Mathematics Mechanization Research Preprints
KLMM, Chinese Academy of Sciences
Vol. 30, 57–89, May 16, 2011

Time-Optimal Interpolation for CNC Machining along Curved Tool Pathes with Confined Chord Error[1)]

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

Abstract. Interpolation algorithms are key factors to achieve high precision and high speed CNC machining. In this paper, two new interpolation algorithms for CNC machining along curved pathes are proposed: a time-optimal interpolation algorithm under chord error, feedrate, and tangential acceleration bounds, and a greedy interpolation algorithm under chord error and tangential jerk bounds. 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. Then, the velocity planning is to find the proper velocity curve governed by the tangential acceleration or jerk bound "under" the chord error velocity limit curve. For two types of simple tool pathes, explicit formulas for the velocity curve are given. We implement the methods in these two cases and conduct real CNC machining to show the feasibility of the methods.

Keywords. CNC controller, parametric curve interpolation, time-optimal velocity planning, chord error, jerk, velocity limit curve, quadratic B-spline, cubic PH curve.

1. introduction

In modern CAD systems, the standard representation for freeform surfaces is parametric functions. While the conventional computed numerically controlled (CNC) controllers mainly use micro line segments (G01 codes) to represent the machining path. To convert the parametric curves into line segments may lead to problems such as large data storage, speed fluctuation, and poor machining accuracy. Chou, Yang, and Shpitalni et al proposed to use smooth parametric curves generated in CAD/CAM systems directly in the CNC controllers to overcome these drawbacks [5, 18, 24]. Now, commercial CNC controller corporations, such as Siemens [19], also provide spline interpolation methods in their high-end products.

Interpolation algorithms, which control how the machine tool moves along the manufacturing path, play a key role in high speed and high precisions CNC machining. An interpolation algorithm in the CNC controller usually consists of two phases: velocity planning and parameter computation. Let $C(u)$ be the manufacturing path. The phase to determine the

[1)] Partially supported by a National Key Basic Research Project of China (2011CB302400) and by a grant from NSFC (60821002).

[2)]This is an extended version of the paper: C.M. Yuan, X.S. Gao. Time-optimal interpolation of CNC machines along parametric path with chord error and tangential acceleration bounds. *MM Research Preprints*, 29, 165-188, 2010.

feedrate $v(u)$ along $C(u)$ is called velocity planning. When the feedrate $v(u)$ is known, the phase 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, please refer to [4, 9, 23] and the literatures therein. Two types of accelerations are usually 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[24] used a uniform parametric feedrate without considering the chord error. Yeh and Hsu used a chord error bound to control the feedrate if needed and used a constant feedarte in other places [25]. However, the machine acceleration capabilities were not considered. Narayanaswami and Yong [26] gave a velocity planning method based on tangential acceleration bounds by computing maximal passing velocities at sensitive corners. Similar ideas were used to compute the feedrate for micro line segments connected with quadratic curve segments in [28]. Cheng and Tsai [4] gave an interpolation method based on different velocity profiles for the tangential acceleration. Velocity planning methods with tangential acceleration and jerk bounds were considered by several research groups [7, 11, 13, 14, 12]. Emami, Arezoo [6] and Lai et al[10] proposed velocity planning methods with confined acceleration and jerk in each axis, and confined chord error 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, 25], 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 all the work mentioned above, discrete models are used. The advantage for such an approach is that the algorithms are generally easier to implement and can be used for more types of tool pathes. On the other hand, some key properties, such as time-optimality and chord error, cannot be guarantied. Using a continuous or analytical model, 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 [21, 22] proposed a time-optimal velocity planning algorithm in CNC machining under the same acceleration constraints. M. Zhang et al simplified the method in [21, 22] for quadratic B-splines and realized real-time manufacturing on industrial CNC machines [29]. K. Zhang et al [27] gave a greedy algorithm for velocity planning under multi-axis jerk bounds. These optimal methods use the "Bang-Bang" control strategy, that is, at least one of the axes reaches its acceleration or jerk bound all the time. But they do not consider the chord error which is an important factor for high precision CNC-machining.

In this paper, velocity planning along a curved tool path under a chord error bound and tangential acceleration and jerk bounds is considered. To control the chord error, we introduce the key concept of chord error velocity limit curve (abbr. CEVLC). Other main ingredients of the method include how to compute the analytical formulas for the velocity with a given tangential acceleration or jerk value and how to plan the velocity to give a time-optimal interpolation. In the case of tangential acceleration bound, we give a time-optimal velocity planning algorithm, which seems to be the first proved time-optimal algorithm with confined chord error. In the case of tangential jerk bound, we give a greedy velocity planning

algorithm which is time-optimal under certain greedy conditions. To find a time-optimal velocity curve with jerk bounds using methods similar to that in [3, 16, 21] is still an open problem. See [27] for more comments on this problem.

We show that the chord error bound can be approximately reduced to a centripetal acceleration bound. Furthermore, if the centripetal acceleration reaches its bound, the velocity can be written as an algebraic function in the parameter u of the tool 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.

Our algorithm uses the "Bang-Bang" control strategy, which will be proven to be a necessary way to achieve time-optimality. Then the final velocity curve is governed either by the centripetal acceleration bound or by the tangential acceleration or jerk bound, and the later one is called the integration velocity trajectory. The main task of the velocity planning algorithm is to find the switching points between the integration velocity trajectory and the CELVC.

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 on a three axis industrial CNC machine to show the feasibility of our method. To implement our method in CNC controllers, we first compute the velocity curves off-line and then use the velocity curves as parts of the input to the CNC controllers to achieve real-time interpolation. This strategy is adopted in many existing work such as [9, 10, 29].

As a final remark, we want to mention that using the tangential acceleration is optional. For instance, by combining the CEVLC introduced in this paper and the method in [21, 22], it is possible to give a time optimal velocity planning method with confined chord error and acceleration bounds along the x-, y-, and z-axis. Furthermore, by combining the CEVLC and the method proposed by us in [27], it is possible to give an algorithm with confined chord error and multi-axis jerk bounds.

The rest of the paper is organized as follows. Section 2 gives a time-optimal velocity planning algorithm with chord error and acceleration bounds. Section 3 gives a greedy velocity planning algorithm with chord error and jerk bounds. Section 4 gives the details for computing the time optimal velocity curves for quadratic B-splines and cubic PH-splines and the experimental results. Section 5 concludes the paper.

2. Time-optimal velocity planning with chord error and tangential acceleration bounds

In this section, we will give a time-optimal velocity planning algorithm under the chord error, feedrate, and tangential acceleration bound constraints.

2.1. Problem

We consider a spatial piecewise parametric curve $C(u), u = 0..1$ with C^1 continuity, such as B-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.

In order to control the CNC machine cutting tools, we need to know the velocity at

each point on the tool path, 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 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 the 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

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

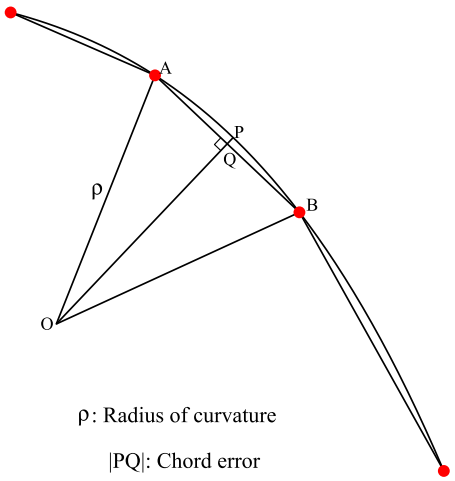


Fig. 1. The chord error

When the cutting tool moves from point A to point B on $C(u)$, the line segment AB is generally considered to be a first order approximation of the machining trajectory, and the distance between line segment AB and the curve segment is called the *chord error* [6, 25, 10], which is one of many sources 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 of the CNC machine, δ the *chord error bound*, and $\pm A_T$ the tangential acceleration bounds. As shown in Fig. 1, the chord error $|PQ|$ is generally taken as

$$|PQ| = \rho - \sqrt{\rho^2 - |AB|^2/4}$$

in the literature [6, 10, 25]. From the above formula, we have $-2|PQ|\rho + |PQ|^2 = -|AB|^2/4$. In general, the chord error $|PQ|$ is much less than the radius of curvature ρ. Based on this fact and omitting the second order small quantity $|PQ|^2$, the chord error formula at each parametric value u is derived [20, page 103]:

$$|PQ| \approx \frac{|AB|^2}{8\rho}.$$

If we denote by $\delta(u)$ the interpolating chord error at u with feedrate $v(u)$, then from the above formula

$$q(u) = v^2(u) = |AB|^2/T^2 \approx \frac{8\delta(u)\rho(u)}{T^2}. \tag{2}$$

Let

$$a_N(u) = v(u)^2/\rho(u) = k(u)v(u)^2 = k(u)q(u) \tag{3}$$

be the centripetal acceleration and

$$A_N = \frac{8\delta}{T^2}. \tag{4}$$

Then, from (2), (3), and (4) the chord error bound is transformed to the centripetal acceleration bound:

$$\delta(u) \le \delta \Longleftrightarrow a_N(u) \le A_N.$$

Since

$$\frac{\mathrm{d}}{\mathrm{d}t} = \frac{\mathrm{d}s}{\mathrm{d}t}\frac{\mathrm{d}u}{\mathrm{d}s}\frac{\mathrm{d}}{\mathrm{d}u} = \frac{v}{\sigma}\frac{\mathrm{d}}{\mathrm{d}u}, \tag{5}$$

the tangential acceleration is

$$a_T(u) = \frac{\mathrm{d}v(u)}{\mathrm{d}t} = \frac{v(u)v'(u)}{\sigma(u)} = \frac{q'(u)}{2\sigma(u)}. \tag{6}$$

From (5), the time optimal velocity planning is to find a velocity curve $v(u)$ such that

$$\min_{v(u)} t = \int_0^1 \frac{\sigma(u)}{v(u)}\mathrm{d}u, \tag{7}$$

under the following constraints

$$a_N(u) \le A_N, u = 0..1, \tag{8}$$

$$|a_T(u)| \le A_T, u = 0..1, \tag{9}$$

where $a_N(u)$ and $a_T(u)$ are the centripetal acceleration and tangential acceleration respectively, and A_N is computed from the chord error bound δ with formula (4).

2.2. CEVLC and its key points

In this section, we define the CEVLC, which will play a key role in our algorithms.

If the centripetal acceleration reaches its bound A_N, that is the equality holds in (8), then from (3) we have $q(u) = \frac{A_N}{k(u)}$, which defines a curve

$$q_{lim}(u) = v_{lim}^2(u) = A_N/k(u) = \frac{8\delta}{k(u)T^2}, u \in [0,1] \text{ or}$$

$$v_{lim}(u) = \sqrt{A_N/k(u)} = \frac{\sqrt{8\delta/k(u)}}{T}, u \in [0,1], \tag{10}$$

in the u-v plane with v as the vertical axis. We call this curve the *velocity limit curve with the chord error bound*, 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.

Certain points on the CEVLC play an important role in our algorithms, which are called *switching points* or *key points*.

