Mathematics Mechanization Research Preprints
KLMM, Chinese Academy of Sciences
Vol. 30, 39–56, May 16, 2011

Efficient Algorithm for Feedrate Planning and Smoothing with Confined Chord Error and Acceleration for Each Axis[1)]

Ke Zhang, Chun-Ming Yuan, Xiao-Shan Gao
KLMM, Institute of Systems Science, Chinese Academy of Sciences

Abstract. In this paper, an algorithm is proposed to generate a near time-optimal velocity function with continuous accelerations for feedrate planning along a parametric tool path for 3-axis CNC machines with a feedrate bound, an acceleration bound for each axis, and a chord error bound. The algorithm first gives a discrete and computationally efficient algorithm to find a velocity function under the feedrate, acceleration, and chord error bounds and a proof is provided to show that this solution is globally time-optimal under the constraints. Then, a linear programming strategy is proposed to smooth the velocity curve, which incurs only at a small number of points where the acceleration of the feedrate function is not continuous. Combining the two steps, our algorithm gives a near time-optimal solution for the problem. Examples are presented to illustrate the feasibility of our algorithm.

Keywords. Optimal feedrate planning, chord error, confined acceleration, discrete velocity searching, linear programming, feedrate smoothing.

1. Introduction

The problem of time-optimal feedrate planning along parametric tool paths has received a significant amount of attention in both the robotics and CNC machining literature. In the feedrate planning, the acceleration on each axis of the machine must be constrained, because the torque (or force) capabilities of the axes' drives are limited. Therefore, the problem is how to identify the feedrate along a given tool path such that the machining time is minimal without exceeding the capabilities of the actuators.

Bobrow et al [2], Shiller and Lu [11] gave algorithms to determine the time-optimal motion for a robot manipulator along a specific path (at least a smooth curve) with acceleration bounds on x, y, z axes. Farouki and Timar [13, 14] planned the feedrate for CNC machining with acceleration bounds on x, y, z axes, and gave a piecewise-analytic expression of the optimal velocity planning function. They also proposed a method in [3] to generate smoothed feedrate functions, that incur only finite rates of change of motor torque and remain consistent with the axis acceleration bounds. Following the method in [13, 14], Zhang et al gave a simplified time-optimal velocity planning method for quadratic B-splines and realized real-time manufacturing on industrial CNC machines [18]. Yuan and Gao [16] provided a time-optimal feedrate planning method with tangential acceleration and chord error bounds.

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

Zhang et al tried to extend the above methods to the case of jerk bounds and gave a greedy feedrate planning algorithm [17].

All of the methods mentioned above use the velocity limit curve and its switching points on the $u-\dot{u}$ phase plane to obtain an optimal solution which is a continuous time-optimal velocity function along a specific path. However, these methods have two disadvantages. First, the computational expense of these algorithms is high, especially for tool pathes described by complex parametric functions. Computing the velocity limit curve and the switching points lead to the solving of high degree equations. Second, the constraints in this scheme are not easy to be generalized because the acceleration is directly integrated to find closed form solutions. To overcome these drawbacks, discrete models of the problem have been proposed and developed.

Luh et al [9] developed an algorithm to schedule the time intervals between each pair of adjacent knots such that the total traveling time is minimized subject to the physical constraints on joint velocities, accelerations, and jerks. Cubic spline functions in time t are used for constructing joint trajectories. The physical constraints can be expressed as inequality constraints of time intervals. Then it becomes a nonlinear programming problem with the sum of time intervals as the objective function.

Neuman and Tourassis [10] introduced an inherently discrete-time dynamic model for robotic manipulators. Tan and Potts [12] used the model [10] in minimum time trajectory planning. The joint torque constraints, joint jerk and joint velocity constraints are incorporated into the model. Their nonlinear optimization problem is partially linear which enables the iterative method of approximate programming to be used in solving the problem. It is verified numerically that the convergence of the iterative algorithm is quadratic so that the trajectory planner is computationally efficient.

Altintas and Erkorkmaz [6] presented a quintic spline trajectory generation algorithm that produces continuous position, velocity, and acceleration profiles with confined tangential acceleration and jerk. The reference trajectory generated with varying interpolation period is re-sampled at the servo loop closure period using fifth order polynomials.

Dong et al [4, 5] gave a discrete greedy algorithm for the velocity planning problem with velocity, acceleration, and jerk bounds on each axis based on a series of single variable optimization subproblems. The sub-optimization structure makes it easy to deal with any state-dependent constraints.

Lai et al [8] further proposed a method which can generate velocities with jerk limits as well as chord error, speed, and acceleration limits. The method uses a discrete model and satisfies all these constraints by backtracking at each step.

Most of these discrete algorithms adopted nonlinear programming to find the solutions, which is sensitive to initial values and cannot guarantee to be optimal. In this paper, we consider the problem of optimal feedrate planning along a smooth parametric tool path $\vec{r}(u)$ with chord error bound and acceleration limits for all the axes. First, we discretize the path curve and velocity to construct our discrete model. Then we give a velocity reachability function to determine if the velocities of two neighboring knots on the path are reachable. Based on the velocity reachability function, we propose a discrete velocity search algorithm.

The general idea of our algorithm is to search for the maximal velocity under the velocity limit curve in the forward direction, then search for the maximal velocity under the forward

velocity in the reverse direction. Here, we need only to compute the values of the velocity limit curve at the discrete points, which is computationally easy. A proof is followed to show that the solution of our algorithm is globally optimal under all the constraints.

The algorithm can be considered as a discrete version of the algorithm given in [2, 11, 13]. By using a discrete model, we can find optimal solutions for tool pathes described by any parametric functions efficiently. We show that the computational complexity of our algorithm is $O(NM)$ in terms of the number of floating point arithmetic operations, where N is the number of segments to discrete the tool path and M is the number of intervals to the feedrate.

After that, we adopt a linear programming strategy to adjust the velocities around each acceleration discontinuity point of the velocity locally to smooth the velocity curve and maintain a near time-optimal form of the feedrate.

Combining the two steps, our algorithm gives a near time-optimal velocity curve with continuous acceleration under the given constraints.

The rest of our paper will be organized as follows. Section 2 gives the mathematical description and theoretical analysis of our feedrate optimization problem. Section 3 gives the discrete velocity search algorithm. Section 4 proves the optimality of our algorithm. Section 5 gives the feedrate smoothing strategy. Section 6 gives two simulation examples. Section 7 concludes the paper.

2. Problem description and analysis

For brevity, we consider a plane parametric curve as the tool path, which has at least C^1 continuity:

$$\vec{r}(u) = (x(u), y(u)), 0 \le u \le 1.$$

The extension to spatial paths is straightforward. We denote the derivatives with respect to time t and the parameter u by dots and primes, respectively:

$$\dot{u} = du/dt, x' = dx/du.$$

In our feedrate planning, the acceleration bound on each axis, the chord error bound, and the tangential velocity bound of the machine will be considered.

