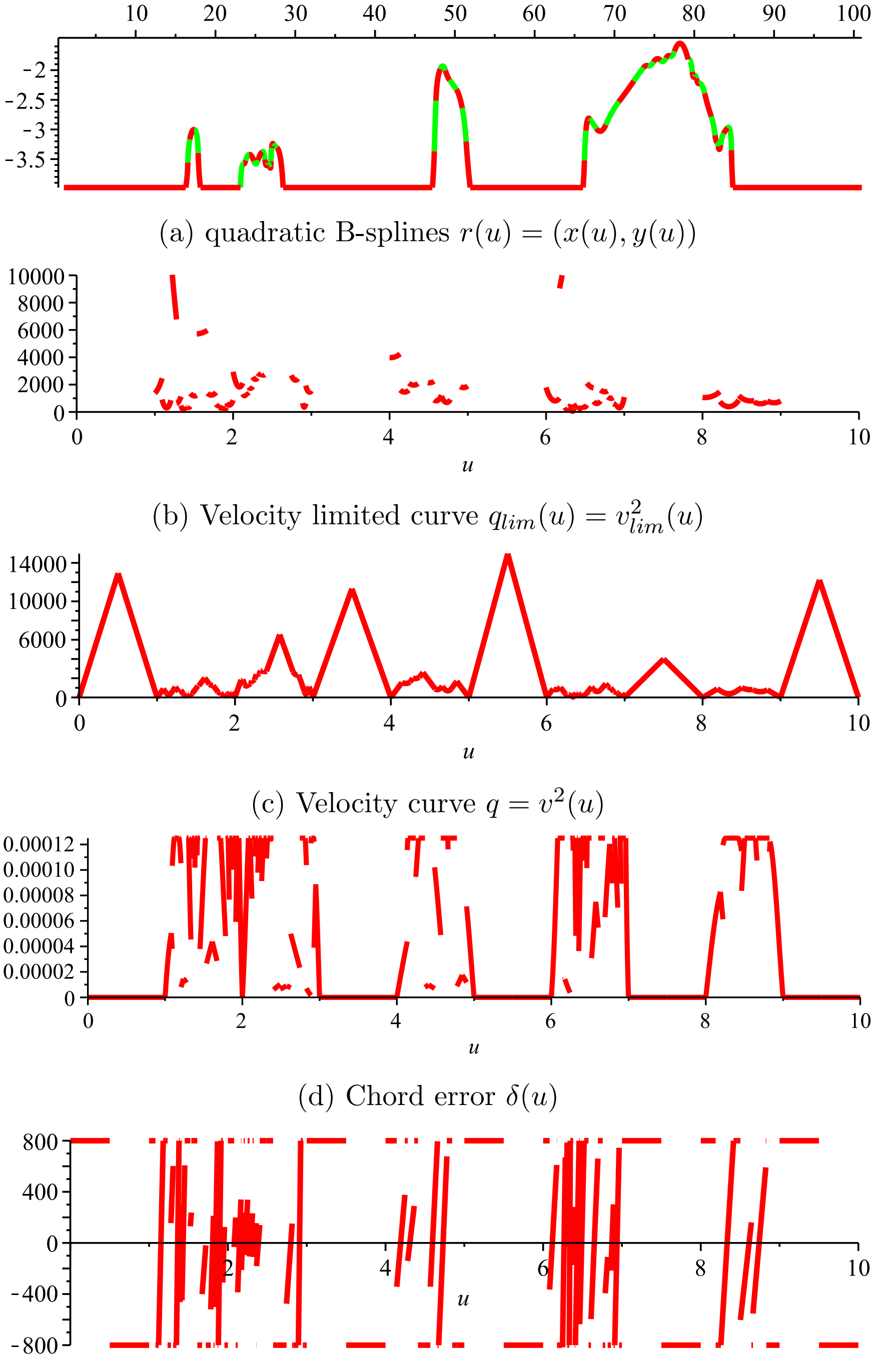
The curvature of $r(u)$ is $k(u) = \frac{|r' \times r''|}{\sigma^3} = \frac{E}{\sigma^2}$, where $\sigma = |r'| = f(u)^2 + g(u)^2 + m(u)^2 + n(u)^2$ and E is a constant. The CEVLC of $r(u)$ is

$$q = v^2 = \frac{B}{|k(u)|} = G\sigma^2,$$

where G is a constant. The tangential acceleration along the CEVLC is: $a_T = \frac{(G\sigma^2)'}{2\sigma} = G\sigma'$. Since σ is of degree two, $a_T(u)$ is a linear function in the parameter u. The key points of the third type can be computed by solving linear equations $a_T = \pm A$ directly.

