Efficient Algorithm for Time-optimal Feedrate Planning and Smoothing with Confined Chord Error and Acceleration [*]

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

Abstract

In this paper, an efficient algorithm is proposed to generate a smooth near-time-optimal feedrate function along a parametric tool path for 3-axis CNC machining under 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 sequence of globally optimal velocity points under the feedrate, acceleration, and chord error bounds. A linear programming strategy is proposed to smooth the velocity points sequence. Finally, the velocity points sequence is fitted into a cubic spline to obtain a near-time-optimal and smooth feedrate function. Simulation and experiment results for NURBS curves are presented to illustrate the feasibility of the algorithm.

Keywords. Time-optimal feedrate planning, chord error, confined acceleration, discrete velocity searching, 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 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. Furthermore, smoothed feedrate functions are desired to reduce vibrations.

Bobrow et al [1] and Shiller [2] gave algorithms to determine the time-optimal motion for a robot manipulator along a given parametric path under actuator torque constraints. Farouki and Timar [3, 4] gave a piecewise-analytic expression of the optimal feedrate function for CNC machining with acceleration bounds. A smoothed feedrate function consistent with the acceleration bounds is generated afterward [5]. Following [3, 4], Zhang et al [6] gave a simplified feedrate planning method for quadratic B-splines and realized industrial CNC machining. Zhang et al [7] extended the above methods to the case of jerk bounds and gave

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

[†]Corresponding author.

a greedy feedrate planning algorithm. Yuan et al [8] extended the algorithms by including chord error bounds.

The methods mentioned above are called phase space analysis methods, because they use the velocity limit curve or surfaces in the u-$\dot{u}$-$\ddot{u}$ phase plane or space to obtain an optimal solution. However, these methods have two limitations. First, their computational expense is high, especially for tool pathes described by complex parametric functions, which lead to the solving of high degree equations. Second, the constraints in these methods are not easy to be generalized because time-optimality is difficult to prove for jerk bounds.

Another class of feedrate planning methods is based on numerical nonlinear programming algorithms. Lin et al [9] developed an algorithm to schedule the time intervals between each pair of adjacent knots as cubic splines such that the total traveling time is minimized subject to confined velocities, accelerations, and jerks. Altintas and Erkorkmaz [10] presented a quintic spline trajectory generation algorithm that produces continuous position, velocity, and acceleration profiles with confined tangential acceleration and jerk. They also developed a feedrate optimization technique in [11] for minimizing the cycle time with axis velocity, torque and jerk limits. Sencer-Altintas-Croft further proposed a nonlinear optimization based interpolation method under confined jerk and chord error for 5-axis CNC machines [12]. Dong et al [13, 14] gave discrete greedy algorithms with constraints of parametric velocity, acceleration, and jerk based on a series of single variable optimization subproblems. The sub-optimization structure makes it easy to deal with any state-dependent constraints. In [15], in order to reduce vibrations, Gasparetto et al considered a feedrate planning problem where the objection function is a linear combination of time and the jerk.

A third approach to feedrate planning is the direct sampling methods which compute the feedrate at every sampling time based certain strategies. Yeh and Hsu used a chord error bound to control the feedrate if needed and used a constant feedrate in other places [16]. Ernesto and Farouki [17], Feng et al [18], and Sun et al [19] proposed a method to generate chord error and acceleration limited velocity functions. Nam and Yang [20] proposed a recursive method to generate jerk limited velocity functions. Emami-Arezoo [22] and Lai et al [21] proposed time-optimal velocity planning methods with confined acceleration, jerk, and chord error by adjusting the velocity if any of the bounds is violated through backtracking. Lee et al gave a jerk limited velocity planning algorithm based trigonometric acceleration and deceleration profiles [23]. Fan et al proposed a method to generate jounce confined velocity functions [24].

In this paper, the problem of finding a time-optimal smooth feedrate function along a parametric tool path $\vec{r}(u), u \in [0,1]$ with a feedrate bound, an acceleration bound for each axis, and a chord error bound is considered. The main contribution of the paper is to give a computationally efficient and practical algorithm for the above problem. The algorithm consists of three major steps.

Firstly, the u-v phase plane is discretized into a grid by first dividing the tool path parameter interval $[0,1]$ into sub-intervals, and then, for each parameter knot u_i, dividing the feedrate interval $[0, V_{im}]$ into equal intervals of length Δv, where $V_{im} = V_{lim}(u_i)$ is the maximal feedrate at u_i obtained from the velocity limit curve $V_{lim}(u)$ defined in [1, 2, 3, 8]. Each point in the grid is of the form (u_i, v_j) where v_j is a possible feedrate at parameter value u_i.