Let $\sigma(u) = |\vec{r}'(u)| = \sqrt{x'(u)^2 + y'(u)^2}$. Then the tangential velocity along the tool path is $v(u) = \sigma(u)\dot{u}$. Firstly, we consider the bounds on the x and y acceleration components, which are A_x, A_y. Let $q(u) = v^2(u)$. Then the accelerations on the x and y axes are

$$\begin{cases} a_x = \ddot{x} = (x'\dfrac{v}{\sigma})'\dfrac{v}{\sigma} = \dfrac{1}{\sigma}(\dfrac{x'}{\sigma})'q + \dfrac{x'}{2\sigma^2}q' \\ a_y = \ddot{y} = (y'\dfrac{v}{\sigma})'\dfrac{v}{\sigma} = \dfrac{1}{\sigma}(\dfrac{y'}{\sigma})'q + \dfrac{y'}{2\sigma^2}q'. \end{cases} \tag{1}$$

At each $u \in [0, 1]$, the x and y acceleration constraints $-A_x \le a_x \le A_x, -A_y \le a_y \le A_y$ can be reduced to

$$\begin{cases} -A_x \le \dfrac{1}{\sigma}(\dfrac{x'}{\sigma})'q + \dfrac{x'}{2\sigma^2}q' \le A_x \\ -A_y \le \dfrac{1}{\sigma}(\dfrac{y'}{\sigma})'q + \dfrac{y'}{2\sigma^2}q' \le A_y, \end{cases} \tag{2}$$

which defines the interior of a parallelogram as the set of possible (q, q') values. Then the maximal value of q is identified by the right-most vertex of the parallelogram in the (q, q') phase plane [13].

Solving equations

$$\begin{cases} \frac{1}{\sigma}(\frac{x'}{\sigma})'q + \frac{x'}{2\sigma^2}q' = \alpha_x A_x \\ \frac{1}{\sigma}(\frac{y'}{\sigma})'q + \frac{y'}{2\sigma^2}q' = \alpha_y A_y, \end{cases} \tag{3}$$

where $\alpha_x = \pm 1, \alpha_y = \pm 1$, we have

$$q = \frac{\alpha_x A_x y' - \alpha_y A_y x'}{x''y' - y''x'}\sigma^2,$$

which are the four values of q in the four vertices of the parallelogram. The maximal value of q at each $u \in [0, 1]$ is then easy to obtain. Now we have the *x and y axes velocity limit curve* introduced in [2, 11, 13] which is determined by the acceleration constraints on the x and y axes:

$$V_{xy}(u) = \sigma\sqrt{\frac{A_x|y'| + A_y|x'|}{|x''y' - y''x'|}} \tag{4}$$

It is called velocity limit curve because the real velocity curve must be smaller that $V_{xy}(u)$ at each u.

Secondly, we consider the *chord error bound* ε. Let the curvature and radius of curvature of the tool path be

$$\kappa(u) = \frac{x'y'' - y'x''}{\sigma^3(u)}, \rho(u) = 1/|\kappa(u)|.$$

In general, the chord error bound is much less than the radius of curvature at each point on the tool path. By the chord error formula [16, 15], we have

$$q(u) = v(u)^2 \leq \frac{8\varepsilon\rho(u) - 4\varepsilon^2}{T^2} \approx \frac{8\varepsilon\rho(u)}{T^2} = \frac{8\varepsilon}{|\kappa(u)|T^2}$$

where T is the sampling period. Then the chord error constraint can be transformed to the centripetal acceleration constraint [16]. Let $A_N = 8\varepsilon/T^2$, we have

$$q(u) = v(u)^2 \leq \frac{8\varepsilon}{|\kappa(u)|T^2} \Longleftrightarrow |a_N(u)| = |\kappa(u)|v^2(u) \leq A_N.$$

Based on the above relation, the *chord error velocity limit curve* is

$$V_N(u) = \sqrt{\frac{A_N}{\kappa(u)}}. \tag{5}$$

Note that the real velocity curve must also be smaller than or equal to $V_N(u)$ for each u.

Together with the tangential velocity bound V_{max}, we obtain the *velocity limit curve* (abbr. VLC) with all the constraints:

$$V_{lim}(u) = \min\{V_{max}, V_{xy}(u), V_N(u)\}. \tag{6}$$

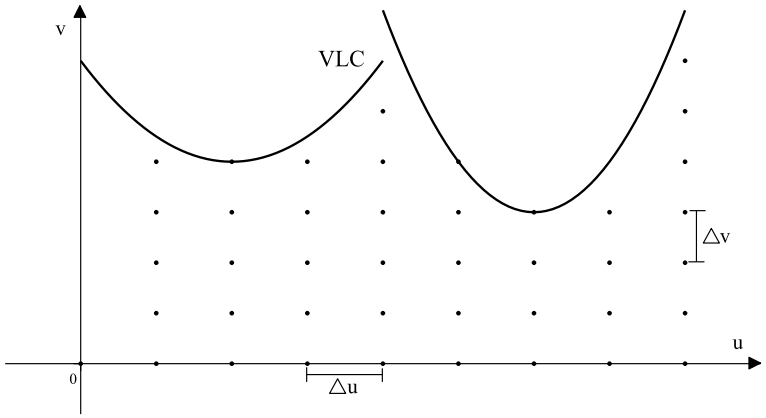


Fig. 1. Discretization of the parameter and the velocity

Note that a segment of $V_{lim}(u)$ could be used as the real velocity curve and such a segment is called *feasible*. In general, a feasible segment is either V_{max} or V_N, where the other two constraints are satisfied. A segment of V_{xy} is generally not feasible, since in such a case all the inequalities in (2) become equality which is not possible.

Then our feedrate optimization problem becomes to plan the velocity $v(u)$, such that the machining time is minimal:

$$\min t_f = \int_0^1 \frac{\sigma(u)}{v(u)} du \tag{7}$$

under the following constraints:

$$\begin{cases} |a_x(u)| \leq A_x \\ |a_y(u)| \leq A_y \\ 0 \leq v(u) \leq V_{lim}(u) \\ v(0) = v(1) = 0. \end{cases} \tag{8}$$