The *first type key 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 (10) and (1), the limit velocity v_{lim} is ∞ at these points and the real velocity curve can not reach it, so we can remove the singular points from the key points. Thus, only the connection points need to be considered.

At a first type key point u, the velocity is not continuous and we denote by $v^+(u), v^-(u)$ $(a^+(u), a^-(u))$ the left and right side velocities (accelerations) respectively. If $v^+(u) < v^-(u)$, let the velocity and acceleration of this point be $(v^+(u), a^+(u))$; otherwise, the velocity and acceleration of this point are defined to be $(v^-(u), a^-(u))$.

The *second type key 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. At a second type key point u, the tangential acceleration is not continuous and is defined to be $\min\{a^+(u), a^-(u)\}$.

For a differentiable segment of the CEVLC divided by the above two types of key points, we can further divide it according to whether the tangential acceleration of the CEVLC is $\pm A_T$, where A_T is from (9). A point on the CEVLC is called a *third type key point* if the tangential acceleration along the CEVLC at this point is $\pm A_T$. We can find the third type key points by solving the following algebraic equation in u

$$a_{lim}(u) = \frac{q'_{lim}(u)}{2\sigma(u)} = \left(\frac{A_N \sigma(u)^3}{|C'(u) \times C''(u)|}\right)' / (2\sigma(u)) = \pm A_T.$$

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_T. 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_T. An unfeasible CEVLC segment cannot be a part of the final velocity curve, and the final velocity curve must be strictly under it due to the constraint $a_N(u) \leq A_N$.

If the curve segments on the left and right sides of a second type key point are both feasible, we can delete this key point since it does not affect the velocity planning.

2.3. Integration trajectory

In this section, we will show how to compute the velocity curve when the tangential acceleration reaches its bound.

We use "Bang-Bang" control, that is, at least one of "=" holds in inequalities (8) or (9). If the centripetal acceleration reaches its bound A_N, from Section 2.2., we will obtain the CEVLC. If the tangential acceleration reaches its bound, then from (6) the square of the

velocity $q(u) = v(u)^2$ can be obtained by solving the differential equation $q'(u) = 2A_T\sigma(u)$, whose solution is

$$q = 2A_T\pi(u) + c, \tag{11}$$

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 $(u^*, q(u^*))$ on the velocity curve:

$$c = q(u^*) - 2A_T\pi(u^*).$$

The curve (11) is called an A_T *integration curve* or an A_T *integration trajectory*. If the tangential acceleration reaches the negative bound $-A_T$, we can just replace A_T with $-A_T$ to obtain the $-A_T$ *integration trajectory*.

Before presenting the algorithm, we first give a description of the solution to problem (7), which will be helpful for understanding the algorithm.

Let $v_P(u)$ be the A_T integration trajectory starting from a point $P = (u_P, v_{lim}(u_P))$ on the CEVLC both for the forward (the $+u$) and the backward (the $-u$) directions. Let $v_0(u)$ be the A_T forward integration trajectory from the start point $P_0 = (0, 0)$ and $v_1(u)$ the A_T backward integration trajectory from the end point $P_1 = (1, 0)$. Of course, all the velocity curves mentioned above are defined in $[0, 1]$.

Then, the solution to the optimization problem (7) is given below.

Theorem 2.1 *Let $\mathcal{K}$ be the finite set of key points of the CEVLC. Then*

$$v(u) = \min_{P\in\mathcal{K}}(v_{lim}(u), v_0(u), v_1(u), v_P(u)) \tag{12}$$

is the solution to the optimal problem (7).

We will prove the theorem in the appendix of the paper.

From Theorem 2.1, we see that the time optimal velocity curve is the minimal values of the CEVLC and the integration trajectories passing through the start point $(0, 0)$, the end point $(1, 0)$, and all the key points of the CEVLC in the u-v plane. The algorithm we will given in the next section is an efficient realization of this theorem. Also from the theorem, $v(u)$ is a piecewise continuous curve for $u \in [0, 1]$.

2.4. The time optimal velocity planning algorithm

Since we use the "Bang-Bang" control strategy, the real velocity curve must be either a feasible part of the CEVLC or a segment of an integration trajectory under the CEVLC. What we need to do is to find the "switching points" between these two kinds of curves.

We first give the main idea of the velocity planning algorithm which will compute the velocity curve $v(u)$ in the u-v plane with u as the horizontal axis.

Firstly, we compute the CEVLC, find its key points, and the speeds at the key points. Compute the forward A_T integration trajectory v_s from the start point $(u, v(u)) = (0, 0)$. Find the intersection point $(u_l, v_s(u_l))$ of v_s and the CEVLC. Compute the backward A_T integration trajectory v_e from the end point $(1, 0)$. Find the intersection point $(u_r, v_e(u_r))$ of v_e and the CEVLC.

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

If the next segment of CEVLC starting from point P_c in the forward direction is not feasible and the forward A_T 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. See Fig. 2 for an illustration. Merge $v_f(u), u \in [u_r, u_i]$ into v_s. Let $u_l = u_i$ and set $P_c = (u_l, v_f(u_l))$ to be the new current point.

If the next segment of CEVLC starting from point P_c in the forward direction is not feasible and the A_T 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 A_T 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. See Fig. 3 for an illustration. Set P_n as the new current point.

With the new current point, we can repeat the above procedure until the velocity curve is found.

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 2.2 (VP_CETA) *The input of the algorithm is the curve $C(u), u \in [0,1]$, a chord error bound δ, and a tangential acceleration bound A_T. The output is the velocity curve $v(u), u \in [0,1]$ which is the solution to the optimization problem (7).*

1 Compute the centripetal acceleration bound A_N with formula (4), the CEVLC in (10), and its key points as shown in Section 2.2..

2 From the start point $(u, v(u)) = (0, 0)$, compute the forward A_T integration trajectory v_s. Compute the first intersection point $(u_l, v_s(u_l))$ of v_s and the CEVLC. If there exist no intersections, denote $u_l = 1$.

3 From the end point $(u, v(u)) = (1, 0)$, compute the backward A_T integration trajectory v_e. Compute the first intersection point $(u_r, v_e(u_r))$ of v_e and the CEVLC. If there exist no intersections, 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

$$v(u) = \begin{cases} v_s, 0 \leq u \leq u_i \\ v_e, u_i < u \leq 1. \end{cases} \tag{13}$$

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

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

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

(5.3) If $v^+_{lim}(u_l) \geq v_s(u_l) > v^-_{lim}(u_l)$, 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. Consider three cases:

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

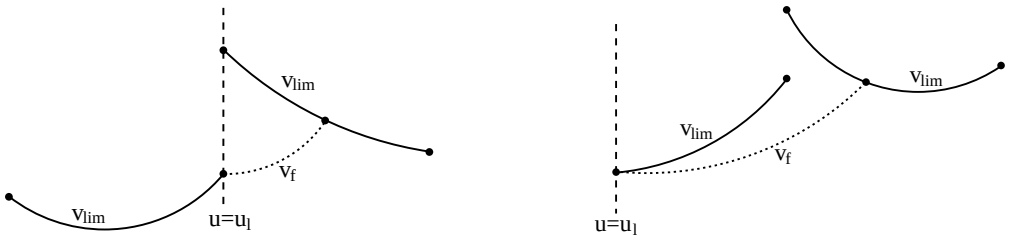
(6.2) If $a^-_{lim}(u_l) \geq A_T$, goto step 8.

(6.3) If $a^-_{lim}(u_l) \leq -A_T$, 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, the CEVLC over the interval (u_l, u_n) is feasible. Update v_s to be

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

Let $u_l = u_n$, goto step 4.



(a) From step (5.2). The current point is discontinuous and forward integration is possible

(b) From step (6.2). The current point is continuous and forward integration is possible

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

8 Starting from $(u_l, v_s(u_l))$, compute the forward A_T integration trajectory v_f. Find the first intersection point $(u_i, v_f(u_i))$ of v_f and the CEVLC (Fig. 2). If there exist no intersections, set $u_l = 1$. Update v_s to be

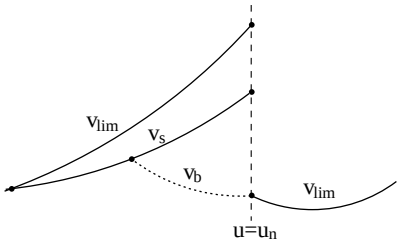
$$v_s(u) = \begin{cases} v_s(u), & 0 \leq u < u_l \\ v_f(u), & u_l \leq u \leq u_i. \end{cases}$$

Let $u_l = u_i$, goto step 4.

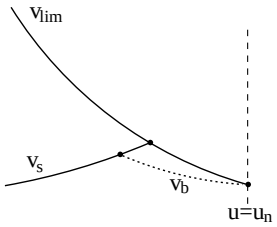
9 Starting from point $(u_n, v_{lim}(u_n))$, compute the backward A_T integration trajectory v_b in the $-u$ direction. Find the intersection point[2)] $(u_i, v_b(u_i))$ of v_b and v_s (Fig. 3). Update v_s to be

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

[2)] We will show that there exists a unique intersection point in Lemma 6.4.



(a) From step (5.3). The current point is discontinuous. Forward integration is impossible and backward integration is needed



(b) From step (6.3). The current point is continuous. Forward integration is impossible and backward integration is needed

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

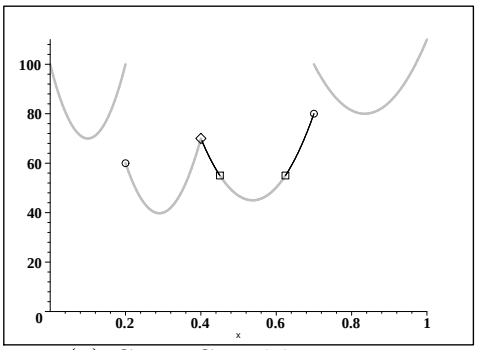
Let $u_l = u_n$, goto step 4.

The following theorem shows that the proposed algorithm computes the unique solution to the optimization problem (7). The proof of this theorem will be given the the appendix of this paper. Further improvements of the algorithm are given in Section 2.5..

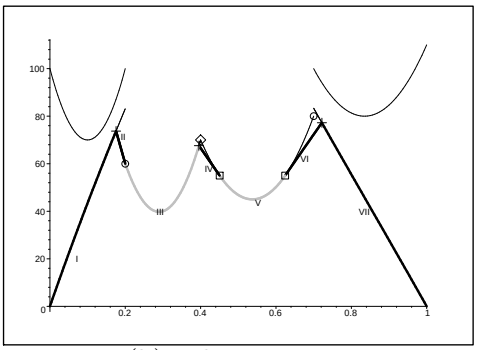
Theorem 2.3 *The velocity curve computed with Algorithm* **VP_CETA** *is the velocity curve defined in equation (12) and is the only solution to the optimization problem (7). More precisely, we will show that the velocity curve will reach its maximal possible value at every point of the tool path under the given constraints.*

As a consequence of the above theorem, we can see that the "Bang-Bang" control strategy is a necessary way to achieve time-optimality to the velocity planning problem under the given constraints.