Secondly, a maximal discrete velocity curve is generated, which is a sequence of points $(u_i, v_{n_i}), i = 0, \ldots, N$ in the grid such that either $(u_{i+1}, v_{n_{i+1}})$ is reachable from (u_i, v_{n_i}) with the maximal accelerations and has the largest velocity $v_{n_{i+1}}$, or $(u_{i+1}, v_{n_{i+1}}) = V_{im}$ and all (u_{i+1}, v_k) are not reachable from (u_i, v_{n_i}) with the maximal accelerations. We then prove that the solution of this algorithm is globally optimal under all the constraints when the discretization is sufficiently small.

Finally, a linear programming strategy is used to adjust the velocities around each acceleration discontinuity point locally to smooth the sequence of velocity points. Then the velocity points are fitted into a cubic spline to obtain a near-time-optimal smooth feedrate function.

The major advantage of the new algorithm is that near-time-optimal solutions can be found very efficiently. The computational complexity of the algorithm is shown to be $O(N/\Delta v)$ in terms of floating point arithmetic operations, where N is the number of segments to discrete the tool path and Δv is the length of the velocity intervals. It is easy to see that $N/\Delta v$ is proportional to the number of points in the grid.

The algorithm can be considered as a discrete version of the algorithms given in [1, 2, 3, 8]. Comparing to these methods, our discretized algorithm is much more efficient especially for tool paths described by complex parametric functions, since the new algorithm avoids the time consuming task of equation solving. Comparing to the feedrate planning methods based on the nonlinear programming, the new method can find the globally optimal solution with a nice computational complexity bound, while most existing methods based on nonlinear programming cannot guarantee to find the globally optimal solution. Furthermore, solving nonlinear programming problems could be slow in many cases. Comparing the direct sampling methods, our algorithm needs not backtracking, and hence is more efficient. The method is verified on a commercial CNC system to machine a cubic NURBS curve on a three-axis CNC machine.

The rest of this paper is organized as follows. Section 2 gives the description and theoretical analysis of the feedrate planning problem. Section 3 gives the discrete velocity search algorithm and proves the optimality of the algorithm. Section 4 gives the feedrate smoothing and fitting methods. Section 5 gives the experiment results. Section 6 concludes the paper.

2 Problem description and analysis

For brevity, the tool path is considered to be a plane parametric curve, which has at least C^1 continuity:

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

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

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

In the 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, the bounds on the x and y acceleration components are considered to be A_x, A_y respectively. Let $q(u) = v^2(u)$. Then the accelerations on the x and y axes are

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

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

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

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 [3].

Solve 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 $\alpha_x = \pm 1, \alpha_y = \pm 1$, to obtain

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

which are the four values of q in the four vertices. The maximal value of q at each $u \in [0,1]$ is then easy to obtain. Now the *x and y axes velocity limit curve* [1, 2, 3] is determined by the acceleration constraints on the x and y axes:

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

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

Secondly, the *chord error bound* ε is considered. 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, 8],

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

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

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

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}, the *velocity limit curve* (abbr. VLC) is obtained with all the constraints:

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

Note that a segment of $V_{lim}(u)$ could be used as the real velocity curve and such a segment is called *feasible* [8]. In general, a feasible segment is either V_{max} or V_N, where the corresponding constraint is 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 the 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

In this section, an algorithm is given to find a globally optimal discrete velocity curve under the given constraints.

3.1 Discretization of the phase plan

We will show how to divide the u-v phase plane into a grid. Firstly, the tool path is divided into N' segments of equal parametric length Δu. The knots are $i\Delta u, i = 0, \ldots, N'$, where $\Delta u = 1/N'$. For the accuracy of the algorithm, if $V_{lim}(i\Delta u)$ is local minimal, that is $V_{lim}(i\Delta u) < \min(V_{lim}((i-1)\Delta u), V_{lim}((i+1)\Delta u))$, then the parameter interval $[(i-1)\Delta u, (i+1)\Delta u]$ is further subdivided to ten equal sub-intervals. Denote the new knots by

$$u_i, i = 0, \ldots, N,$$

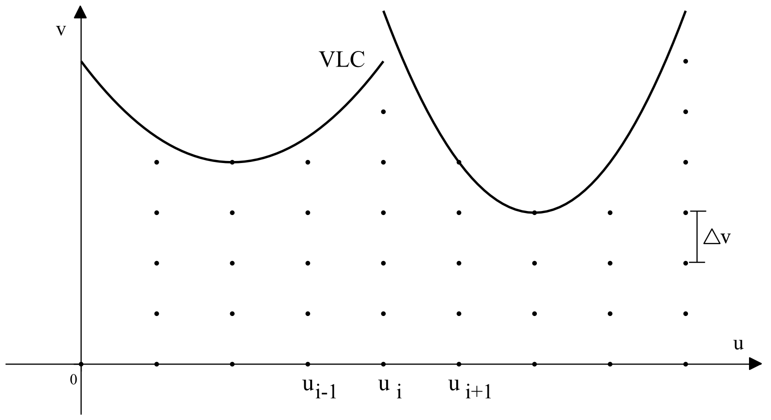


Figure 1: Discretization of parameter and velocity.

where N is the number of the final knots.