3. Discrete velocity search algorithm

We divide the tool path into N segments of parametric length Δu. The knots are

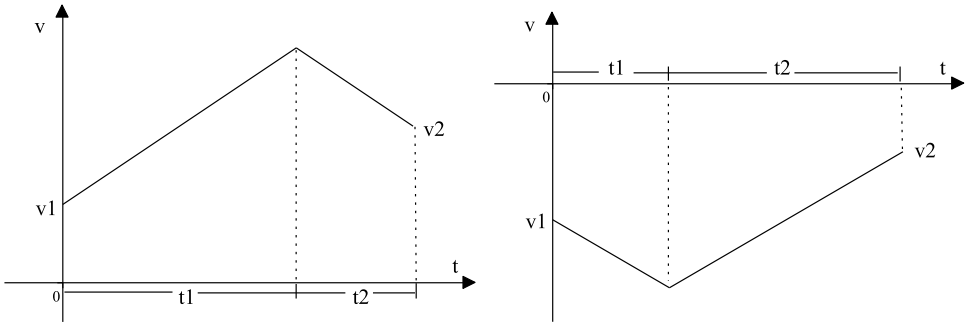
$$u_i = i\Delta u, i = 0, \ldots, N, \tag{9}$$

where $\Delta u = 1/N$. We choose an appropriate Δv to discretize the value of velocity at each u_i. Then the maximal feasible discrete velocity at u_i is $V_{lim}(u_i)$ which can be computed with (6), and the number of intervals for the feasible velocity at u_i is

$$N_v(i) = \lfloor \frac{V_{lim}(u_i)}{\Delta v} \rfloor. \tag{10}$$

Note that we need only to know the values $V_{lim}(u_i)$ of the VLC for the parameter value u_i. In the rest of the section, when we say VLC, we mean this discretized VLC. The velocity at u_i will be chosen in a series of discrete values (see Fig.1):

$$0, \Delta v, 2\Delta v, \ldots, N_v(i)\Delta v.$$



(a) $v_1 \geq 0, v_2 \geq 0$ (b) $v_1 \leq 0, v_2 \leq 0$

Fig. 2. Two cases of acceleration profile between two knots

3.1. Velocity reachability

Before presenting the complete algorithm, we need to give a subfunction $\mathrm{VR}(\mathrm{v_s}, \mathrm{u_s}, \mathrm{v_e}, \mathrm{u_e})$ which decides if two neighboring knots u_s, u_e with their velocities v_s, v_e are reachable. That is, whether it is possible to move the tool from point $(x(u_s), y(u_s))$ with initial velocity v_s to point $(x(u_e), y(u_e))$ with final velocity v_e under the constraints (8).

Firstly, we consider the reachability on each axis. Suppose that the velocity components (with signs) on an axis (x or y) of two neighboring knots u_s, u_e are v_1, v_2, respectively, the projection distance (with sign) on this axis between the two knots is Δs, and the acceleration bound of this axis is A. We define a function

$$\mathrm{vr}(v_1, \Delta s, v_2, A) = \begin{cases} 1 & \text{if the two knots with their velocities on this axis are reachable;} \\ 0 & \text{otherwise.} \end{cases}$$

Because the distance between the two neighboring knots is quite small, we simplify the problem to be only with the acceleration bound of this axis here, which means that $V_{lim}(u)$ is not achievable. Minimum-time path traversals generally involve a "Bang-Bang control" strategy. Then, it is easy to show that there has only two cases of acceleration profile when v_1, v_2 have the same signs (see Fig.2).

If v_1 and v_2 are both nonnegative, the case is shown in Fig.2(a): use A to accelerate and then $-A$ to decelerate. If v_1 and v_2 are both less than or equal to 0, the case is shown in Fig.2(b): use $-A$ first and then A. The distance $|\Delta s|$ has a lower bound when v_1 and v_2 have the same signs. Setting t_1 or t_2 to be 0, the area of the trapezoid on the (t, v) phase plane in Fig.2 is just the lower bound of $|\Delta s|$. The area is easy to compute as $\frac{|v_2^2 - v_1^2|}{2A}$. So, when $v_1 v_2 \geq 0$, the two knots are reachable on this axis if and only if $|\Delta s| \geq \frac{|v_2^2 - v_1^2|}{2A}$.

If v_1 and v_2 have the opposite signs, it means that there exists a point whose velocity on this axis is 0 between the two knots. We can obtain the parameter u_m of this point by solving $x'(u) = 0$ or $y'(u) = 0$ (we use the bisection method between u_s and u_e). Then the projection distance Δs is divided into Δs_1 which is the projection distance from u_s to u_m, and Δs_2 which is the projection distance from u_m to u_e. Then, when $v_1 v_2 < 0$, the two knots are reachable on this axis if and only if $|\Delta s_1| \geq \frac{v_1^2}{2A}$ and $|\Delta s_2| \geq \frac{v_2^2}{2A}$.

Then we have

$$\mathrm{vr}(v_1, \Delta s, v_2, A) = \begin{cases} 1 & v_1 v_2 \geq 0 \text{ and } |\Delta s| \geq \frac{|v_2^2 - v_1^2|}{2A}, \\ & \text{or } v_1 v_2 < 0 \text{ and } |\Delta s_1| \geq \frac{v_1^2}{2A}, |\Delta s_2| \geq \frac{v_2^2}{2A}; \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that two knots with their velocities are reachable if and only if both their projections on x and y axis are reachable. We have

$$\begin{aligned} \mathrm{VR}(v_s, u_s, v_e, u_e) =& \mathrm{vr}(\frac{x'(u_s)}{\sigma(u_s)} v_s, x(u_e) - x(u_s), \frac{x'(u_e)}{\sigma(u_e)} v_e, A_x) \cdot \\ & \mathrm{vr}(\frac{y'(u_s)}{\sigma(u_s)} v_s, y(u_e) - y(u_s), \frac{y'(u_e)}{\sigma(u_e)} v_e, A_y). \end{aligned} \tag{11}$$

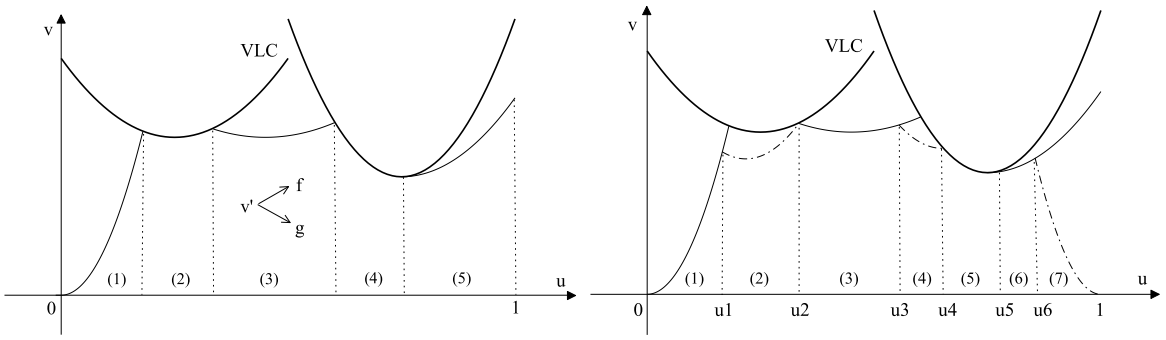
3.2. Feedrate planning algorithm

Our algorithm consists of a forward direction pass and a reverse direction pass. During the forward pass, a trajectory is generated to search for the next maximal feasible velocity under the VLC at each step; otherwise it will maintain on the VLC. The current maximal feasible velocity is obtained by computing the velocity reachability function. During the reverse pass, the maximal feasible preceding velocity under the forward velocity is obtained at each step; otherwise it will be equal to the forward velocity. Now we state the complete algorithm below:

Algorithm 1. *Discrete velocity search algorithm*
Input: $\vec{r}(u), V_{max}, A_x, A_y, \varepsilon, N, \Delta v$.
Output: the velocity v_i^* at each u_i, $i = 0, \ldots, N$.