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. The horizontal axis is the parameter of the curve $C(u)$. The vertical axis is the velocity.

Fig. 4 is an illustrative example of the algorithm. We first compute the CEVLC v_{lim} and the key points as shown in Fig. 4(a), where $\circ$ represents the first type key points and the corresponding parameters are $0.2, 0.7$; the $\diamond$ represents the second type key point, and

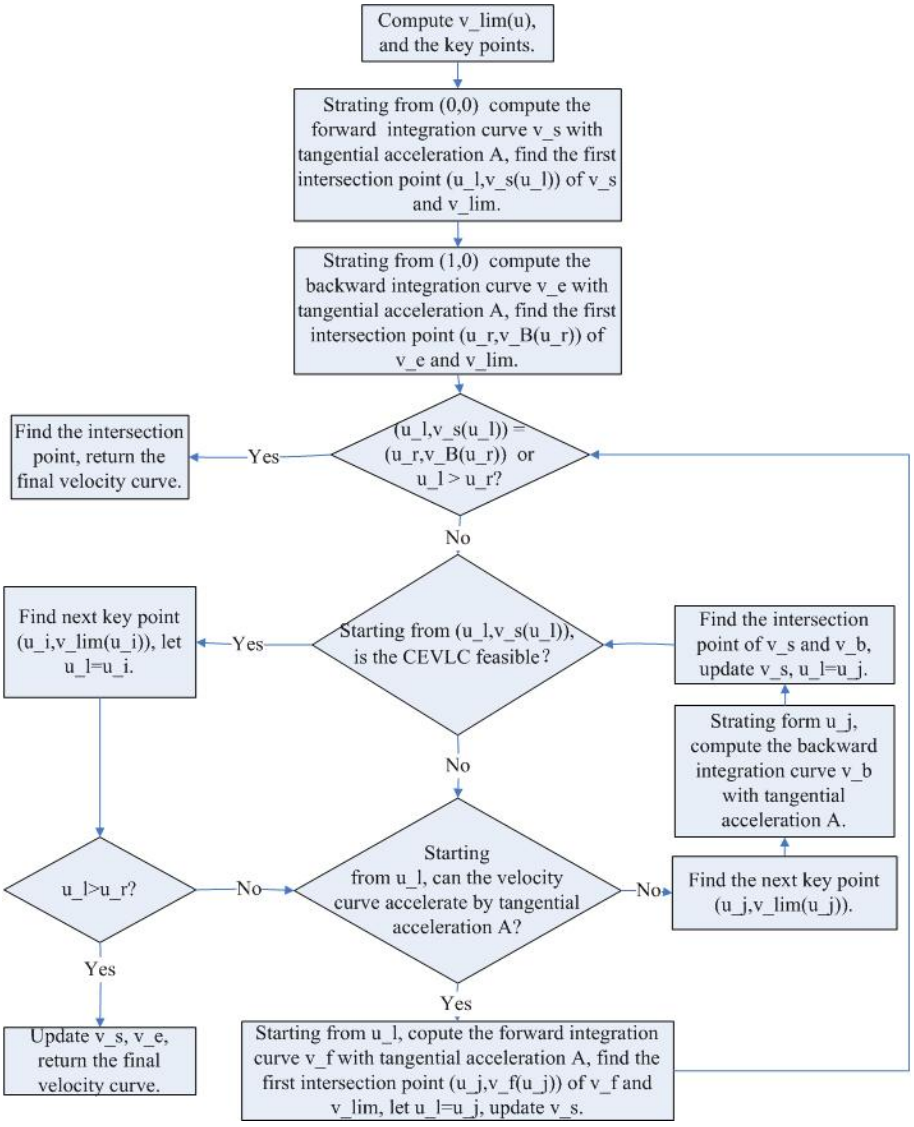


Fig. 5. Flow chart of the velocity planning

the corresponding parameter is 0.4; and the □ represents the third type key point. The gray parts are feasible segments.

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 (5.3) is executed. In step 9, starting from the first ∘ 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 Fig. 4(b).

From the first point marked by ∘, 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 ⋄ be the current point. From the current point, the CEVLC is not feasible and case (6.3) is executed. In step 9, we select the next key point which is the first point marked by □. 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 $\mathbf{I}, \mathbf{II}, \mathbf{III}, \mathbf{IV}$.

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

Let the second point marked by □ to be the current point. Starting from this point, the CEVLC is not feasible and case (6.2) 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 $\mathbf{VII}$.

Note that the CEVLC can be parts of the final velocity curve quite often, while in [21, 22], the VLC is rarely a part of the final velocity curve.

Remark 2.4 In Algorithm **VP_CETA**, 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 4., we will show how to give explicit formulas for the integration curve for two types of simple tool pathes.

2.5. 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 need the following properties of the CEVLC, the proofs of which are given in the appendix as Lemma 6.3 and Lemma 6.2 respectively.

Proposition 2.5 *Let $(u_l, v_{lim}(u_l))$ and $(u_n, v_{lim}(u_n))$ be two adjacent key points of the CEVLC. Then an A_T or $-A_T$ integration trajectory can intersect the curve segment $v_{lim}(u)$, $u \in (u_l, u_n)$ once at most.*

Proposition 2.6 *Let $v_1(u)$ and $v_2(u)$ be two velocity curves for the tool path $C(u)$ defined on $[u_1, u_2]$ and $a_{1T}(u), a_{2T}(u)$ their tangential accelerations respectively. If $v_1(u_1) \le v_2(u_1)$ and $a_{1T}(u) \le a_{2T}(u)$ for $u \in [u_1, u_2]$, then $v_1(u) \le v_2(u)$ 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 $u \in [u_1, u_2]$.*

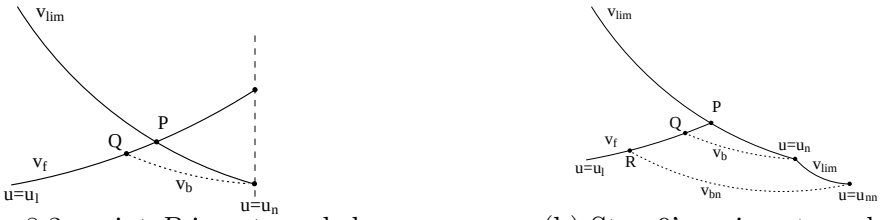
We will propose three improvements which are summarized as three remarks below.

Remark 2.7 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_T in the $+u$ direction. We can modify step 8 as follows:

8.1 If $v_f(u_n) < v_{lim}(u_n)$, by Proposition 2.5, 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 $v_f(u_n) \geq v_{lim}(u_n)$.

8.2 If $v^+_{lim}(u_n) \geq v_f(u_n) \geq v^-_{lim}(u_n)$ or $v^+_{lim}(u_n) = v_f(u_n) \leq v^-_{lim}(u_n)$, 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 Proposition 2.5. 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. 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 Algorithm **VP_CETA** can be modified similarly as step 8.

Remark 2.8 Step 9 can be simplified as follows. We call a parameter u_n *useless* if $v^+_{lim}(u_n) \geq v^-_{lim}(u_n)$, $a^-_{lim}(u_n) \leq -A_T$, and the next CEVLC segment is not feasible. Note that the second and third conditions mentioned above are equivalent to the following condition: $a_{lim}(u) < -A_T, u \in (u_n, u_{nn})$, 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 Proposition 2.6 (Fig. 6(b)), and as a consequence v_b will not be a part of the final velocity curve due to Theorem 2.1. 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 procedure until either $u_n = 1$ or u_n is not useless. Use this u_n to compute the backward integration trajectory v_b and update v_s.

Remark 2.9 Due to Theorem 2.1 and Proposition 2.6, 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 plus the start and end 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 A_T integration trajectories starting from point P respectively, which can be computed with the methods in Remark 2.7. 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 2.1 and Proposition 2.6.

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 is not processed and has the smallest velocity.

2.6. Feedrate override in CNC-machining

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

Algorithm 2.10 (Interpolation algorithm) *The input is the current parameter u_i, the velocity curve $v(u)$, maximum feedrate v_{max}, and the sampling time T. The output is the parameter of the next interpolation point u_{i+1}.*

1. According to the velocity curve and the maximum feedrate, let $v_i = \min(v(u_i), v_{max})$. 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} wit the method given in ([4, 23]).

Since for any parametric value u, whenever the value of $v^*(u)$ is taken from $v(u)$ or v_{max}, the left and right limitations of the tangential acceleration are satisfied. Hence, $v^*(u)$ satisfies the maximum feedrate, chord error, and tangential acceleration bounds. Furthermore, in CNC-machining, the users can change the maximum feedrate during manufacturing, which is called *feedrate override*. Although the new feedrate 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 feedrate override problem efficiently.

Algorithm 2.11 (Feedrate override) *The input is the velocity curve $v(u)$, the current parameter u_i, the current feedrate v_*, the modified feedrate limitation $\bar{v}_{max}$. The output is the parametric values $u_{i+1}, u_{i+2}, \ldots$ of the interpolation points.*

1 If $v_* > \bar{v}_{max}$, then let $\bar{v}_i = v_* - A_T T$, $\check{v}_i = \max(\bar{v}_i, \bar{v}_{max})$, $v_i = \min(\check{v}_i, v(u_i))$. Find the next interpolation point u_{i+1} according to Algorithm 2.10, $i = i + 1, v_* = v_i$, and repeat step 1.

2 If $v_* \leq \bar{v}_{max}$, then let $v_i = \min(v(u_i), \bar{v}_{max}, v_* + A_T T)$. Find the next interpolation point u_{i+1} according to Algorithm 2.10. 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 feedrate as soon as possible when the current feedrate is larger than the modified feedrate. Step 2 of the above algorithm is exactly Algorithm 2.10, where the maximum feedrate is replaced by the modified feedrate.

One advantage of using tangential acceleration is that feedrate override can be carried out easily. In the case of multi-axis acceleration mode, when the maximal feedrate is changed to v_{max}, we cannot simply take $v^*(u) = \min(v(u), v_{max})$ to be the new velocity curve and the procedure to compute the new velocity curve is complicated.

3. Velocity planning with chord error and jerk bounds

In this section, we consider the velocity planning under a chord error bound δ and a jerk bound J. We are able to give a greedy velocity planning algorithm for this problem.

By (6), the jerk of a velocity curve $v(u)$ for the tool path $C(u), u = 0..1$ is

$$j_T(u) = \frac{\mathrm{d}a_T(u)}{\mathrm{d}t} = \frac{\mathrm{d}a_T(u)}{\mathrm{d}u}\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{v}{\sigma}(\frac{vv'}{\sigma})'. \tag{14}$$

Then the velocity planning problem is to find a velocity curve $v(u), u = 0..1$, such that

$$\min_{v(u)} t = \int_0^1 \frac{\sigma(u)}{v(u)}\mathrm{d}u, \tag{15}$$

under the following constraints

$$|j_T(u)| \leq J, a_N(u) \leq A_N, u = 0..1, \tag{16}$$

where J is the jerk bound and A_N is the centripetal acceleration bound computed from the chord error bound with formula (4).