An appropriate Δv is chosen to discretize the value of velocity at each u_i. From (6), the maximal feasible discrete velocity at u_i is $V_{lim}(u_i)$. Divide the verlocity interval $[0, V_{lim}(u_i)]$ into $N_v(i)$ equal sub-intervals, where

$$N_v(i) = \lfloor \frac{V_{lim}(u_i)}{\Delta v} \rfloor, i = 0, \ldots, N. \tag{9}$$

The velocity at u_i will be chosen in a series of discrete values (see Fig.1):

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

Then the u-v phase plane is discretized into a grid and the number points in the grid is

$$\mathbb{N} = \sum_{0 \le i \le N} \{(N_v(i) + 1)\}. \tag{10}$$

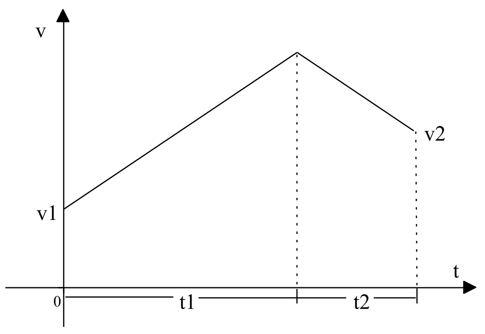
3.2 Velocity reachability

Before presenting the complete algorithm, it needs to give a subfunction $\mathrm{VR}(\mathrm{v_s}, \mathrm{u_s}, \mathrm{v_e}, \mathrm{u_e})$ which decides if the two neighboring points (u_s, v_s) and (u_e, v_e) in the grid 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).

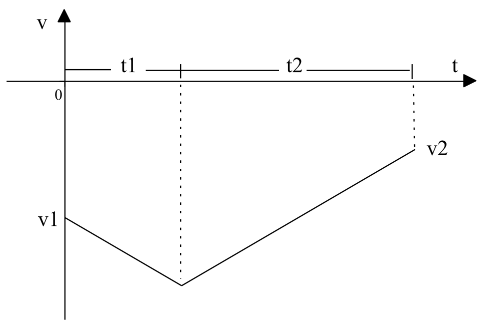
Firstly, the reachability on each axis is considered. Suppose that the velocity components (with signs) on an axis (x or y) of the two neighboring knots u_s, u_e are v_1, v_2, respectively, the projection distance (with sign) on this axis between the two knots is Δs, and the acceleration bound of this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

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

Because the distance between the two neighboring knots is quite small, the problem is simplified to be only with the acceleration bound of this axis here, which means that $V_{lim}(u)$



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



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

Figure 2: Two cases of acceleration profile between two consecutive knots.

is not achievable. Minimum-time path traversals with only acceleration limits generally involve a "Bang-Bang control" strategy. Then, it is easy to show that there has only two cases of acceleration profiles 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 $|\Delta s| \geq \frac{|v_2^2 - v_1^2|}{2A}$.

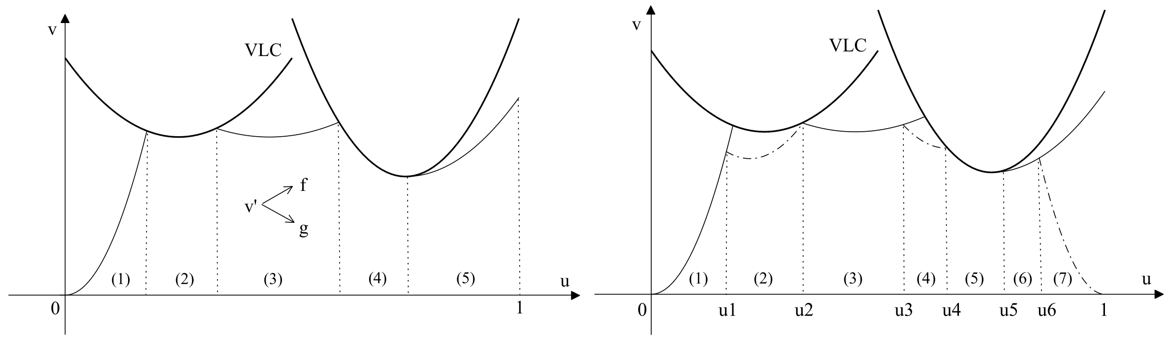
If v_1 and v_2 have the opposite signs, it means that there exists a point whose velocity on this axis is 0 between the two knots. The parameter u_m of this point can be obtained by solving $x'(u) = 0$ or $y'(u) = 0$ (the bisection method can be used between u_s and u_e). Then the projection distance 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 the reachability function on each axis is

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

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. Then the reachability function is

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



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

Figure 3: An illustration of Algorithm **DIS_VP**. The horizontal axis is the parameter u of the tool path. The vertical axis is the tangential velocity.