1. Compute the number of velocity intervals $N_v(i)$ at each u_i, $i = 0, \ldots, N$ with (10).
2. $v_0 = 0$, $i = 0$.
3. Traverse v_{i+1} from $N_v(i+1)\Delta v, (N_v(i+1)-1)\Delta v, \ldots,$ to 0, until $\mathrm{VR}(v_i, u_i, v_{i+1}, u_{i+1}) = 1$. If $v_{i+1} = 0$, set $v_{i+1} = N_v(i+1)\Delta v$.
4. $i = i + 1$. If $i < N$ go to step 3, otherwise continue.
5. $v_N^* = 0$,$i = N$.
6. Traverse v_{i-1}^* from $v_{i-1}, v_{i-1} - \Delta v, \ldots,$ to 0, until $\mathrm{VR}(v_{i-1}^*, u_{i-1}, v_i^*, u_i) = 1$. If $v_{i-1}^* = 0$, set $v_{i-1}^* = v_{i-1}$.
7. $i = i - 1$. If $i > 0$ go to step 6, otherwise continue.
8. Output v_i^*, $i = 0, \ldots, N$.

We use Fig.3 to illustrate Algorithm 1, where Fig.3(a) shows the forward pass (steps 2-4) and Fig.3(b) the reverse pass (steps 5-7). In Fig.3(a), the forward velocity curve starts from $(0, 0)$ and searches for the current maximal feasible velocity under the VLC at each step in interval (1) in Fig.3. Then it maintains on the VLC in interval (2) because the velocity



(a) The forward direction pass (b) The reverse direction pass

Fig. 3. An illustration of Algorithm 1. The horizontal axis is the parameter u of the tool path. The vertical axis is the tangential velocity.

reachability function has no solution or the velocity is feasible on the VLC. Then the velocity curve departs from the VLC and searches for the maximal feasible velocity again in interval (3). Intervals (4) and (5) are similar to intervals (2) and (3) respectively. In Fig.3(b), the reverse velocity curve starts from point $(1, 0)$ and searches for the maximal feasible preceding velocity under the forward velocity curve. It updates the forward velocity curve in intervals (2)(4)(7). Then we obtain the velocity values at each u_i.

Now, we discuss how to chose the numbers N and Δv properly so that the algorithm will give a valid solution. The number of segments N can be chosen more or less as the number of interpolation steps along the tool path. It can be estimated as $S/(V_{ave}T)$, where S is an estimation of the length of the tool path, V_{ave} is an estimation of the average velocity.

The value Δv can not be too large, otherwise Algorithm 1 may not obtain a solution. Here we give a limit to Δv as below. In the forward pass, $v_i = 0$ leads to the bounds of v_{i+1} from the velocity reachability function:

$$\begin{cases} -2A_x|x(u_{i+1}) - x(u_i)| \le (\frac{x'(u_{i+1})}{\sigma(u_{i+1})}v_{i+1})^2 \le 2A_x|x(u_{i+1}) - x(u_i)| \\ -2A_y|y(u_{i+1}) - y(u_i)| \le (\frac{y'(u_{i+1})}{\sigma(u_{i+1})}v_{i+1})^2 \le 2A_y|y(u_{i+1}) - y(u_i)| \end{cases}$$

Then we have

$$v_{i+1} \le \min\{\frac{\sigma(u_{i+1})}{|x'(u_{i+1})|}\sqrt{2A_x|x(u_{i+1}) - x(u_i)|}, \frac{\sigma(u_{i+1})}{|y'(u_{i+1})|}\sqrt{2A_y|y(u_{i+1}) - y(u_i)|}\} \triangleq V_a(i+1)$$

To make sure that there is a feasible discrete velocity value at each u_{i+1}, Δv can not be greater than $V_a(i+1)$ at each u_{i+1}, $i = 0, \ldots, N-1$. Then we have

$$\Delta v \le \min_{1 \le i \le N} V_a(i) \triangleq V_a$$

Together with the similar upper bound V_b in the reverse pass, the limit to Δv is $\min\{V_a, V_b\}$. We can choose the value of Δv to be $\min\{V_a, V_b\}/100 \sim \min\{V_a, V_b\}/10$ in Algorithm 1.

Let $M = \max_{0 \le i \le N}\{N_v(i)\}$. The computational complexity of Algorithm 1 is $O(NM)$ of arithmetic operations of floating-point calculation since there are twice loop operations in

the algorithm, and in each step inside the loop we need only to perform a fixed number of floating point arithmetic operations. A more detailed analysis of Algorithm 1 will be given in the next section.

4. Time-optimality of the algorithm

In this section, we will prove that the algorithm does obtain a globally time-optimal solution for problem (7) under constraints (8) as Δu and Δv approach to 0.

The method that we use to prove optimality is similar to that used in [2, 11, 13, 16]. The phase plane (u, v) and the VLC are used to construct a solution for the minimum time control problem. We analyze our algorithm in the same phase plane and then show it is time-optimal.

Using (2), we can rewrite the x and y acceleration constraints to be the constraints of v':

$$\begin{cases} f_x(u,v) \le v' \le g_x(u,v) \\ f_y(u,v) \le v' \le g_y(u,v) \end{cases} \tag{12}$$

Let $f = \max\{f_x, f_y\}, g = \min\{g_x, g_y\}$. Then the constraints (12) lead to

$$f(u,v) \le v' \le g(u,v), \tag{13}$$

where $f(u,v), g(u,v)$ are piecewise continuously differentiable functions of u, v. The inequality (13) shows that v' has an upper and lower bound at any point on the phase plane (u, v) under the VLC.

When Δu and Δv approach to 0, we can see that step 3 in Algorithm 1 searches for the current maximal v' under all the constraints. Although only a finite number of values for the velocity is found, we could consider a *velocity curve* $v(u)$ is computed in our proof when Δu and Δv approach to 0.

In steps 2-4, the segments of the velocity curve are either under the VLC or equal to $V_{lim}(u)$. If the velocity curve segment is under the VLC, it satisfies $v' = g(u,v)$, which means that the maximal acceleration A_x or A_y is adopted. If the segment is equal to $V_{lim}(u)$, there are two cases: $V_{lim}(u)$ is a feasible solution in step 3, which means $f(u,v) \le V'_{lim}(u) \le g(u,v)$; or there has no solution in step 3, which means $V'_{lim}(u) \le f(u,v) \le g(u,v)$. Together with the two cases, we have $V'_{lim}(u) \le g(u,v)$ when the velocity curve segment is equal to $V_{lim}(u)$.