For a cubic PH-curve, when the velocity is controlled by the tangential acceleration, the integration trajectory is a polynomial in u with degree three. The integration trajectory is the solution of the following differential equation:

$$q' = \pm 2A\sigma = \pm 2A(au^2 + bu + c), \tag{13}$$



(a) quadratic B-splines $r(u) = (x(u), y(u))$

(b) Velocity limited curve $q_{lim}(u) = v_{lim}^2(u)$

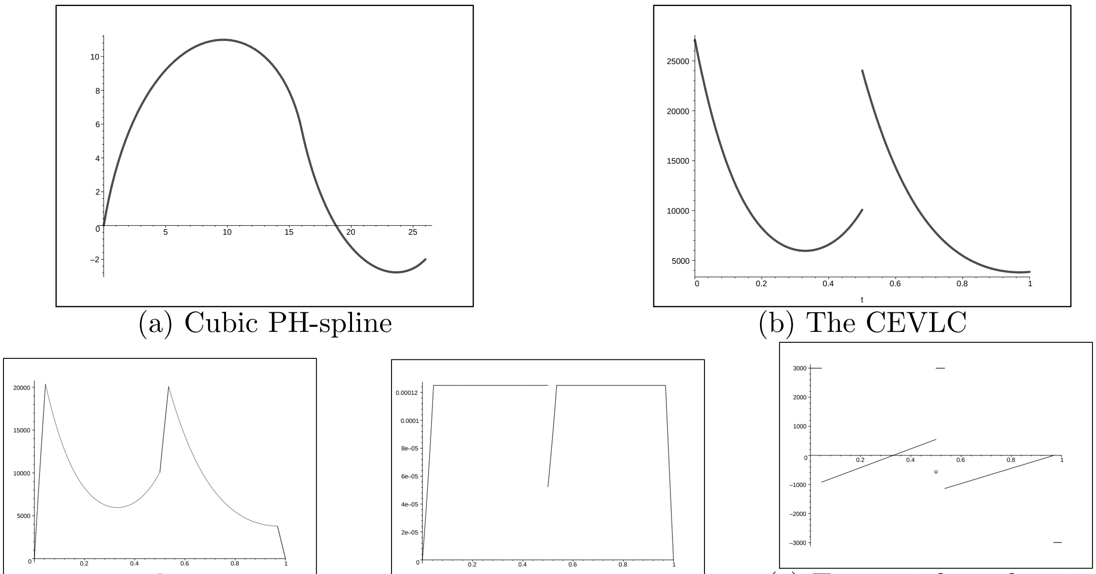
(c) Velocity curve $q = v^2(u)$

(d) Chord error $\delta(u)$

(e) Tangential acceleration $a_T(u)$

Fig. 8. Optimal velocity planing for quadratic B-splines from a vase

where a, b, c are constants. Let $i(u) = \pm 2A(1/3\,au^3 + 1/2\,bu^2 + cu)$. Then, $q(u) = i(u) -$



(a) Cubic PH-spline

(b) The CEVLC

(c) Velocity $q = v^2(u)$

(d) Chord error $\delta(u)$

(e) Tangential acceleration of velocity curve

Fig. 9. Optimal velocity planning for a cubic PH-spline

$i(u^*) + q(u^*)$ is the solution of the differential equation (13) with initial value $(u^*, q(u^*))$.

Now, we give an illustrative example for a cubic PH-spline (Fig. 9(a)).

The cubic PH-spline consists of two pieces of PH-curves and has C^1 continuity. Let the tangential and centripetal acceleration bounds be $A = 3000mm/s^2$ and $B = 1000mm/s^2$ respectively. Here are the details to get the velocity curve. Firstly, we compute the CEVLC of the curve which consists of two feasible segments. So the key points are the starting point, the ending point, and the connection point of the spline. Let u_* be the parametric value of the connection point of the spline. Secondly, we compute the integration trajectory v_s from the starting point $(u_0, v_0) = (0, 0)$ by tangential acceleration A. Since $v_s(u_*) > v_{lim}(u_*)$, we see that it must intersect the first CEVLC segment. Find the intersection point $(u_1, v_{lim}(u_1))$ of v_s and the first CEVLC segment. Let $[0, u_1]$ be the defining interval of v_s. Compute the backward integration trajectory v_e from the ending point $(1, 0)$. Since $v^+_{lim}(u_*) < v_e(u_*) < v^-_{lim}(u_*)$, v_e does not intersect the second CEVLC segment. Let $(u_*, v_e(u_*))$ be the intersection point of v_e and the CEVLC and $[u_*, 1]$ be the defining interval of v_e. Starting from $(u_1, v_{lim}(u_1))$, the CEVLC is feasible, we merge this feasible part of CEVLC into v_s. Let $(u_*, v_s(u_*))$ be the current point. Since $v_s(u_*) = v^+_{lim}(u_*) < v_e(u_*)$, execute step 8 by computing the integration trajectory v_f which intersects v_e at $(u_2, v_f(u_2))$. The final velocity curve has four segments, where the $1, 3, 4$-th segments are controlled by the tangential acceleration and the 2-th piece is a feasible CEVLC segment.