3.3 Feedrate planning algorithm

The algorithm consists of a forward direction pass and a reverse direction pass. During the forward pass, a trajectory is generated to search for the current 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 the complete algorithm is stated below:

Algorithm DIS_VP. Discrete velocity planning algorithm.
Input: $\vec{r}(u)\, u \in [0,1], V_{max}, A_x, A_y, \varepsilon, N', \Delta v$.
Output: A set of knots $u_0 = 0 < u_1 < \cdots < u_N = 1$ and the velocity v_i^* at u_i.

1. Pre-process: compute the knots u_i, $i = 0, \ldots, N$ as shown in the first paragraph of section 3.1 and the velocity interval number $N_v(i)$ at each u_i with (9).
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^*, $i = 0, \ldots, N$.

Fig.3 is used to illustrate Algorithm **DIS_VP**, 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 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 the velocity values at each u_i is obtained.

We now show how to chose the two parameters N' and Δv in the input of the algorithm.

The segments number N' will be chosen to be at least 10% of the number of interpolation steps along the tool path for the computational accuracy. It can be estimated as $S/(10V_{ave}T)$, 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 **DIS_VP** may not obtain a solution. Here a limit to Δv is given as below. In the forward pass, $v_i = 0$ leads to the bounds of v_{i+1} from the velocity reachability function:

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

Then

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

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

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

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

Let $\mathbb{N}$ be the number of points in the discretization grid of the u-v phase plane given in (10). Then the *computational complexity* of the algorithm is $O(\mathbb{N})$ in terms of floating-point arithmetic operations, since there are twice loop operations in the algorithm with $\mathbb{N}$ as the maximal number of loops, and in each step inside the loop it only needs to perform a fixed number of floating point arithmetic operations. It is clear that $N_v(i) \leq V_{max}/\Delta v$ and $\mathbb{N} \leq V_{max}N/\Delta v + N$. Since V_{max} is a fixed value, the computational complexity of the algorithm is $O(N/\Delta v)$. A more detailed analysis of Algorithm **DIS_VP** will be stated in the next subsection.

3.4 Optimality of the algorithm

In this subsection, it will be proved that Algorithm **DIS_VP** does obtain a globally optimal solution for problem (7) under constraints (8) as Δu and Δv approach to 0.

The method to prove the optimality is similar to that used in [1, 2, 3, 8]. The phase plane (u, v) and the VLC are used to construct a solution for the minimum time control problem. The algorithm is analyzed in the same phase plane and then the optimality is shown.

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

$$\begin{cases} f_1(u,v) \leq v' \leq g_1(u,v) \\ f_2(u,v) \leq v' \leq g_2(u,v) \end{cases} \tag{12}$$

Let $f = \max\{f_1, f_2\}, g = \min\{g_1, g_2\}$. Then the constraints (12) lead to

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

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, it can be seen that step 3 in Algorithm **DIS_VP** searches for the current maximal v' under all the constraints. Although only a finite number of values for the velocity is found, it can be considered that a *velocity curve* $v(u)$ is computed in this 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 $v' = g(u,v)$. 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 $f(u,v) \leq V'_{lim}(u) \leq g(u,v)$; or there has no solution in step 3, which means $V'_{lim}(u) \leq f(u,v) \leq g(u,v)$. Together with the two cases, $V'_{lim}(u) \leq g(u,v)$ satisfies 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 $v' = f(u,v)$; if the segment is equal to $V_{lim}(u)$, it satisfies $f(u,v) \leq V'_{lim}(u)$.

Now it is clear that the complete velocity curve consists of three types of segments (Fig.3(b) is used 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 $V'_{lim}(u) \leq g(u,v)$ and $f(u,v) \leq V'_{lim}(u)$. 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, it only needs to show that the velocity curve given by Algorithm **DIS_VP** is higher than any other velocity curve on the phase plane (u, v), which satisfies constraints (8). In order to prove this, it needs the following result.

Comparison Theorem (p.25, [27]): 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, it can be proven 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 **DIS_VP**.

Fig.3(b) is used to illustrate the proof. In segment (1) in Fig.3(b), there are $\hat{v}(0) = v(0)$ and

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

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

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

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

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

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

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, it can be shown that

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

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

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

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

Together with $\hat{v} \le v, u \in [u_4, u_5]$, it can be shown that $\hat{v} \le v, u \in [0, 1]$. From

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

then

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

which proves the optimality of the algorithm.

4 Feedrate smoothing and fitting

In this section, we will show how to obtain a near-time-optimal smooth velocity function from the sequence of velocity values obtained in section 3.

It is known that the velocity found with Algorithm **DIS_VP** is a discretization of the velocity curve found with the methods in [1, 2, 3, 8], which is piecewise differentiable. So its acceleration profile has discontinuities at a finite 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.