The analysis of steps 5-7 is similar. Step 6 searches for the current minimal v' with all the constraints. If the velocity curve segment is under the VLC, it satisfies $v' = f(u,v)$, which means that the minimal acceleration $-A_x$ or $-A_y$ is adopted. If the segment is equal to $V_{lim}(u)$, it satisfies $f(u,v) \le V'_{lim}(u)$.

Now it is clear that the complete velocity curve consists of three types of segments (we use Fig.3(b) to illustrate): the segments which satisfy $v' = g(u,v)$ (see Fig.3(b) segments (1)(3)(6)), the segments which satisfy $v' = f(u,v)$ (see Fig.3(b) segments (2)(4)(7)), and the feasible segments which are parts of the VLC curve $V_{lim}(u)$ (see Fig.3(b) segment (5)). If the velocity curve segment is equal to $V_{lim}(u)$, it satisfies the x and y acceleration constraints since it satisfies $V'_{lim}(u) \le g(u,v)$ and $f(u,v) \le V'_{lim}(u)$. Then all the segments of the velocity curve satisfy the x and y acceleration constraints and are under the VLC.

Hence, to prove the optimality, we need only to show that the velocity curve given by Algorithm 1 is higher than any other velocity curve on the phase plane (u, v), which satisfies constraints (8). In order to prove this, we need the following result.

Comparison Theorem (p.25, [1]): 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 $x \in [a, b]$.

Using the comparison theorem, we can prove that the algorithm obtains a globally optimal solution. Let $\hat{v}$ be any feasible solution for (8) and v be the solution given by Algorithm 1.

We use Fig.3(b) to illustrate the proof. In segment (1) in Fig.3(b), we have $\hat{v}(0) = v(0)$, and

$$v' = g(u, v), \hat{v}' \leq g(u, \hat{v}), u \in [0, u_1].$$

$g(u, v)$ satisfies Lipschitz's condition since it is piecewise continuously differentiable. From the comparison theorem, we have $\hat{v} \leq v, u \in [0, u_1]$.

In segments (3) and (6) in Fig.3(b), we have $\hat{v}(u_2) \leq v(u_2), \hat{v}(u_5) \leq v(u_5)$ since $v(u_2), v(u_5)$ are on the VLC. We can change the condition $y(a) = z(a)$ to $y(a) \leq z(a)$ in the comparison theorem. The result will not change according to the proof of the comparison theorem. Then we have

$$\hat{v} \leq v, u \in [0, u_1] \cup [u_2, u_3] \cup [u_5, u_6].$$

In segment (7) in Fig.3(b), we have $\hat{v}|_{\tilde{u}=0} = v|_{\tilde{u}=0}$, and

$$\frac{d}{d\tilde{u}} v(1 - \tilde{u}) = -f(1 - \tilde{u}, v(1 - \tilde{u})), \frac{d}{d\tilde{u}} \hat{v}(1 - \tilde{u}) \leq -f(1 - \tilde{u}, \hat{v}(1 - \tilde{u})), \tilde{u} \in [0, 1 - u_6]$$

by making the change of variable $\tilde{u} = 1 - u$. $-f(1 - \tilde{u}, v(1 - \tilde{u}))$ satisfies Lipschitz's condition since $f(u, v)$ is piecewise continuously differentiable. From the comparison theorem, we have

$$\hat{v}(1 - \tilde{u}) \leq v(1 - \tilde{u}), \tilde{u} \in [0, 1 - u_6],$$

which is equivalent to $\hat{v} \leq v, u \in [u_6, 1]$.

In segments (2) and (4) in Fig.3(b), the proof is similar to what we have done for segments (3) and (6). Then we have

$$\hat{v} \leq v, u \in [u_1, u_2] \cup [u_3, u_4] \cup [u_6, 1].$$

Together with $\hat{v} \leq v, u \in [u_4, u_5]$, we have $\hat{v} \leq v, u \in [0, 1]$. Then from

$$\frac{\sigma(u)}{v(u)} \leq \frac{\sigma(u)}{\hat{v}(u)},$$

we have

$$\int_0^1 \frac{\sigma(u)}{v(u)} du \leq \int_0^1 \frac{\sigma(u)}{\hat{v}(u)} du,$$

which proves the optimality of our algorithm.

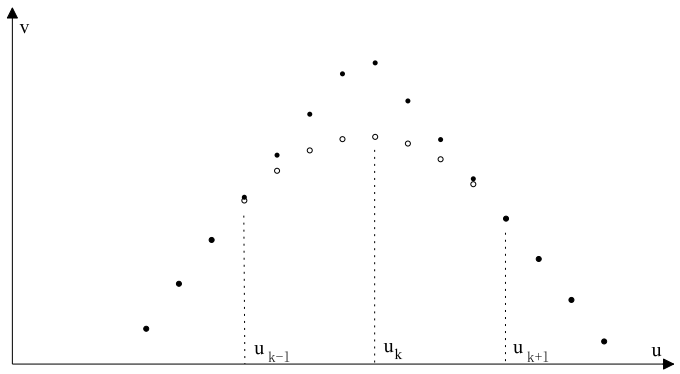


Fig. 4. Feedrate smoothing around the velocity derivative discontinuity

5. Feedrate smoothing

It is known that the velocity found with Algorithm 1 is a discretization of the velocity curve found with the methods in [2, 11, 13, 16], which is piecewise differentiable. So its acceleration profile has discontinuities at a small number of parametric values, which will cause vibrations and then large contouring errors. From (1), when the parametric tool path has at least C^2 continuity, the acceleration discontinuity points are just the discontinuity points of v' (or q'). In this case, one method to reduce vibrations is smoothing the velocity curve.

In the proof of optimality of Algorithm 1 in section 4, we show that the velocity curve consists of three parts: the segments which satisfy $v' = g(u, v)$, the segments which satisfy $v' = f(u, v)$, and the segments which are parts of $V_{lim}(u)$. Then, every discontinuity point of v' is either the intersection of any two parts of the above three, or the derivative discontinuity of the VLC. The first kind of discontinuities is easy to find throughout the process of Algorithm 1. The location of the second kind of discontinuities is not direct. We will deal with this case later.

First, we consider the case that v' suddenly decreases at a point. To find these points, we just need to add steps 1', 3' and 6' after steps 1, 3 and 6 respectively in Algorithm 1:

1'. Initialization: $\text{flag}(i) = 0$, $i = 0, \ldots, N$

3'. If $v_i < N_v(i)\Delta v$ and $v_{i+1} = N_v(i+1)\Delta v$, set $\text{flag}(i+1) = 1$.

6'. If $v_i^* < v_i$ and $v_{i-1}^* = v_{i-1}$, set $\text{flag}(i-1) = 1$; if $v_{i-1}^* < v_{i-1}$ and $\text{flag}(i-1) = 1$, set $\text{flag}(i-1) = 0$.