Similar to the method given in Section 2.2 of [27], we can show that a solution to the time-optimal problem (15) must satisfy the "Bang-Bang" control strategy. That is, the velocity curve is governed either by the jerk bound or by the chord error bound. Since the CEVLC defined in Section 2.3. is determined by the chord error bound, all we need to do is to compute a velocity curve governed by the jerk bound, which is "under" the CEVLC.

3.1. Key points of the CEVLC related to the jerk bound

Similar to Section 2.2., we also need to consider the switching points or key points of the CEVLC w.r.t. the jerk bound.

Let $a_{lim}(u)$ be the tangential acceleration of the CEVLC. If the CEVLC is not differentiable at u, we use the left and right limitations to define $a^+_{lim}(u), a^-_{lim}(u)$. There exist five types of key points.

The first and second types of key points are the same as that given in Section 2.2.. These are the connecting points of two adjacent segments of $C(u)$.

Since we also consider the jerk value of the CEVLC, the points with non-differentiable tangential accelerations on the CEVLC are also selected as switching points. So, the third type switching points are the continuous but non-differentiable points of $a_{lim}(u)$.

For a differentiable segment of the CEVLC divided by the above three types of switching points, we can divide it according to whether the jerk of the CEVLC is $\pm J$, where J is the jerk bound. A point on the CEVLC is called a fourth type key point if the jerk along the CEVLC at this point is $\pm J$. We can find the fourth type switching points by solving the following algebraic equation in u

$$j_{lim}(u) = \frac{v_{lim}(u)}{\sigma(u)}\left(\frac{v_{lim}(u)v'_{lim}(u)}{\sigma(u)}\right)' = \frac{\sqrt{A_N/k(u)}}{\sigma(u)}\left(\frac{(A_N/k(u))'}{2\sigma(u)}\right)' = \pm J. \tag{17}$$

The fifth type switching points are the velocity extremal points, where the velocity reaches a local extremal value. These points can be computed by solving the following algebraic equation in u

$$a_{lim}(u) = \frac{v_{lim}(u)v'_{lim}(u)}{\sigma(u)} = 0. \tag{18}$$

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

1. A curve segment is called *jerk feasible* if the absolution values of jerk at all points are bounded by J. A jerk feasible CEVLC segment can be a part of the final velocity curve.

2. A curve segment is called *unfeasible* if the absolution values of jerk at all points are larger than J. 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 third type key point are both jerk feasible, and they have the same acceleration value at that point, we can delete this switching point since it does not affect velocity planning. If a fifth type switching point is on a feasible segment, we can also delete this point.

3.2. Integration curve with a given jerk

In this section, we will derive the velocity curve when the jerk reaches its bound J. If the velocity curve is governed by the jerk bound J, from (14), we have

$$\frac{v}{\sigma}(\frac{vv'}{\sigma})' = J. \tag{19}$$

That is, we need to solve the above second order differential equation to obtain $v(u)$. Let $\pi = \int \sigma \mathrm{d}u$ and $g = \frac{\mathrm{d}v}{\mathrm{d}\pi} = \frac{v'}{\sigma}$. Then, (19) becomes

$$\frac{v}{\sigma}(\frac{vv'}{\sigma})' = \frac{v}{\sigma}(vg)' = \frac{v}{\sigma}(v'g + vg') = v(g^2 + v\frac{\mathrm{d}g}{\mathrm{d}\pi}) = vg^2 + v^2 g\frac{\mathrm{d}g}{\mathrm{d}v} = J. \tag{20}$$

Let $h = g^2$. Then, (20) becomes

$$\frac{\mathrm{d}h}{\mathrm{d}v} = \frac{2J}{v^2} - \frac{2h}{v}. \tag{21}$$

Solving the above differential equation in h, we have

$$h = \frac{2J}{v} - \frac{c_1}{v^2}, \tag{22}$$

where c_1 is an integration constant. So, we have

$$\frac{\mathrm{d}v}{\mathrm{d}\pi} = \pm\frac{\sqrt{2Jv - c_1}}{v}. \tag{23}$$

Solving the above equation, we have

$$\pi - c_2 = \pm\int \frac{v\mathrm{d}v}{\sqrt{2Jv - c_1}} = \pm\frac{(Jv + c_1)\sqrt{2JV - c_1}}{3J^2}, \tag{24}$$

where c_2 is another integration constant. Solving this algebraic equation in v, we have

$$v = \frac{1}{2J}\left[\omega\left(U + \sqrt{U^2 + c_1^3}\right)^{\frac{2}{3}} + \omega^2\left(U + \sqrt{U^2 + c_1^3}\right)^{\frac{2}{3}} - c_1\right], \tag{25}$$

where $U = 3J^2(\pi - c_2), \omega^3 = 1$.

Now we give the expressions for computing the integration constants c_1, c_2. From equations (22) and (24), we have

$$\begin{aligned} c_1 &= 2Jv - (vg)^2 = 2Jv - (\tfrac{vv'}{\sigma})^2 = 2Jv - a_T^2, \\ c_2 &= \pi \mp \tfrac{(Jv+c_1)\sqrt{2JV-c_1}}{3J^2} = \pi \mp \tfrac{(3Jv-a_T^2)|a_T|}{3J^2} = \pi - \tfrac{(3Jv-a_T^2)a_T}{3J^2}. \end{aligned} \tag{26}$$

The constants c_1, c_2 can be determined by a specific point $(u*, v(u*), a_T(u*))$ on the integration curve.

In (25), if $U^2 + c_1^3$ is negative in some value interval of u, the expression of v should be changed. We substitute ω by $e^{\frac{2}{3}ik\pi}(k = 0, 1, 2)$ to obtain

$$\begin{aligned} v &= \tfrac{-c_1}{2J}[e^{\frac{2}{3}ik\pi}(\tfrac{U}{(-c_1)^{3/2}} + i\sqrt{1 - \tfrac{U^2}{(-c_1)^3}})^{2/3} + e^{-\frac{2}{3}ik\pi}(\tfrac{U}{(-c_1)^{3/2}} - i\sqrt{1 - \tfrac{U^2}{(-c_1)^3}})^{2/3} + 1] \\ &= \tfrac{-c_1}{2J}[e^{\frac{2}{3}ik\pi}e^{\frac{2}{3}i\arccos\frac{U}{(-c_1)^{3/2}}} + e^{-\frac{2}{3}ik\pi}e^{-\frac{2}{3}i\arccos\frac{U}{(-c_1)^{3/2}}} + 1] \\ &= \tfrac{-c_1}{2J}[2\cos\tfrac{2}{3}(\arccos\tfrac{U}{(-c_1)^{3/2}} + k\pi) + 1]. \end{aligned} \tag{27}$$

The velocity curve governed by J is called the J_+ *trajectory*. If the jerk bound is $-J$, we just need to replace J by $-J$ in the above solutions. And we call the velocity curve governed by $-J$ the J_- *trajectory*.

3.3. Velocity planning with confined chord error and jerk

In this section, we will give a velocity planning algorithm which can be considered as a solution to problem (15) under a greedy rule to be explained below.

Contrary to problem (7), it is still an open problem to design a time-optimal solution to problem (15) or similar problems with jerk bounds on the x-, y-, and z-axis [27] using the continuous model. More discussion about this issue can be found in [27]. At the beginning of Section 3., we showed that a solution to problem (15) must be "Bang-Bang" in the sense that either the jerk or the chord error reaches its bound at any time. What we will do below is to design a velocity curve which satisfies the "Bang-Bang" control strategy and obeys the following "greedy rule": we will use the J_+ trajectory as much as possible. In other words, we only use the J_- trajectory to decelerate when we have to do so.

We now give the algorithm.

Algorithm 3.1 (VP_CETJ) *The input of the algorithm is the tool path $C(u), u \in [0,1]$, a chord error bound δ, and a jerk bound J. The output is the velocity curve $v(u), u \in [0,1]$ which is a solution to problem (15) under the greedy rule.*

The algorithm consists of two phases. The first phase is quite similar to Algorithm **VP_CETA** and can be obtained from Algorithm **VP_CETA** by making two changes

Firstly, we need to replace the A_T integration trajectory by the J_+ trajectory, replace $a_{lim}(u)$ by $j_{lim}(u)$, and replace feasible segments of CEVLC by jerk feasible segments of CEVLC.

Secondly, Step (6.3) need to be modified. In this case, we cannot use a backward J_+ trajectory starting from point $(u_n, v_{lim}(u_n), a_{lim}(u_n))$, since this trajectory will be above the CEVLC. See Fig. 3(b) for an illustration. The reason is that the jerk at any point in (u_l, u_n) for the CEVLC is less than $-J$, and if we use a backward J_+ trajectory v_b, then both its acceleration and speed will be larger than that of the CEVLC at a small neighborhood of u_n. In Algorithm **VP_CETA**, using a backward A_T trajectory is possible, because the acceleration of the CEVLC in the backward direction at any point in (u_l, u_n) is larger than A_T. As a consequence, the backward A_T trajectory will be below the CEVLC. A rigorous proof of this fact can be found in the appendix of the paper.

We will modify Step (6.3) as follows. Due to the above analysis, what we need to do is to lower the start acceleration a_n at $u = u_n$ such that there exists a backward J_+ trajectory $v_b(u)$ which passes through $(u_n, v_{lim}(u_n), a_n)$ and intersects with v_s (Fig. 3(b)). Let u_1 be the parameter value for the intersection of v_b and the trajectory v_s. From (26), the integration constants of v_b can be expressed as $c_1(u, v(u), a_T(u)), c_2(u, v(u), a_T(u))$. Since v_b has the same velocity and acceleration with $v_s(u)$ at $u = u_1$ and c_1, c_2 are constants on v_b, we have the following equations

$$\begin{cases} c_1(u_1, v_s(u_1), a_s(u_1)) = c_1(u_n, v_{lim}(u_n), a_n), \\ c_2(u_1, v_s(u_1), a_s(u_1)) = c_2(u_n, v_{lim}(u_n), a_n) \end{cases} \tag{28}$$

where $a_s(u_1)$ is the acceleration of $v_s(u)$ at $u = u_1$. We can solve the above algebraic equation system to obtain u_1 and a_n. Then the trajectory v_b can be found with (25).

The first phase of the algorithm outputs a continuous velocity curve. But, at the intersection point of two velocity curve segments, the tangential acceleration of $v(u)$ might

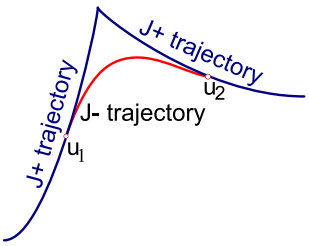


Fig. 7. Connect two velocity curve segments with a J_- trajectory (red one) to obtain a velocity curve with continuous acceleration

not be continuous. The second phase of the algorithm will connect the two velocity curve segments with J_- trajectories to obtain a velocity curve with continuous tangential accelerations (Fig. 7). Here the greedy rule is used: we now must use a $J-$ trajectory to make the connection. Two cases are considered.