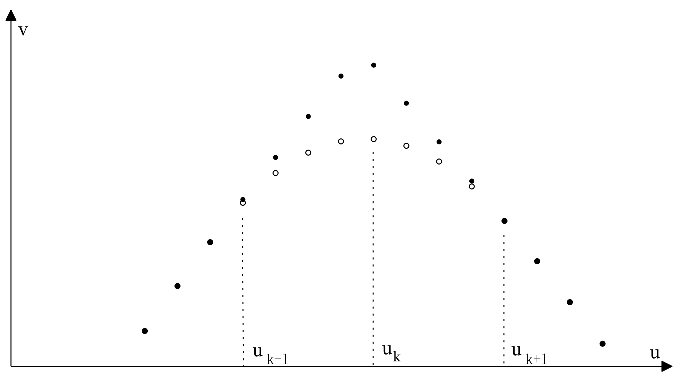


Figure 4: Feedrate smoothing around the velocity derivative discontinuity.

In the proof of optimality of Algorithm **DIS_VP** in section 3.4, it is shown 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 are easy to find throughout the process of Algorithm **DIS_VP**. The locating of the second kind of discontinuities is not direct. This case will be dealt with later.

First, the case that v' suddenly decreases at a point is considered. To find these points, it just needs to add steps 1', 3' and 6' after steps 1, 3 and 6 respectively in Algorithm **DIS_VP**:

1'. Initialization: $flag(i) = 0$, $i = 0, \dots, N$

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

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

In the forward pass of Algorithm **DIS_VP**, step 3' marks the points where the segments which satisfy $v' = g(u, v)$ intersect the VLC with $flag(i+1) = 1$ at u_{i+1}. In the reverse pass, step 6' marks the points where the segments which satisfy $v' = f(u, v)$ intersect the forward velocity curve and removes the marks where the forward velocity curve is updated by the reverse velocity curve.

Suppose $flag(k) = 1$, that is, v' suddenly decreases at parametric u_k. Choosing an appropriate small positive integer l, the velocities computed by Algorithm **DIS_VP** at parametric $u_{k-l}, \dots, u_{k+l}$ are decreased to obtain a finite rate of change of v' (or q') (see Fig.4). With the approximation of q' and q'' by

$$q_i' = \frac{q_i - q_{i-1}}{u_i - u_{i-1}}, q_i'' = \frac{q_i' - q_{i-1}'}{u_i - u_{i-1}},$$

solve the 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

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

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

Note that the second and third constraints in (15) are discrete version of constraints (2), which guarantee that the acceleration on each axis is confined. The fourth constraint in (15) is used for smoothing the velocity.

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'' bound J is large enough. Since for $i = k-l, \ldots, k+l+2$,

$$|q_i''| \le \frac{2\max_{k-l-1 \le i \le k+l+2} |q_i'|}{\min_{k-l \le i \le k+l+2}(u_i - u_{i-1})} \le \frac{4\max_{k-l-2 \le i \le k+l+2}\{q_i\}}{\min_{k-l-1 \le i \le k+l+2}(u_i - u_{i-1})^2},$$

the initial value of J can be chosen to be

$$\frac{4\max_{k-l-2 \le i \le k+l+2}\{v_i^{*2}\}}{\min_{k-l-1 \le i \le k+l+2}(u_i - u_{i-1})^2}.$$

Then J is decreased by using dichotomy and the above linear programming is solved 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 **DIS_VP** as above, a velocity adjusting interval $[u_{k-l_1}, u_{k+l_2}]$ can be chosen to cover this point, the preceding and the next v' discontinuity point. Then the linear programming above is solved together with the three discontinuities. If the discontinuity is on the VLC and its position is not known, the velocity adjusting interval can be chosen to cover this VLC segment as well.

The feedrate smoothing with linear programming will not increase the complexity of the algorithm since the linear programming has efficient polynomial-time algorithms [28] with complexity $O(n^{3.5})$, and the dimension of the problem $n = 2l + 1 \approx N/50$ is quite small compared with N. So, the total computational complexity of the whole procedure is the same as that of Algorithm **DIS_VP**.

For the interpolation process, the sequence of velocity points should be fitted into a feedrate function. Sencer et al [12] expressed the feedrate function in a cubic B-spline form. Ernesto and Farouki [17] specified the square of the feedrate as a Bernstein-form polynomial. In this paper, the A-spline [29] is adopted to fit the the velocity points into a smooth curve. It is based on a piecewise function composed of a set of polynomials, each of degree three

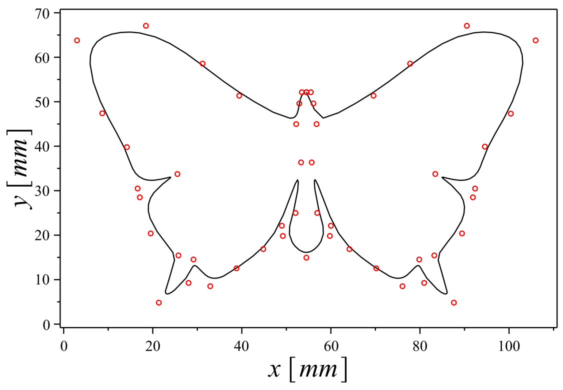


Figure 5: The butterfly curve and its control points.

at most. Then the feedrate $v(u)$ along the tool path $\vec{r}(u)$ is obtained, which is a cubic spline with C^1 continuity. With an acceptable curve fitting error, the feedrate function is near-time-optimal.