In the forward pass of Algorithm 1, step 3' marks the points where the velocity segment satisfying $v' = g(u, v)$ intersects the VLC with $\text{flag}(i+1) = 1$ at u_{i+1}. In the reverse pass, step 6' marks the points where the velocity segment satisfying $v' = f(u, v)$ intersects the forward velocity curve and removes the marks where the forward velocity curve is updated by the reverse velocity curve.

Suppose $\text{flag}(k) = 1$, that is, v' suddenly decreases at parametric u_k. Choosing an appropriate small positive integer l, we decrease the velocities computed by Algorithm 1 at

parametric $u_{k-l}, \ldots, u_{k+l}$, to obtain a finite rate of change of v' (or q') (see Fig.4). With the approximation of q' and q'' by

$$q'_i = \frac{q_i - q_{i-1}}{\Delta u}, q''_i = \frac{q_{i+1} - 2q_i + q_{i-1}}{\Delta u^2},$$

we solve the following linear programming problem to obtain the new q_i:

$$\min \sum_{i=k-l}^{k+l} ({v_i^*}^2 - q_i) \tag{14}$$

subject to

$$\begin{cases} 0 \le q_i \le {v_i^*}^2, \ i = k-l, \ldots, k+l \\ -A_x \le \dfrac{1}{\sigma}(\dfrac{x'}{\sigma})'(u_i)q_i + \dfrac{x'}{2\sigma^2\Delta u}(q_i - q_{i-1}) \le A_x, \ i = k-l, \ldots, k+l+1 \\ -A_y \le \dfrac{1}{\sigma}(\dfrac{y'}{\sigma})'(u_i)q_i + \dfrac{y'}{2\sigma^2\Delta u}(q_i - q_{i-1}) \le A_y, \ i = k-l, \ldots, k+l+1 \\ -J \le \dfrac{1}{\Delta u^2}(q_{i+1} - 2q_i + q_{i-1}) \le J, \ i = k-l-1, \ldots, k+l+1 \end{cases} \tag{15}$$

where v_i^* are the velocities computed by Algorithm 1, J is an appropriate limit of q''.

Note that the second and third constraints in (15) are approximate version of constraints (2), which guarantee that the acceleration on each axis is confined. The fourth constraint in (15) can be considered to have a confined jerk approximately, which will lead to continuous accelerations.

The new velocities at parametric values $u_{k-l}, \ldots, u_{k+l}$ satisfy all the original constraints and have eliminated the discontinuity of q' at u_k in the sense of discrete model.

Usually, the small positive integer l can be predetermined (for example, $l = N/100$). It is clear that the above linear programming always has a solution when the q'' (approximate jerk) bound J is large enough. Since

$$|q_{i+1} - 2q_i + q_{i-1}| \le |q_{i+1} - q_i| + |q_i - q_{i-1}|, \ i = k-l-1, \ldots, k+l+1,$$

we have

$$|q_{i+1} - 2q_i + q_{i-1}| \le 2 \max_{i=k-l-2,\ldots,k+l+1} |q_{i+1} - q_i|, \ i = k-l-1, \ldots, k+l+1.$$

So we can choose the initial value of J to be

$$J = \frac{2}{\Delta u^2} \max_{i=k-l-2,\ldots,k+l+1} |{v_{i+1}^*}^2 - {v_i^*}^2|.$$

For this J, the linear programming problem (14) always has solutions. Then we decrease J step by step and solve the above linear programming problem every time until an acceptable solution is obtained.

In the case that v' suddenly increases at u_k, which is also easy to find throughout the process of Algorithm 1 as above, we choose a velocity adjusting interval $[u_{k-l_1}, u_{k+l_2}]$ to

cover this point, the preceding and the next v' discontinuity point. Then we solve the linear programming problem above together for the three discontinuities. If the discontinuity is on the VLC and we do not know where it is, the velocity adjusting interval can be chosen to cover this VLC segment as well. However, this two cases rarely occur.

The feedrate smoothing with linear programming problem will not increase the complexity of our algorithm since the linear programming problem has efficient polynomial-time algorithms [7] with complexity $O(n^{3.5})$, and the dimension of our linear programming problem $n = 2l + 1 \approx N/50$ is quite small compared with N.

6. Simulation results

In this section, we use two examples to illustrate the feasibility of our algorithm. The first example is used to show that Algorithm 1 can be used to obtain the time-optimal solution and the complexity of the algorithm is $O(NM)$. The second example, which is often used as a manufacturing model in the literature, is used to illustrate that our method can obtain a near time-optimal velocity function with continuous acceleration efficiently.

First, we use an example to illustrate Algorithm 1. The curve in Fig.5 is a quadratic B-spline consisting of 14 pieces of quadratic curve segments, which has C^1 continuity. We set the tangential velocity and x, y acceleration bounds to be $V_{max} = 50mm/s, A_x = 2000mm/s^2, A_y = 1000mm/s^2$, respectively. The chord error bound is $\varepsilon = 1\mu m$ and the sampling period is $T = 2ms$, which means that the centripetal acceleration bound is $A_N = 2000mm/s^2$.

Table 1. Executing time in seconds of Algorithm 1 with different N and Δv.

N	100	100	200	200	300	300
Δv (mm/s)	0.02	0.01	0.02	0.01	0.02	0.01
Duration (s)	0.011	0.022	0.021	0.042	0.033	0.064

For discretization, we choose $N = 100, 200, 300$ and $\Delta v = 0.01, 0.02mm/s$ respectively in this example. The executing time of Algorithm 1 with different N and Δv are listed in Table 1. (CPU: Intel Core2 Duo, 2.93GHz; programming software: Microsoft Visual C++ 6.0). From the table, we can see that the executing time is approximately proportional to NM or $\frac{N}{\Delta V}$, which validates the $O(MN)$ computational complexity of the algorithm.

In Fig.6(a)(b)(c), the dotted curve is the VLC computed by (6), and the solid ones are the velocity curves obtained with Algorithm 1 when $\Delta v = 0.01mm/s$ and $N = 100, 200, 300$, respectively. We can see that the velocity curve obtained with $N = 100$ is already very close to the time-optimal solution. On the other hand, the velocity curve is improved when N becomes larger. For instance, the velocity in Fig.6(b) is larger than that in Fig.6(a) at $u = 0.08$ and the velocity in Fig.6(c) is larger than that in Fig.6(b) at $u = 0.23$.

Fig.6(d) shows the chord error at each parametric u when $\Delta v = 0.01mm/s, N = 300$. Fig.6(e) and (f) shows the x and y accelerations when $\Delta v = 0.01mm/s, N = 300$, respectively. We can see that it satisfies "Bang-Bang" control, that is, at least one of the constraints reaches its limit at any time, which also validates that the velocity curve is time-optimal.