The CEVLC, velocity curve , chord error and tangential acceleration of the velocity curve are shown in Figure 9, where the sampling period is $T = 1ms$.

4 Feed-rate override in NC-machining

During the NC-machining, there exists another constraint: the maximal feed-rate v_{max}. In this situation, all we need to do is to change the velocity curve to $q^*(u) = \min(q(u), v_{max})$, where $q(u)$ is the optimal velocity curve obtained in the preceding section. The procedure of the interpolation is as follows.

Algorithm 4.1 ***Interpolation algorithm***

Input: current parameter u_i*, the velocity curve* $q(u)$*, maximum feed-rate* v_{max}*, and the sampling time* T

Output: the parameter of the next interpolation point u_{i+1}

1. *According to the velocity curve and the maximum feed-rate, let* $v_i = \min(\sqrt{q(u_i)}, v_{max})$*. The step size is* $\Delta L = v_i \cdot T$*.*
2. *According to the step size* ΔL*, compute the parameter of the next interpolation point* u_{i+1}*([4, 21]).*

Since for any parametric value u, whenever the value of $q^*(u)$ is taken from $q(u)$ or v_{max}, the left and right limitations of the tangential acceleration are satisfied. Hence, $q^*(u)$ satisfies the maximum feed-rate, chord error, and tangential acceleration bounds. Furthermore, in CNC-machining, the users can change the maximum feed-rate during manufacturing, which is called *feed-rate override*. Although the new feed-rate limitation is not required to respond immediately, the real velocity should decrease to the new lower speed as soon as possible. The following algorithm solves the feed-rate override problem efficiently.

Algorithm 4.2 ***Feed-rate override algorithm***

Input: the velocity curve $q(u)$*, current parameter* u_i*, the current feed-rate* v_**, the modified feed-rate limitation* v'_{max}*;*

Output: the parametric values $u_{i+1}, u_{i+2}, \ldots$ *of the interpolation points.*

1 *If* $v_* > v'_{max}$*, then let* $v'_i = v_* - AT$*,* $v''_i = \max(v'_i, v'_{max})$*,* $v_i = \min(v''_i, \sqrt{q(u_i)})$*. Find the next interpolation point* u_{i+1} *according to Algorithm* 4.1*,* $i = i + 1, v_* = v_i$*, and repeat step* 1*;*

2 *If* $v_* \leq v'_{max}$*, then let* $v_i = \min(\sqrt{q(u_i)}, v'_{max}, v_* + AT)$*. Find the next interpolation point* u_{i+1} *according to Algorithm* 4.1*. If* $u_{i+1} > 1$*, then let* $u_{i+1} = 1$ *and terminate; else let* $i = i + 1, v_* = v_i$*, and repeat step* 2*.*

Since the final velocity is under the CEVLC, the error bound is satisfied. Step 1 of the above algorithm is to slow down feed-rate as soon as possible when the current feed-rate is larger than the modified feed-rate. Step 2 of the above algorithm is exactly Algorithm 4.1, where the maximum feed-rate is replaced by the modified feed-rate.

One advantage of the algorithm presented in this paper is that feed-rate override can be carried out easily. In the case of multi-axis acceleration mode, when the maximal feed-rate is changed to v_{max}, we cannot simply take $q^*(u) = \min(q(u), v_{max})$ to be the new velocity curve.

5 Conclusion

In this paper, we give the first time-optimal velocity planning method for parametric curves with confined chord error. We adopt the simplest acceleration mode: the linear acceleration for tangential accelerations. With the method introduced in this paper, it is not difficult to give time-optimal velocity planning method with confined chord error and multi-axis acceleration modes.

The key idea is to reduce the chord error bound to a centripetal acceleration bound. When the centripetal acceleration reaches its bound, the velocity curve is an algebraic curve and is called the CEVLC. With the CEVLC, the final velocity curve is the minimum of all the integration velocity curves starting from the key points of the CEVLC the starting point, and the ending point. We also give a practical algorithm to compute the time-optimal velocity curve and implemented the algorithm for two types of curves.

References

[1] D. Bedi, I. Ali, N. Quan. Advanced techniques for CNC machines, *Journal of Enginerring for Industry*, 115, 329-336, 1993.