We finally remark that a time-optimal smooth velocity function to the feedrate planning problem under confined acceleration and chord error does not exist. In Section 3.4, it is shown that a time-optimal velocity function exists under confined acceleration and chord error, which achieves maximal value at any point. Also see [1, 2, 3, 8]. When we want to smooth the time-optimal solution, at any discontinues point, the velocity function can be approximated by a smooth one with any precision and there does not exists an optimal one. See Figure 4 for an illustration. Therefore, a near-time-optimal solution is the best we can have for the problem.

5 Experimental results

In this section, we will present the experimental result. The butterfly curve from [25] is used to illustrate the algorithm. It is a cubic NURBS curve (see Fig.5) with 51 control points and has C^2 continuity since each knot has multiplicity 1.

The constraints are set to be $V_{max} = 120mm/s, A_x = A_y = 800mm/s^2, \varepsilon = 1\mu m, T = 2ms$, where T is the sampling period of the CNC machine. For discretization, $N' = 300, 500, 1000$ and $\Delta v = 0.01, 0.02mm/s$ are chosen respectively in this example. The executing times of Algorithm **DIS_VP** with different N' and Δv are listed in Table 1 (CPU: Intel Core2 Duo, 2.93GHz; programming software: Microsoft Visual C++ 6.0). From Table 1, we can see that the executing time of Algorithm **DIS_VP** is approximately proportional to $O(N'/\Delta v)$, which confirms the computational complexity analysis of the algorithm in the last paragraph of section 3.3.

The discretization with $N' = 1000, \Delta v = 0.01mm/s$ is adopted to show the results of the algorithm. The total number of refined tool path segments is $N = 1452$ in Algorithm **DIS_VP**. In Fig.6, the dotted curve is the VLC computed by (6), and the solid curve is the optimal velocity given by Algorithm **DIS_VP**.

Fig.7 shows the detail of the velocity smoothing given by linear programming in section

Table 1: Executing times of Algorithm **DIS_VP** with different N' and Δv.

N'	300	300	500	500	1000	1000
Δv (mm/s)	0.02	0.01	0.02	0.01	0.02	0.01
Velocity Planning (s)	0.043	0.085	0.067	0.134	0.128	0.255

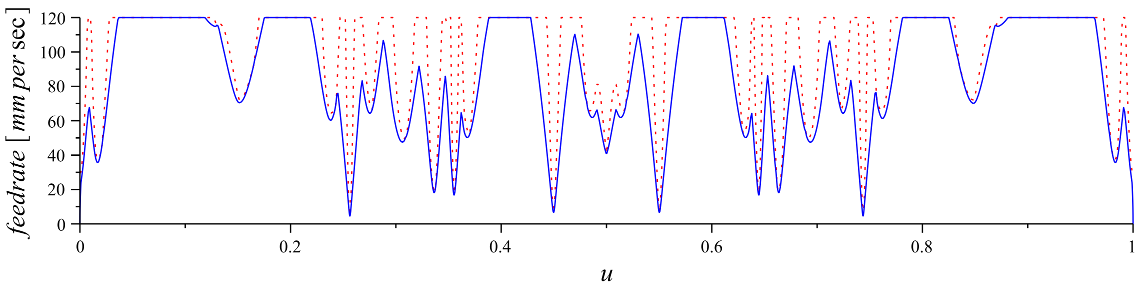


Figure 6: Dotted curve: VLC; Solid curve: velocity given by Algorithm **DIS_VP**.

4. The dotted curve is the original optimal velocity. The solid curve is the smoothed velocity.

Then the velocity points sequence is fitted into a cubic spline with 229 cubic polynomials. In Fig.8(a)(b), the points are the sequence of velocity points, and the solid curve is the fitted velocity curve. Fig.8(b) shows the detail of Fig.8(a).