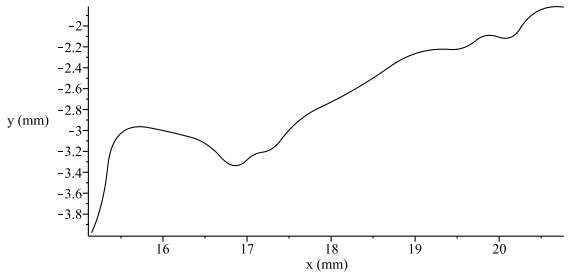
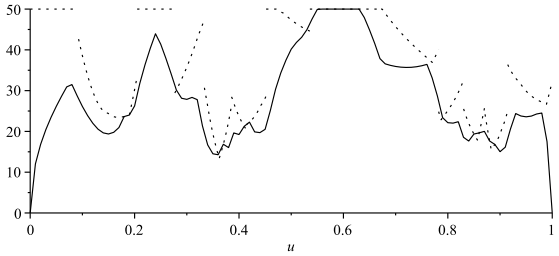
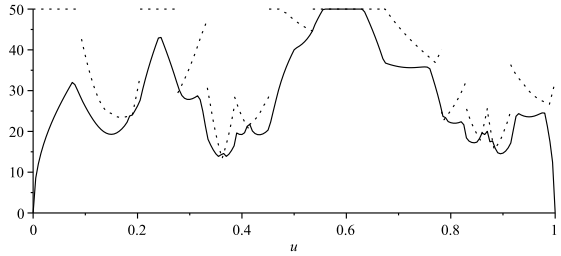


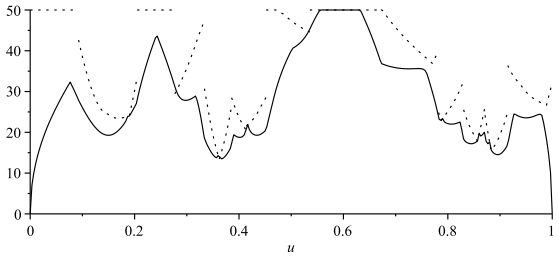
Fig. 5. A planar quadratic B-spline



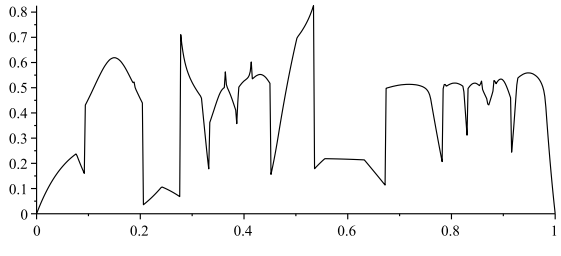
(a) Vel. for $N = 100, \Delta v = 0.01mm/s$



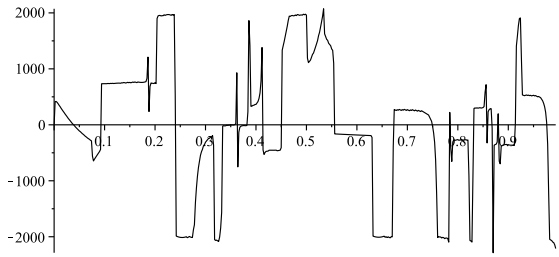
(b) Vel. for $N = 200, \Delta v = 0.01mm/s$



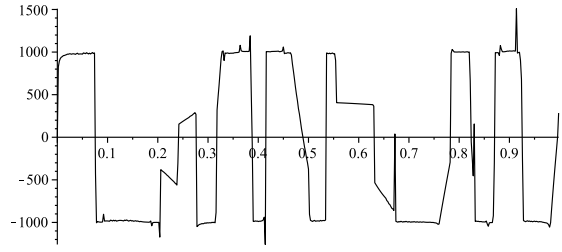
(c) Vel. for $N = 300, \Delta v = 0.01mm/s$



(d) The chord error (μm)



(e) Acceleration on the x-axis (mm/s^2)



(f) Acceleration on the y-axis (mm/s^2)

Fig. 6. The velocity, chord error, and acceleration on each axis. The horizontal axis is the parameter u of the tool path.

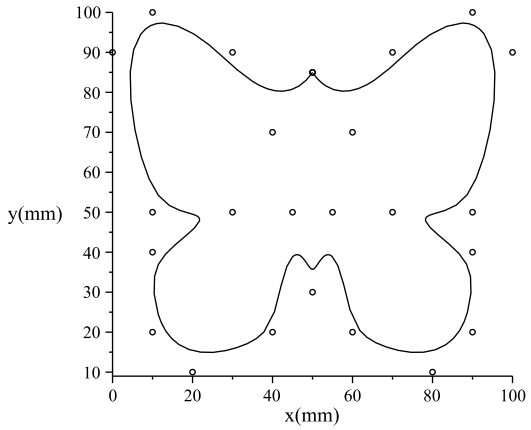


Fig. 7. A degree 3 NURBS curve and its control points

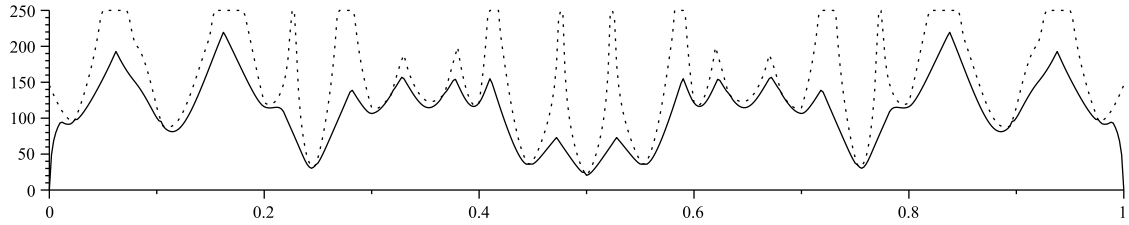
Now we use another example to demonstrate the feedrate smoothing given in section 5. The curve in Fig.7 is a degree three NURBS curve with 25 control points. The parameters of the curve are listed in Table 2. It is a C^2 curve since each knot has multiplicity 1.

Table 2. Parameters of a butterfly curve.

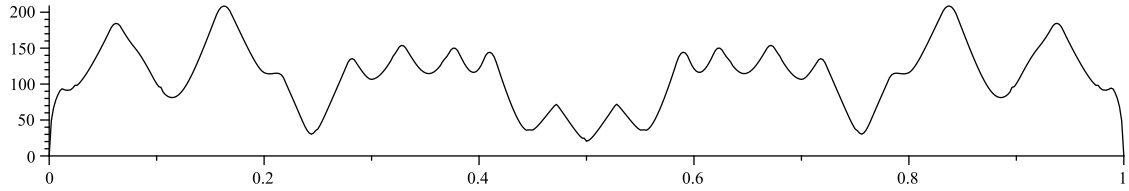
Parameters	Items
Control points:	(50, 85), (40, 70), (30, 90), (10, 100), (0, 90), (10, 50), (30, 50), (10, 40), (10, 20), (20, 10), (40, 20), (45, 50), (50, 30), (55, 50), (60, 20), (80, 10), (90, 20), (90, 40), (70, 50), (90, 50), (100, 90), (90, 100), (70, 90), (60, 70), (50, 85)
Knot vector:	0, 0, 0, 0, 0.08, 0.1, 0.15, 0.2, 0.24, 0.3, 0.33, 0.38, 0.4, 0.42, 0.5, 0.58, 0.6, 0.62, 0.67, 0.7, 0.76, 0.8, 0.85, 0.9, 0.92, 1, 1, 1, 1
Weights:	1, 0.7, 1, 1, 1, 1.5, 1, 1, 0.5, 0.5, 1, 1, 2, 1, 1, 0.5, 0.5, 1, 1, 1.5, 1, 1, 1, 0.7, 1
Degree:	3

We set $V_{max} = 250mm/s, A_x = 1000mm/s^2, A_y = 1000mm/s^2, \varepsilon = 1\mu m, T = 2ms, N = 500, \Delta v = 0.05mm/s$, $l = 3$, and $J = (\frac{28}{\Delta u}mm/s)^2$ in this example.