Firstly, we assume that the definition interval $[u_1, u_2]$ of the J_- trajectory does not contain any connection point of the tool path $C(u)$. From (26), the integration constants of the J_- trajectory can be expressed as $c_1(u, v(u), a_T(u)), c_2(u, v(u), a_T(u))$. Let the two velocity segments be $v_l(u)$ and $v_r(u)$ with tangential accelerations $a_l(u)$ and $a_r(u)$. Since the J_- trajectory has the same velocity and acceleration with $v_l(u)$ (or $v_r(u)$) at u_1 (or u_2) and c_1, c_2 are constants on the J_- trajectory, we have the following equations (Fig. 7)

$$\begin{cases} c_1(u_1, v_l(u_1), a_l(u_1)) = c_1(u_2, v_r(u_2), a_r(u_2)), \\ c_2(u_1, v_l(u_1), a_l(u_1)) = c_2(u_2, v_r(u_2), a_r(u_2)) \end{cases} \tag{29}$$

We can solve the above algebraic equation system to obtain u_1, u_2. Then the integration constants of the J_- trajectory are $c_1(\bar{u}_1, v_l(\bar{u}_1), a_l(\bar{u}_1))$ and $c_2(\bar{u}_1, v_l(\bar{u}_1), a_l(\bar{u}_1))$, where $\bar{u}_1$ is a solution of (29). After the two integration constants are obtained, the J_- trajectory can be computed with the methods in Section 3.2..

Secondly, if the definition interval $[u_1, u_2]$ of the J_- trajectory contains one connection point of the tool path $C(u)$, say u^*. Let $\sigma_1(u), \sigma_2(u)$ be the two parametric speeds of the two segments of $C(u)$, and $\rho_1(u) = \int \sigma_1(u)\mathrm{d}u, \rho_2(u) = \int \sigma_2(u)\mathrm{d}u$. Then, from (26), to obtain the J_- trajectory, we need to solve the following algebraic equation system

$$\begin{cases} c_1(u_1, v_l(u_1), a_l(u_1)) = c_1(u_2, v_r(u_2), a_r(u_2)), \\ c_2(u_1, v_l(u_1), a_l(u_1)) - \rho_1(u_1) = c_2(u_2, v_r(u_2), a_r(u_2)) - \rho_2(u_2) \end{cases} \tag{30}$$

to obtain u_1, u_2. Then, similar as above, we obtain the connecting J_- trajectory. If the definition interval of the J_- trajectory contains several connection points of the tool path, one can obtain the J_- trajectory in a similar way.

The output of Algorithm **VP_CETJ** is the velocity curve obtained in phase two, which has confined jerk and chord error.

Remark 3.2 One can add the maximal feedrate constraint in the velocity planning just as a part of the CEVLC, and solve equation(29) to make the velocity curve satisfying the jerk and chord error bounds. We will not give the details here.

4. Experiments with quadratic B-splines and cubic PH-splines

In Algorithms **VP_CETA** and **VP_CETJ**, 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 closed form formulas for $\int \sigma du$. We implemented our algorithms in these two cases in Maple and used examples from real manufacturing parts to show the feasibility of our algorithm. We also implement Algorithm **VP_CETA** with quadratic B-splines in an industrial CNC controller and conduct real CNC machining in a three axis CNC machine.

4.1. Velocity planning for quadratic B-splines with confined acceleration

Let $C(u), u \in [0,1]$ 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:

$$\begin{aligned} C(u) &= (x(u), y(u), z(u)) \\ &= (a_0 + a_1 u + a_2 u^2, b_0 + b_1 u + b_2 u^2, c_0 + c_1 u + c_2 u^2), \end{aligned}$$

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

$$\sigma = |C'| = \sqrt{x'^2 + y'^2 + z'^2} = \sqrt{mu^2 + nu + l}.$$

Since $C(u)$ is quadratic, the parameters m, n, l can be computed as follows

$$m = 4(a_2^2 + b_2^2 + c_2^2), n = 4(a_1 a_2 + b_1 b_2 + c_1 c_2), l = a_1^2 + b_1^2 + c_1^2.$$

And

$$|C' \times C''| = \sqrt{(b_1 c_2 - c_1 b_2)^2 + (c_1 a_2 - a_1 c_2)^2 + (a_1 b_2 - b_1 * a_2)^2}$$

is a constant. Hence the CEVLC is

$$q = v^2 = \frac{A_N}{|k(u)|} = \frac{A_N \sigma^3}{|C' \times C''|} = D\sigma^3 = D(mu^2 + nu + l)^{3/2}, \tag{31}$$

where D is a constant. The tangential acceleration along the CEVLC is $a_{lim} = \frac{(D\sigma^3)'}{2\sigma} = \frac{3}{2} D\sigma\sigma' = \frac{3}{4} D(\sigma^2)'$. Hence, $a_{lim}(u)$ is a linear function in the parameter u. Then, the key points of third type can be computed by solving linear equations $a_{lim} = \pm A_T$. 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.

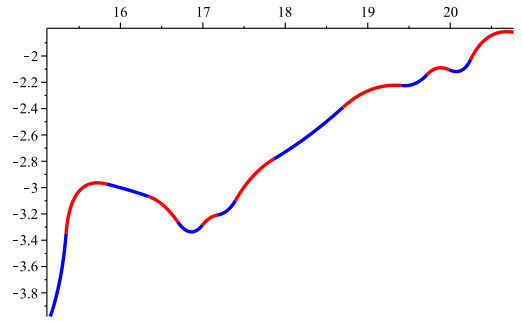
Now, we show how to compute the A_T integration trajectory. When the tangential acceleration reaches its bounds $\pm A_T$, we need to compute the solution of the differential equation:

$$q' = \pm 2 A_T \sigma. \tag{32}$$

Let

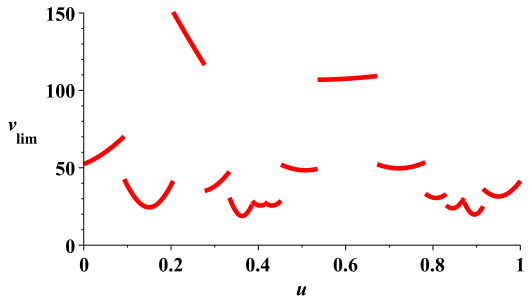
$$\begin{aligned} i(u) = \;& \pm 2 A_T \pi(u) \text{ where} \\ \pi(u) = \;& [\tfrac{1}{4} \tfrac{(2mu+n)\sqrt{mu^2+nu+l}}{m} + \tfrac{1}{2} \ln(\tfrac{\frac{1}{2}mu+n}{\sqrt{m}} + \sqrt{mu^2+nu+l}) l \tfrac{1}{\sqrt{m}}] \\ & - \tfrac{1}{8} \ln(\tfrac{\frac{1}{2}mu+n}{\sqrt{m}} + \sqrt{mu^2+nu+l}) n^2 m^{-\frac{3}{2}}. \end{aligned} \tag{33}$$

Then, $q(u) = i(u) - i(u^*) + q(u^*)$ is the solution of the differential equation (32) with initial value $(u^*, q(u^*))$.

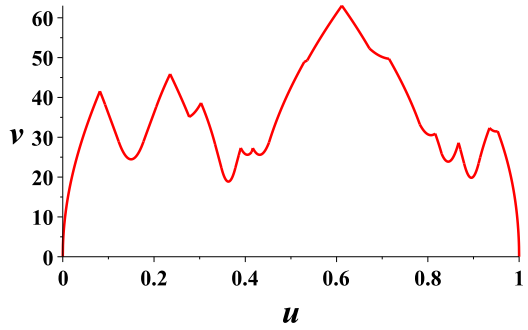


(a) A quadratic B-spline in x, y plane

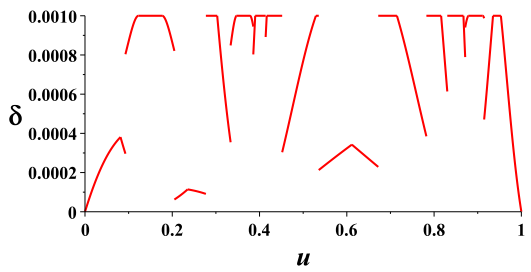
(b) The vase model



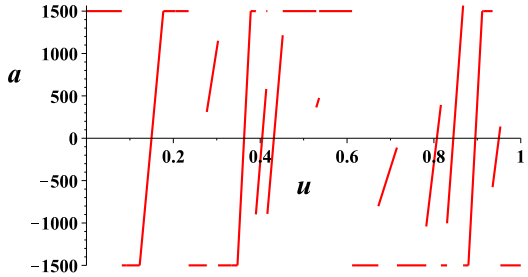
(c) The CEVLC $v_{lim}(u)$



(d) Velocity curve $v(u)$



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



(f) Tangential acceleration $a_T(u)$

Fig. 8. Optimal velocity planing for a quadratic B-spline with confined chord error and acceleration. Except (a) and (b), the horizontal axis is the parameter of $C(u)$. The units for the velocity, acceleration, and chord error are mm/s, mm/s^2, and mm respectively.

We use an example to illustrate the algorithm. The curve in Fig. 8(a) is a planar quadratic B-spline $(x(u), y(u)), u \in [0, 1]$ consisting of 14 pieces of quadratic curve segments, which is from the tool path of the vase in Fig. 8(b). We set the tangential acceleration and chord error bounds to be $A_T = 1500\ mm/s^2$ and $\delta = 1\mu m$. If the sampling period is $T = 2ms$, then from (4), the centripetal acceptilation bound is $A_N = 2000mm/s^2$.

According to Algorithm **VP_CETA**, we first compute the CEVLC with the maximal

centripetal acceleration, which is shown in Fig. 8(c). The final velocity curve computed with Algorithm **VP_CETA** is shown in Fig. 8(d), which consists of thirty two pieces, where the $4, 8, 11, 14, 16, 21, 23, 25, 31$-th pieces are feasible CEVLC segments, and the others are controlled by the tangential acceleration. Fig. 8(e) is the chord error of the optimal velocity curve. Fig. 8(f) is its tangential acceleration. From these two figures, we can see that the control is "Bang-Bang."

The tool path in Fig. 9(a) is a complete segment $C(u) = (x(u), y(u)), u \in [0, 10]$ of the vase in Fig. 8(b) from top to bottom and consists of five quadratic B-splines with 57 quadratic curve segments and 5 long straight line segments. The tool path in Fig. 8(a) is the first quadratic spline in Fig. 9(a). We use the same acceleration and chord error bounds. Fig. 9(c) is its CEVLC. Fig. 9(d) is the optimal velocity curve computed with our algorithm, which consists of 127 curve segments. Fig. 9(e) is the chord error of the optimal velocity curve. Fig. 9(f) is its tangential acceleration.

Note that in the connection point of two quadratic B-splines, the velocity decreases to zero. This can be improved. But, we will not discuss the issue here.