The CNC system used in the experiment is GJ-210M (see Fig.9), which is a commercial product of Shenyang Lantian CNC Corporation. Since the butterfly curve consists of 48 segments of cubic rational curves, together with the 229 segments of the feedrate function, there are at most 276 segments with different curve expressions or feedrate functions. A new

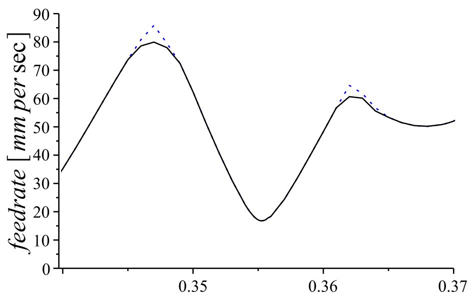
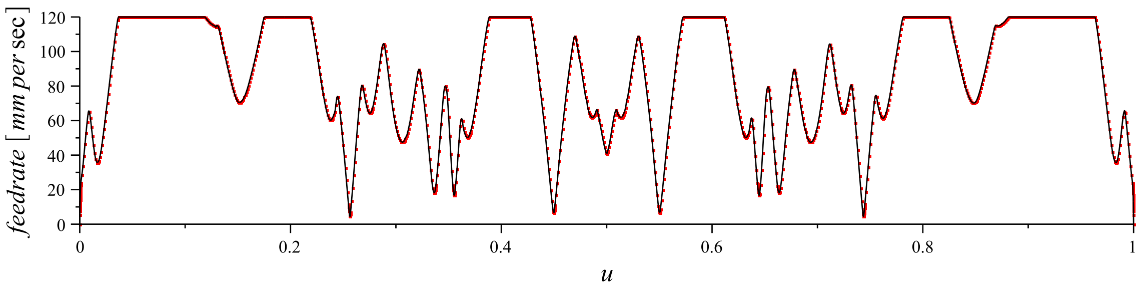
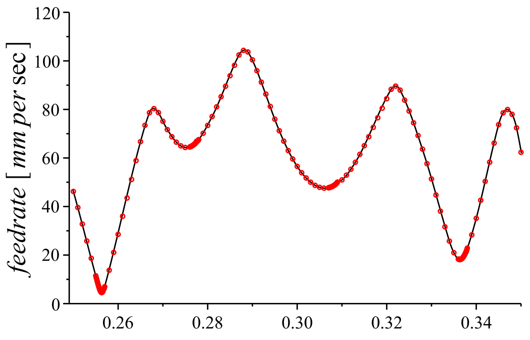


Figure 7: Dotted curve: the optimal velocity; Solid curve: the smoothed velocity.



(a)



(b)

Figure 8: The velocity points and the fitted velocity curve.

G-code is designed for cubic NURBS curves in the CNC system GJ-210M:

$$\begin{array}{lllll} \text{G66.5} & x_0 & x_1 & x_2 & x_3 \\ & y_0 & y_1 & y_2 & y_3 \\ & d_0 & d_1 & d_2 & d_3 \\ & v_0 & v_1 & v_2 & v_3 \\ & u_1 & u_2 & & \end{array}$$

which means on the segment $[u_1, u_2]$, the curve is

$$x(u) = \frac{x_0 + x_1 u + x_2 u^2 + x_3 u^3}{d_0 + d_1 u + d_2 u^2 + d_3 u^3}, y(u) = \frac{y_0 + y_1 u + y_2 u^2 + y_3 u^3}{d_0 + d_1 u + d_2 u^2 + d_3 u^3}$$

and the feedrate function is

$$v(u) = v_0 + v_1 u + v_2 u^2 + v_3 u^3.$$

By employing Taylor's expansion of $u(t)$ at $t = t_k$ and neglecting high-order terms, the second-order Taylor interpolation algorithm [18, 24] is adopted in the controller:

$$u_{k+1} = u_k + \frac{v}{\sigma}(u_k)T + (\frac{v}{\sigma})'\frac{v}{\sigma}(u_k)\frac{T^2}{2}. \tag{16}$$

Detailed analysis of parameter computation can be found in [25, 26].

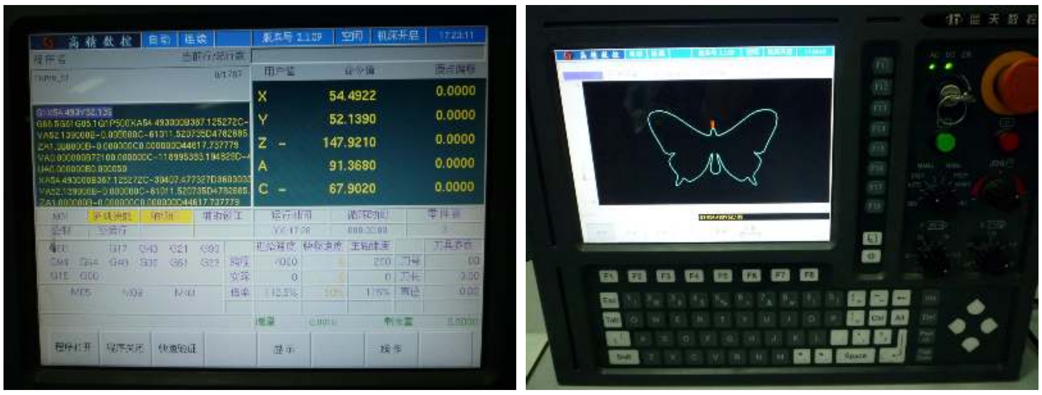


Figure 9: The CNC system used in the experiment.