Fig.8(a) shows the VLC (dotted) and original velocity curve (solid) given by Algorithm 1. There are 16 v' discontinuity points. Figs.8(b) shows the smoothed velocity curve. Fig.9(a)(b)(c) shows the chord error, the x and y accelerations of the smoothed velocity curve, respectively. Comparing Fig.8(a) and Fig.8(b), we can see that the velocity only changes around the 16 v' discontinuity points and the amount of changes is not much. Also, from Fig.9(b) and Fig.9(c), the velocity is approximately "Bang-Bang" control. As a consequence, the final velocity curve is near time-optimal.

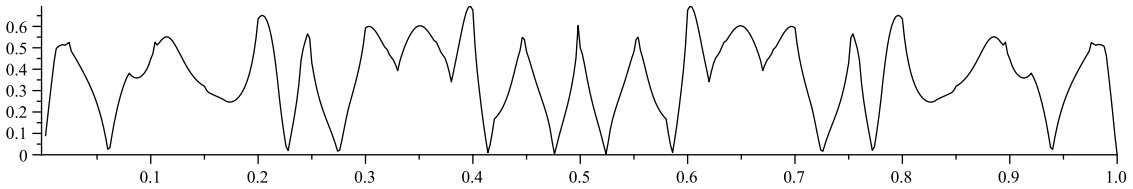


(a) The VLC and velocity curve (mm/s)

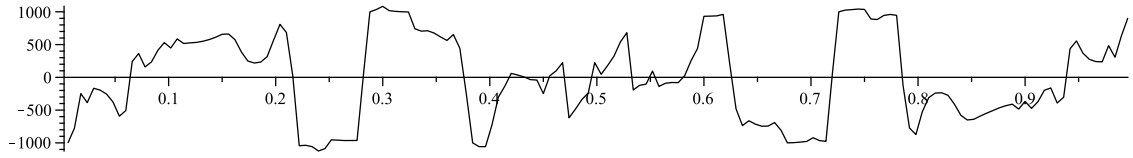


(b) The smoothed velocity curve (mm/s)

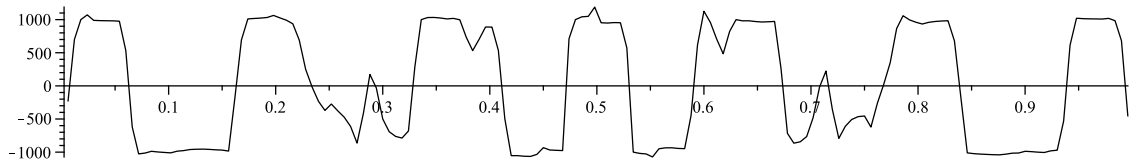
Fig. 8. The velocity curve and its smoothing



(a) The chord error (μm)



(b) The acceleration on the x-axis (mm/s^2)



(c) The acceleration on the y-axis (mm/s^2)

Fig. 9. The chord error and acceleration on each axis

7. Conclusion

The algorithm proposed in this paper is computationally more efficient than the analytical solving techniques when the parametric functions for tool path are complex. We avoid computing the expression of the VLC and switching points. We also need not to integrate the accelerations. Our discrete velocity search algorithm just need to compute a velocity reachability function at each step, and the complexity of the algorithm is $O(NM)$ number of floating-point arithmetic operations.

Comparing with the existing discrete algorithms using nonlinear programming techniques, our algorithm can give time-optimal solution with confined acceleration and chord error. Using a linear programming scheme to remove the sudden jumps occurring in the acceleration of the time-optimal feedrate solution, we obtain a near time-optimal solution with continuous accelerations. Most of the other discrete algorithms which can obtain an optimal solution adopted nonlinear programming. It is not easy to control since the nonlinear programming relies on a feasible initial solution.

References

[1] G. Birkhoff, G.C. Rota. *Ordinary differential equations*. John Wiley, New York, 1969.

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

[3] C.L. Boyadjieff, R.T. Farouki, S.D. Timar. Smoothing of time-optimal feedrates for cartesian CNC machines. *Mathematics of Surfaces XI*, LNCS 3604, 84C101, Springer Berlin, 2005.

[4] J. Dong, J.A. Stori. A generalized time-optimal bi-directional scan algorithm for constrained feedrate optimization. *ASME Journal of Dynamic Systems, Measurement, and Control*, 128, 379-390, 2006.

[5] J. Dong, P.M. Ferreiraa, J.A. Stori. Feedrate optimization with jerk constraints for generating minimum-time trajectories. *International Journal of Machine Tools and Manufacture*, 47, 1941-1955, 2007.

[6] K. Erkorkmaz, Y. Altintas. High speed CNC system design Part I: jerk limited trajectory generation and quintic spline interpolation. *International Journal of Machine Tools and Manufacture*, 41, 1323-1345, 2001.

[7] N. Karmarkar. A new polynomial-time algorithm for linear programming. *ACM Symposium on Theory of Computing*, New York, 302-311, 1984

[8] 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(1-2), 104-121,2008.

[9] C.S. Lin, P.R. Chang, J.Y.S. Luh. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Trans Automat Contr*, AC-28, 1066-1074, 1983.

[10] C.P. Neuman, V.D. Tourassis. Discrete dynamic robot models. *IEEE Trans Syst Man Cybern*, SMC-15(2), 193-204, 1985.

[11] Z. Shiller, H.H. Lu. Robust computation of path constrained time optimal motions. *IEEE International Conference on Robotics and Automation*, Cincinnati, OH, 144-149, 1990.

[12] H.H. Tan, R.B. Potts. Minimum time trajectory planner for the discrete dynamic robot model with dynamic constraints. *IEEE J Robotics Autom*, 4, 174-185, 1988.

[13] 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, 37-53, 2005.

[14] 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(2), 563-579, 2007.

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

[16] 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.

[17] 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.

[18] M. Zhang, W. Yan, C.M. Yuan, D. Wang, X.S. Gao. Curve fitting and time-optimal interpolation on CNC machines. Accepted by Science China, Series E, 2011.