We now consider a space tool path $(x(u), y(u), z(u)), u \in [0, 2]$ shown in Fig. 10(a), which is from the blade of the impeller shown in Fig. 10(b) and consists of two quadratic B-splines with 13 and 14 curve segments respectively [29]. The tangential acceleration bound is $A_T = 1500\ mm/s^2$, the chord error bound is $\delta = 1\mu m$, and the sampling period is $T = 2ms$. Then the centripetal acceptilation bound $A_N = 2000mm/s^2$ can be computed with (4). Fig. 10(c) is its CEVLC. Fig. 10(d) is the optimal velocity curve computed with our algorithm. Fig. 10(e) is the chord error of the optimal velocity curve. Fig. 10(f) is its tangential acceleration.

4.2. Velocity planning for cubic PH-splines with confined acceleration

Let $C(u), u \in [0, 1]$ be a cubic PH-spline. Since a cubic PH-spline only has C^1 continuity, the connection points of the PH-splines 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]:

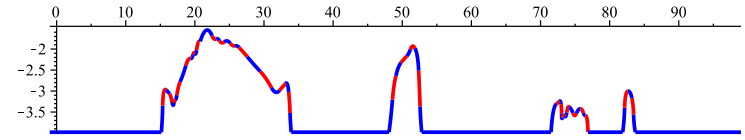
$$\begin{aligned} r'(u) =& (f(u)^2 + g(u)^2 - m(u)^2 - n(u)^2, \\ & 2(f(u)n(u) + g(u)m(u)), \\ & 2(g(u)n(u) - f(u)m(u))). \end{aligned} \tag{34}$$

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

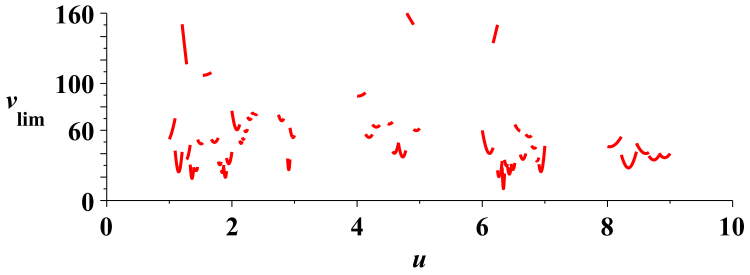
The curvature of $r(u)$ is $k(u) = \frac{|r' \times r''|}{\sigma^3} = \frac{E}{\sigma^2}$, where $\sigma = |r'| = f(u)^2 + g(u)^2 + m(u)^2 + n(u)^2$ and E is a constant. The CEVLC of $r(u)$ is

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

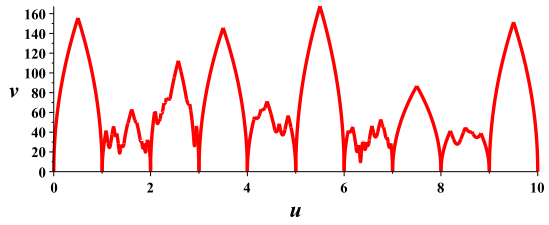
where G is a constant. The tangential acceleration along the CEVLC is: $a_{lim} = \frac{(G\sigma^2)'}{2\sigma} = G\sigma'$. Since σ is of degree two, $a_{lim}(u)$ is a linear function in the parameter u. The key points of the third type can be computed by solving linear equations $a_{lim} = \pm A_T$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



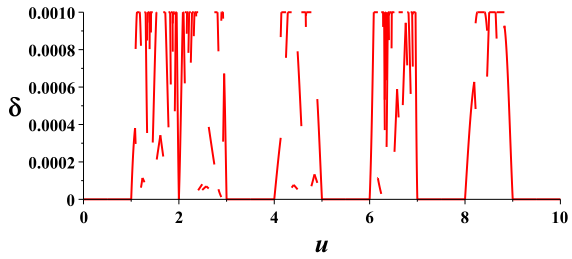
(a) quadratic B-splines $r(u) = (x(u), y(u))$



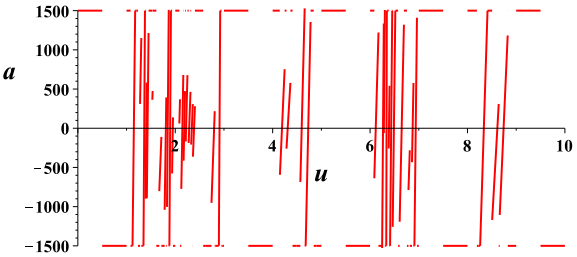
(b) Velocity limited curve $v_{lim}(u)$



(c) Velocity curve $v(u)$

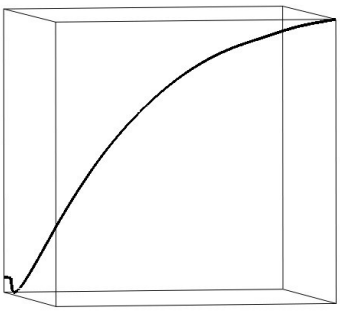


(d) Chord error $\delta(u)$

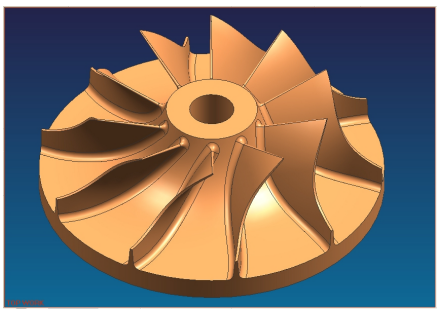


(e) Tangential acceleration $a_T(u)$

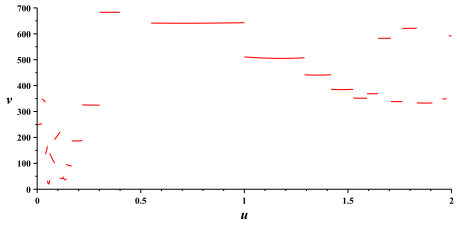
Fig. 9. Optimal velocity planing for quadratic B-splines from a vase with confined chord error and acceleration. Units are the same as that of Fig. 8.



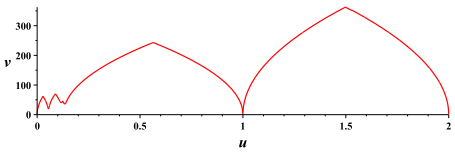
(a) A quadratic B-spline



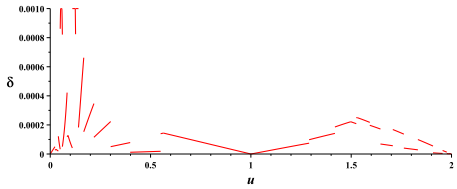
(b) The blade



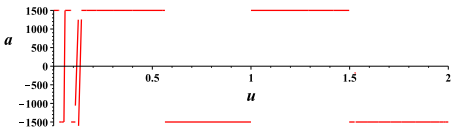
(c) The CEVLC



(d) Velocity curve $v(u)$



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



(f) Tangential acceleration

Fig. 10. Optimal velocity planing for quadratic B-splines with confined chord error and acceleration. Units are the same as that of Fig. 8.

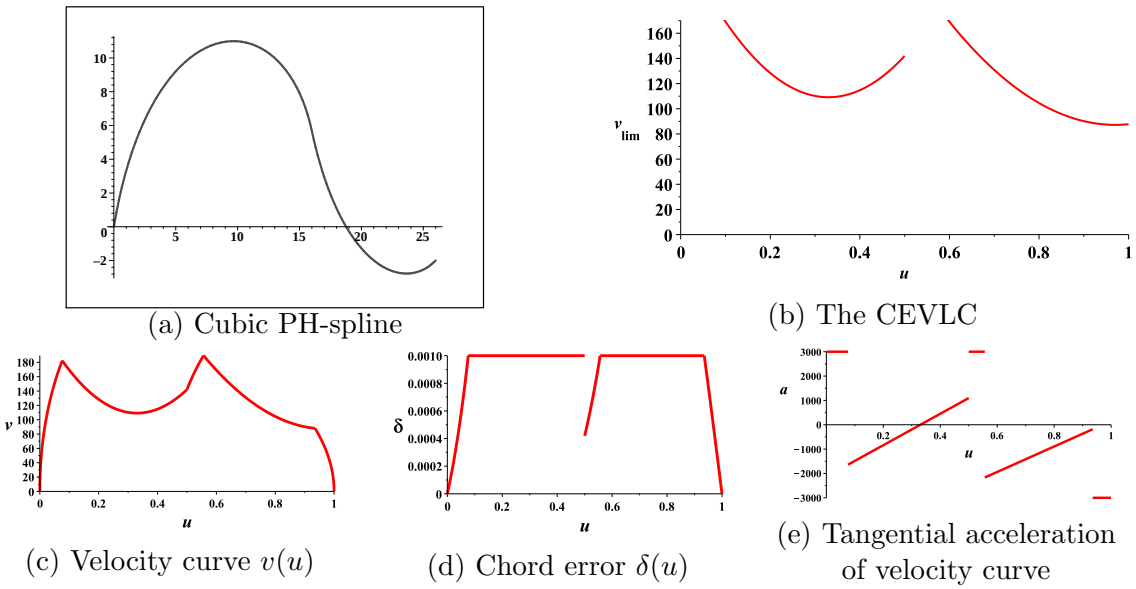
For a cubic PH-curve, the A_T integration trajectory is a polynomial in u with degree three. The integration trajectory is the solution of the following differential equation:

$$q' = \pm 2A_T\sigma = \pm 2A_T(au^2 + bu + c), \tag{35}$$

where a, b, c are constants. Let $i(u) = \pm 2A_T(1/3\,au^3 + 1/2\,bu^2 + cu)$. Then, $q(u) = i(u) - i(u^*) + q(u^*)$ is the solution of the differential equation (35) with initial value $(u^*, q(u^*))$.

Now, we give an illustrative example. The curve $C(u) = (x(u), y(u)), u \in [0, 1]$ shown in Fig. 11(a) is a cubic PH-spline consisting of two pieces of PH-curves with C^1 continuity. The tangential acceleration bound is $A_T = 3000\ mm/s^2$, the chord error bound is $\delta = 1\mu m$, and the sampling period is $T = 2ms$. Then the centripetal acceptilation bound $A_N = 2000mm/s^2$ can be computed with (4). The CEVLC is shown in Fig. 11(b) and the final velocity curve shown in Fig. 11(c) has four segments, where the $1, 3, 5$-th segments are A_T integration trajectories and the $2, 4$-th pieces are feasible CEVLC segments. The chord error and the tangential acceleration of the optimal velocity curve are given in Fig. 11(d) and Fig. 11(e) respectively.