[2] G. Birkhoff, G. Rota. *Ordinary differential equations.* Newyork, Blaisdell, 1969.

[3] J.E. Bobrow, S. Dubowsky, J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *Int. J. Robot. Res.*, 4(3), 3-17, 1985.

[4] C.W. Cheng, M.C. Tsai. Real-time variable feed rate NURBS curve interpolator for CNC machining. *Int. J. Adv. Manuf. Technol.*, 23, 865-873, 2004.

[5] J.J. Chou and D.C.H. Yang. Command generation for three-axis CNC machining. *Journal of Engineering for Industry*, 113, 305-310, 1991.

[6] M.M. Emami, B. Arezoo. A look-ahead command generator with control over trajectory and chord error for NURBS curve with unknown arc length. *Computer-Aided Design*, 4(7), 625-632, 2010.

[7] K. Erkorkmaz, Y. Altintas. High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. *Int. J. of Mach. Tools and Manu.*, 41, 1323-1345, 2001.

[8] R.T. Farouki. *Pythagorean-hodograph curves*, Springer, Berlin, 2008.

[9] R.T. Farouki, Y.F. Tsai. Exact Taylor series coefficients for variable-feedrate CNC curve interpolators. *Computer-Aided Design*, 33(2), 155-165, 2001.

[10] J.Y. Lai, K.Y. Lin, S.J. Tseng, W.D. Ueng. On the development of a parametric interpolator with confined chord error, feedrate, acceleration and jerk. *Int. J. Adv. Manuf. Technol.* , 37: 104C121, 2008.

[11] M.T. Lin, M.S. Tsai, H.T. Yau. Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm. *Int. J. of Mach. Tools and Manu.*, 47(15), 2246-2262, 2007.

[12] S. Macfarlane, E.A. Croft. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Trans. on Robot. and Automa.*, 19: 42-52, 2003.

[13] S.H. Nam, M.Y. Yang. A study on a generalized parametric interpolator with real-time jerk-limited acceleration. *Computer-Aided Design* 36, 27-36, 2004.

[14] J. Park, S.H. Nam, M.Y. Yang. 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, 2005.

[15] Z. Shiller. On singular time-optimal control along specified paths. *IEEE Trans. Robot. Autom.*, 10, 561-566, 1994.

[16] Z. Shiller, H.H. Lu. Robust computation of path constrained time optimal motions. *Proc., IEEE Inter. Conf. on Robot. Autom.*, Cincinnati, OH, 144-149, 1990.

[17] K.G. Shin, N.D. McKay. Minimum-time control of robotic manipulators with geometric path contraints. *IEEE Trans. on Automatic Control*, 30(6), 531-541, 1985.

[18] M. Shpitalni, Y. Koren, C.C. Lo. Realtime curve interpolators. *Computer-Aided Design* 26(1), 832-838, 1994.

[19] S.D. Timar, R.T. Farouki, T.S. Smith, C.L. Boyadjieff. Algorithms for time-optimal control of CNC machines along curved tool paths. *Robotics and Computer-Integrated Manufacturing*, 21(1), 37-53, 2005.

[20] S.D. Timar, R.T. Farouki. Time-optimal traversal of curved paths by Cartesian CNC machines under both constant and speed-dependent axis acceleration bounds. *Robotics and Computer-Integrated Manufacturing*, 23(5), 563-579, 2007.

[21] Z.M. Xu, J.C Chen and Z.J Feng. Performance Evaluation of a Real-Time Interpolation Algorithm for NURBS Curves. *Int. J. Adv. Manuf. Technol.*, 20: 270-276, 2002.

[22] D.C.H. Yang, T. Kong. Parametric interpolator versus linear interpolator for precision CNC machining. *Computer-Aided Design*, 26(3), 225-234, 1994.

[23] S.S. Yeh, P.L. Hsu. Adaptive-feedrate interpolation for parametric curves with a confined chord error. *Computer-Aided Design*, 34, 229-237, 2002.

[24] T. Yong, R. Narayanaswami. A parametric interpolator with confined chord errors, acceleration and deceleration for NC machining. *Computer-Aided Design*, 35, 1249-1259, 2003.

[25] K. Zhang, X.S. Gao, H. Li, C.M. Yuan. A greedy algorithm for feed-rate planning of CNC machines along curved tool paths with jerk constraints. *MM Research Preprints*, 29, 189-205, 2010.

[26] M. Zhang, W. Yan, C.M. Yuan, D. Wang, X.S. Gao. Curve fitting and optimal interpolation on CNC machines based on quadratic B-splines(in Chinese). *MM Research Preprints*, 29, 71-91, 2010.