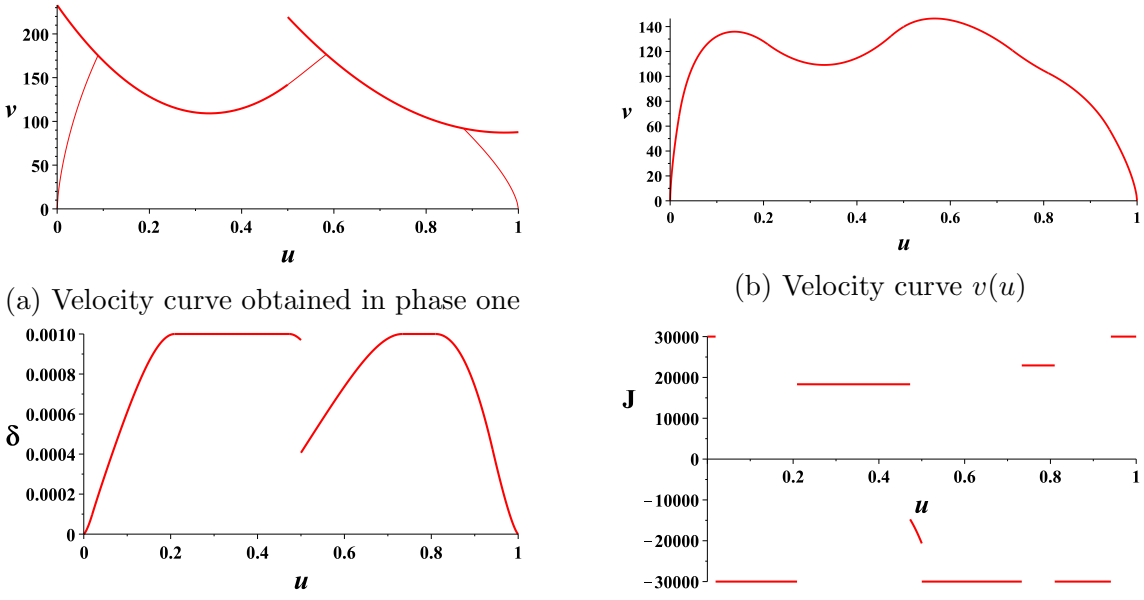
4.3. Velocity planning for cubic PH-splines with confined jerk



(a) Cubic PH-spline (b) The CEVLC

(c) Velocity curve $v(u)$ (d) Chord error $\delta(u)$ (e) Tangential acceleration of velocity curve

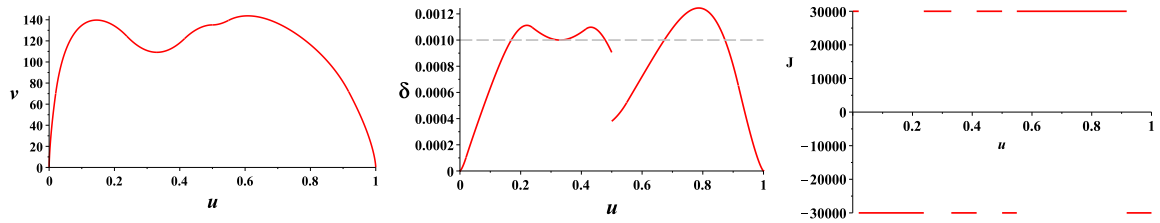
Fig. 11. Optimal velocity planning for a cubic PH-spline with confined chord error and acceleration. Units are the same as that of Fig. 8.



(a) Velocity curve obtained in phase one (b) Velocity curve $v(u)$

(c) The chord error $\delta(u)$ (d) The jerk of $v(u)$

Fig. 12. Velocity planning for a cubic PH-spline with a jerk bound. Units are the same as that of Fig. 8. The unit for the jerk is mm/s^3.



(a) Velocity curve $v(u)$ (b) The chord error $\delta(u)$ (c) The jerk of $v(u)$

Fig. 13. Velocity planning for a cubic PH-spline with sensitive corner method. The dash line in (b) is the chord error bound $\delta = 1\mu$m. We can see that the jerk is confined, but the chord error is beyond the given precision in about 50% of the points.

Let $C(u), u \in [0, 1]$ be a cubic PH-spline. Similar to Section 4.2., 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 a cubic PH-curve $C(u)$ of form (4.2.). Then the jerk of the CEVLC is

$$j_{lim}(u) = \frac{v_{lim}}{\sigma} a'_{lim} = G^{3/2}\sigma'',$$

which is a constant. Then there generally exist no key points of types three and four. The key points of the fifth type can be computed by solving linear equations $a_{lim} = 0$ directly.

Now, we use the cubic PH-spline in Fig. 11(a) to illustrate the Algorithm **VP_CETJ**.

Let the jerk and the chord error bounds be $J = 30000mm/s^3$ and $1\mu m$ respectively. If the sampling period is $T = 2ms$, then the centripetal acceptilation bound is $A_N = 2000mm/s^2$. The CEVLC is the same as Fig. 11(b). Fig. 12(a) is the velocity curve obtained with the first phase of Algorithm **VP_CETJ**. After connecting the adjacent curve segments of this velocity curve by J_- trajectories in the second phase of Algorithm **VP_CETJ**, we obtain the final velocity curve $v(u)$ as shown in Fig. 12(b), which has continuous tangential accelerations. The chord error and the jerk of $v(u)$ are shown in Figs. 12(c) and (d) respectively.

The total machining time for the PH-spline 11(a) with chord error bound $1\mu m$ and tangential acceleration bound $3000mm/s^2$ using Algorithm **VP_CETA** is $0.382s$. The total machining time for the same curve segment with the same chord error bound and jerk bound $30000mm/s^3$ using Algorithm **VP_CETJ** is $0.44s$. So, when we use jerk limitations, the machining time is longer, as expected.

We compare Algorithm **VP_CETJ** with another velocity planning method in Fig. 13. Fig. 13(a) is the velocity curve $v(u)$ for the cubic PH-spline in Fig. 11(a) under the same bounds obtained with a method of detecting the limit speeds at sensitive corners. The method is similar to that given in [26]. Firstly, we detect the local minimal velocity for the CEVLC and the connection points for the spline, in this example, the parametric value of these sensitive points are $u = 0, 0.33, 0.5, 0.97, 1$. Then compute the length between every two adjacent points, and use a backtracking method with jerk bound to delete the useless sensitive points and adjust the velocity of these sensitive points. In this example, the point corresponding to $u = 0, 0.33, 0.5, 1$ is useful. Then between every two adjacent points, use the starting and ending speeds, the curve length between these two points, and the jerk bound to compute the speed for each point between these two adjacent points. Fig. 13(b) is the chord error of $v(u)$, from which we can see that the chord error is beyond the bound

at about 50% of the parametric values. Furthermore, the machining time is $0.432s$ which is longer than the machining time $0.382s$ using Algorithm **VP_CETJ**.

4.4. Real CNC machining with Algorithm VP_CETA

In this section, we will show how to implement Algorithm **VP_CETA** for tool pathes described with quadratic B-splines in a commercial CNC controller and give the results for real CNC machining in a three axis industrial CNC machine.

To implement our method in CNC controllers, we compute the velocity curves off-line first and then use the velocity curves as parts of the input to the CNC controllers to achieve real-time interpolation. This strategy is adopted by many existing work such as [9, 10, 29].

We first design a new G-code which is of the following form:

$$\begin{array}{lllllll} \text{G65.5} & a_0 & a_1 & a_2 & & & \\ & b_0 & b_1 & b_2 & & & \\ & c_0 & c_1 & c_2 & & & \\ & \text{flag} & k & m & n & l & \\ & c^* & & & & & \\ & u_1 & u_2 & & & & \end{array} \tag{36}$$

When the interpreter reads a G65.5 code, the parameters are interpreted as follows. The tool path is represented by the curve segment $C(u) = (x(u), y(u), z(u)), u \in [u_1, u_2]$, where

$$x(u) = a_0 + a_1 u + a_2 u^2, y(u) = b_0 + b_1 u + b_2 u^2, z(u) = c_0 + c_1 u + c_2 u^2.$$

The corresponding velocity curve at point $C(u)$ is $v(u), u \in [u_1, u_2]$, where

$$v(u) = \begin{cases} \sqrt{k\pi(u) + c^*}, & \text{if flag} = 0 \\ \sqrt{\frac{k(mu^2+nu+l)^{3/2}}{c^*}}, & \text{if flag} = 1 \end{cases} \tag{37}$$

In the above equations, if flag $= 0$, $k = \pm 2A_T$ and $\pi(u)$ is from (33), and c^* is the integration constant. In this case, $v(u)$ is the integration trajectory controlled by $k = \pm A_T$. If flag $= 1$, $k = A_N$, $c^* = |C'(u) \times C''(u)|$, and $v(u)$ is the CEVLC of $C(u), u \in [u_1, u_2]$ given in (31).

Since we know the velocity $v(u)$ at point $C(u)$, we can use the interpolation Algorithm 2.10 to do real-time interpolation.

(A) The CNC-Controller

(b) The CNC machine

(c) Manufacturing process

Fig. 14. CNC machining of the vase

The CNC controller used in our experiment is an LT-CNC controller shown in Fig. (14)(a), which is a commercial product of Shenyang LanTian CNC Corporation. The controller is based on Linux OS and is implemented with C language. Therefore, we also implement Algorithm **VP_CETA** with C language. The CNC machine used in our experiment is shown in Fig. 14(b), which is a 3-axis industrial CNC machine. Fig. 14(c) shows the manufacturing process.

The model manufactured in the experiment is the vase shown in Fig. 8(b) and Fig. 15. The vase is a CNC model consists of more than 116000 G01 codes with total tool path length of 20.25m. Its B-spline representation consists of more than 30000 quadratic curve segments [29]. The G01 codes and the spline representation of the model can be found in http://www.mmrc.iss.ac.cn/˜xgao/cnc/. The tool path of the vase fluctuates violently making the optimal speed velocity planning difficult. Also, the examples given in Fig. 8 and Fig. 9 are both tool pathes from the vase model, which give an intuitive view of the velocity curves used in the manufacturing process.

Six machined vases are shown in Fig. 15 with different tangential acceleration and chord error bounds. The machining times for these parameters are given in Table 1. The sampling period is $T = 2ms$. From Table 1, we can see that the machining time for larger A_T is shorter, which is expected. Also, the machining time for larger chord error δ is generally shorter. From Fig. 15, we can see that the machining quality is better for smaller chord error δ with the same A_T, as expected.

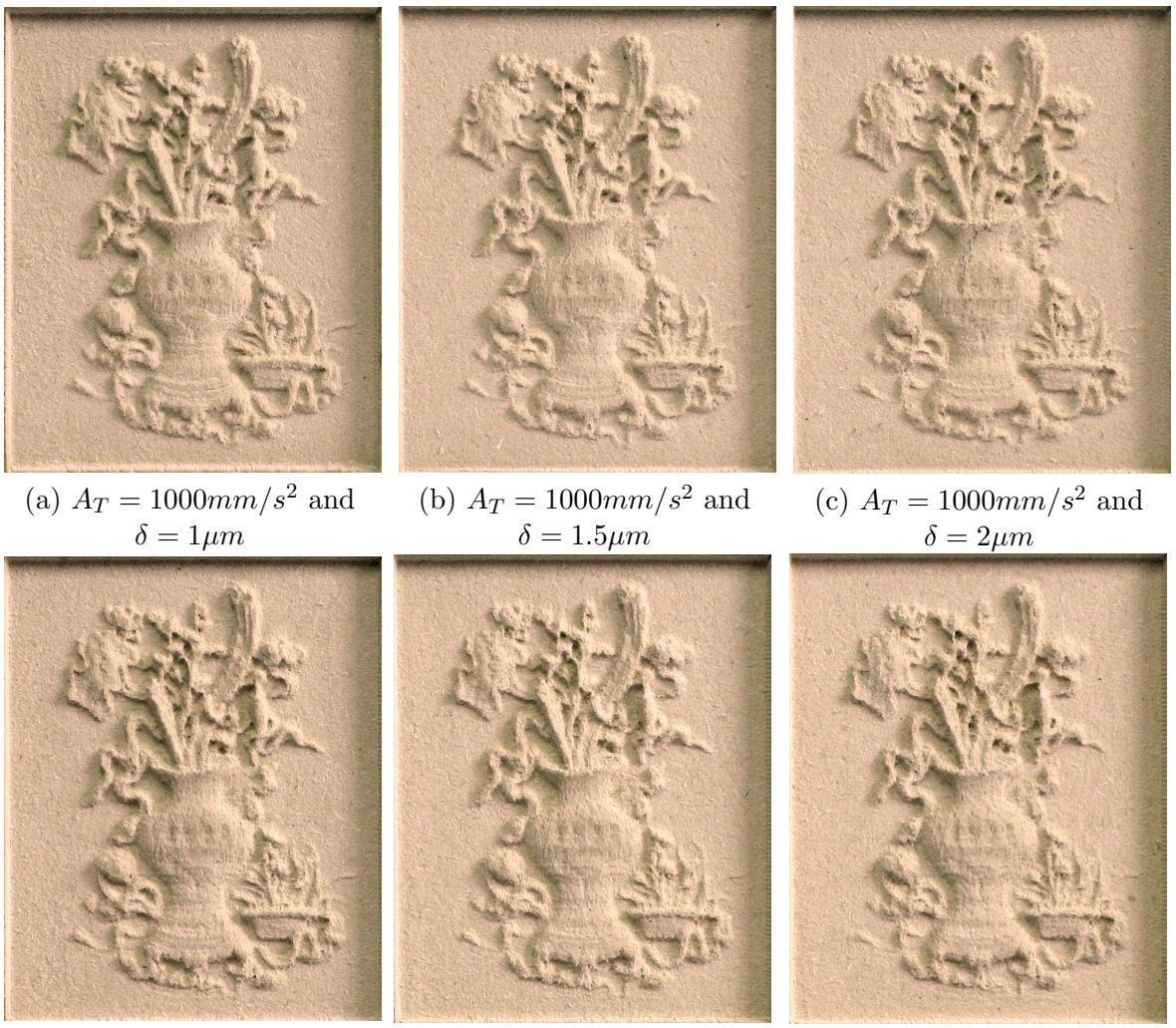
$A_T(mm/s^2)$	$\delta(mm)$	machining time
1000	0.001	$23'33''$
1000	0.0015	$22'48''$
1000	0.002	$22'25''$
2000	0.001	$18'52''$
2000	0.0015	$17'54''$
2000	0.002	$17'23''$

Table 1. The machining time of the vase

5. Conclusion

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

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 trajectories starting from the key points of the CEVLC, the start point, and the end point. We also give a practical algorithm to compute the time-optimal velocity curve and implemented the algorithm for two types of simple tool pathes. For quadratic B-splines



(a) $A_T = 1000mm/s^2$ and $\delta = 1\mu m$ (b) $A_T = 1000mm/s^2$ and $\delta = 1.5\mu m$ (c) $A_T = 1000mm/s^2$ and $\delta = 2\mu m$

(d) $A_T = 2000mm/s^2$ and $\delta = 1\mu m$ (e) $A_T = 2000mm/s^2$ and $\delta = 1.5\mu m$ (f) $A_T = 2000mm/s^2$ and $\delta = 2\mu m$

Fig. 15. The machined vases

and cubic PH-splines, we can obtain the explicit formulas for the time-optimal solutions. Real industrial CNC machining are conducted show the feasibility of our algorithm.

In a similar way, we also give a velocity planning algorithm with the chord error and jerk bounds under a greedy rule. It is interesting to investigate whether the velocity curve thus obtained is time-optimal or not. In principle, the methods in Sections 2 and 3 can be combined to give a velocity planning method with confined jerk and acceleration, the detail of which will be investigated later.

Appendix. Proof of Theorem 2.3

In this appendix, we will show that the velocity curve given by Algorithm **VP_CETA** in Section 2.4. is the only solution to the optimization problem (7). The proof is divided

into two parts. We first show that the velocity curve computed by the algorithm is the curve defined in (12) and then prove that this velocity curve is the solution to problem (7). In order to prove this, we need the following result.

Theorem 6.1 *[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.

Lemma 6.2 *Let $v_1(u)$ and $v_2(u)$ be two velocity curves for the tool path $C(u)$ defined on $[u_1, u_2]$ and $a_{1T}(u), a_{2T}(u)$ their tangential accelerations respectively. If $v_1(u_1) \leq v_2(u_1)$ and $a_{1T}(u) \leq a_{2T}(u)$ 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 $v_1(u_1) = v_2(u_1)$, 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 satisfies the conditions of the lemma. Since $C(u)$ is differentiable to the order of three, $\sigma(u)$ and $a_{1T}(u)$ must be bounded in $[u_1, u_2]$. From (6), $q_1'(u) = 2\sigma(u)a_{2T}(u)$ and $q_2'(u) = 2\sigma(u)a_{2T}(u)$. Then, we have $q_1(u_1) = v_1(u_1)^2 = q_2(u_1) = v_2(u_1)^2$, $2\sigma(u)a_{2T}(u) \leq 2\sigma(u)a_{2T}(u)$ for $u \in [u_1, u_2]$, and $2\sigma(u)a_{1T}(u)$ satisfies the Lipschitz's condition. Using Theorem 6.1, 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 proving Theorem 2.3, 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 (5.2) and (6.2). In case (5.2), we have $v_{lim}^+(u_l) = v_s(u_l) < v_{lim}^-(u_l)$, which means that 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 from $(u_l, v_s(u_l))$, a segment of the A_T integration trajectory is below the CEVLC (See Fig. 2(a)). In case (6.2), there exists a parameter $u_n > u_l$ such that the tangential acceleration of the CEVLC must be strictly larger than A_T for $u \in (u_l, u_n)$. By Lemma 6.2, starting from point $(u_l, v_s(u_l))$, the A_T integration trajectory is below the CEVLC for $u \in (u_l, u_n)$ (See Fig. 2(b)). We thus prove the correctness of step 8. The correctness step 9 can be proved in a similar way.

The following two lemmas show that the intersection of an integration trajectory and a CEVLC segment or another integration trajectory behaves nicely.

Lemma 6.3 *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 can intersect 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 $i(u)$ is an A_T integration trajectory; otherwise, consider the $-u$ direction. Let us assume that u^* be the parameter for a possible intersection point.

Since the two key points are adjacent, there are three cases: (a): $a_{lim}(u) > A_T, u \in (u_l, u_n)$, (b): $a_{lim}(u) < -A_T, u \in (u_l, u_n)$, or (c): $-A_T < a_{lim}(u) < A_T, u \in (u_l, u_n)$. In case (a), since $i(u)$ is an A_T integration trajectory and $a_{lim}(u) > A_T$, by Lemma 6.2, we have $i(u) < v_{lim}(u)$ for $u \in (u_*, u_n]$. In the $-u$ direction, $i(u)$ is a $-A_T$ integration trajectory and $a_{lim}(u) < -A_T$. By Lemma 6.2, we have $i(u) > v_{lim}(u)$ for $u \in [u_l, u_*)$. That is, if they intersect then they only intersect once. Cases (b) and (c) can be proved similarly. ∎

Lemma 6.4 *In step 9 of the algorithm, the backward integration trajectory v_b intersects v_s only once.*

Proof. Two situations lead to step 9: step (5.3) and step (6.3). In case (5.3), the key point at u_n is discontinuous and $v^+_{lim}(u_n) \geq v_s(u_n) > v^-_{lim}(u_n)$. In the interval $[0, u_n]$ in the $-u$ direction, $v_b(u)$ is an A_T integration trajectory and $v_s(u)$ consists of $-A_T$ integration trajectories and feasible CEVLC segments. Also note that $v_s(0) = 0$. Then $v_b(u)$ and $v_s(u)$ must intersect in $[0, u_n]$. By Lemma 6.2, they can intersect only once. Case (6.3) 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.

We now prove Theorem 2.3 which is repeated below.

Theorem 6.5 *The velocity curve computed with Algorithm* **VP_CETA** *is the velocity curve defined in equation (12) and is the only solution to the optimization problem (7).*

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 $|a_T(u)| \leq A_T$. Also, if $|a_T(u)| \neq A_T$, the corresponding $v(u)$ must be a segment of feasible CEVLC. As a consequence, $v(u)$ satisfies conditions (8) and (9) and is bang-bang.

From the algorithm, it is clear that $v(u)$ consists of pieces of CEVLC and that of $v_P(u)$ for all key points P of the CEVLC including the start point $(0, 0)$ and the end point $(1, 0)$. To prove (12), 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 start and end 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_T and $|a_T(u)| \leq A_T$. Then by Lemma 6.2, $v_P(u) \geq v(u)$ 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_T, and hence $v_P(u) \geq v(u)$ for $u \in [0, u_0]$. As a consequence, $v_P(u)$ cannot be strictly smaller $v(u)$ at any u. We thus prove that $v(s)$ is the curve in (12).

We now prove that $v(u)$ 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. Assume the contrary, then there exists another velocity curve $v_*(u)$ satisfying the constraints (8) and (9), 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 A_T integration trajectory in the $+u$ direction, which is called an *increasing*

interval, or an $-A_T$ integration trajectory in the $+u$ direction, which is called a *decreasing interval*. Furthermore, if $[u_1, u_2]$ is an increasing interval, the start 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 $[u_0, u_*]$, since $v(u)$ is an A_T integration trajectory and the acceleration $a_*(u)$ of $v_*(u)$ satisfies $|a_*(u)| \leq A_T$, using Lemma 6.2, we have $v(u_*) \geq v_*(u_*)$, 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_T and the theorem can be proved similarly to the case of increasing intervals. ∎

Acknowledgement

We want to thank the anonymous referees for their valuable suggestions. We also thank Shenyang Institute of Computing Technology, CAS for assisting us to conduct the CNC machining experiments.

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] Seimens' CNC Controller Manual. Sinumerik 840d/810d/fm-nc programmming, advanced. 840d_pga, 5-146.

[20] S.H. Suh, S.K. Kang, D.H. Chung, I. Stroud. *Theory and Design of CNC Systems*, Springer, London, 2008.

[21] 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.

[22] 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.

[23] 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.

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

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

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

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

[28] L.X. Zhang, R.Y. Sun, X.S. Gao, H. Li. High speed interpolation for micro-line trajectory and adaptive real-time look-ahead in CNC machining. Accepted by *Scinece China, Series E*, 2011.

[29] 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. *MM Research Preprints*, 29, 71-91, 2010. Accepted by *Scinece China, Series F*, 2011.