Fig.10 shows the simulation results: the feedrate, the x and y accelerations, and the chord error, respectively. The machining time (5.17 s) is much shorter than the machining time (8.05 s) given by the algorithm proposed in [23] for the same curve and velocity, acceleration and chord error constraints. It should be noted that even though the real jerk constraint has not been considered in this paper, the acceleration on each axis is continuous.

6 Conclusion

The algorithm proposed in this paper is computationally more efficient than the analytical solving methods used in the phase plane analysis approaches when the parametric functions for the tool path are complex. Computing the expression of the VLC and switching points are avoided. It also does not need to integrate the accelerations. The constraints are more easy to be generalized in this algorithm. This discrete velocity search algorithm just need to compute a velocity reachability function at each step, and the computational complexity of the algorithm is $O(M/\Delta v)$ in terms of floating-point arithmetic operations. Comparing with the existing discrete algorithms using nonlinear programming techniques, this algorithm can give time-optimal solution with confined acceleration and chord error. Using a linear programming scheme and velocity curve fitting to remove the sudden jumps occurring in the acceleration of the time-optimal feedrate solution, a near-time-optimal solution with continuous accelerations is obtained. Simulation and experiment results show the feasibility and applicability of the feedrate planning algorithm.

References

[1] Bobrow JE, Dubowsky S, Gibson JS. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research* 1985; 4(3):3-17.

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

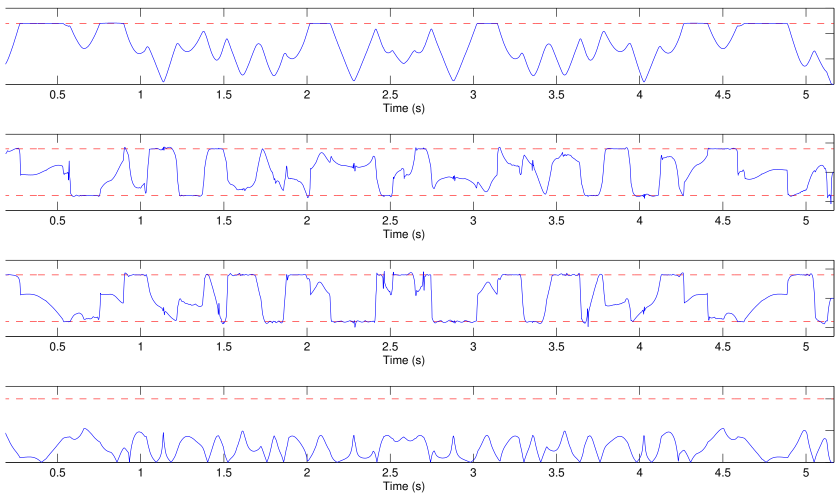


Figure 10: The feedrate, acceleration on each axis and chord error.

[3] Timar SD, Farouki RT, Smith TS, Boyadjieff CL. Algorithms for time-optimal control of CNC machines along curved tool paths. *Robotics and Computer-Integrated Manufacturing* 2005; 21:37-53.

[4] Timar SD, Farouki RT. Time-optimal traversal of curved paths by Cartesian CNC machines under both constant and speed-dependent axis acceleration bounds. *Robotics and Computer-Integrated Manufacturing* 2007; 23(2):563-579.

[5] Boyadjieff CL, Farouki RT, Timar SD. Smoothing of Time-Optimal Feedrates for Cartesian CNC Machines. *Mathematics of Surfaces XI*, LNCS, Springer Berlin, 2005; 3604:84-101.

[6] Zhang M, Yan W, Yuan CM, Wang DK, Gao, XS. Curve fitting and optimal interpolation on CNC machines based on quadratic B-splines. *Science China - Information Sciences* 2011; 54(7):1407-1418.

[7] Zhang K, Yuan CM, Gao XS, Li HB. A greedy algorithm for feedrate planning of CNC machines along curved tool paths with confined jerk. *Robotics and Computer-Integrated Manufacturing* 2012; 28:472-483.

[8] Yuan CM, Zhang K, Fan W, Gao, XS. Time-optimal interpolation for CNC machining along curved tool paths with confined chord error. *MM Research Preprints* 2011; 30:57-89.

[9] Lin CS, Chang PR, Luh JYS. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Trans. Robot. Autom.* 1983; AC-28:1066-1074.

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

[11] Altintas Y, Erkorkmaz K. Feedrate Optimization for Spline Interpolation In High Speed Machine Tools. *CIRP Annals - Manufacturing Technology* 2003; 52(1):297-302.

[12] Sencer B, Altintas Y, Croft E. Feed optimization for five-axis CNC machine tools with drive constraints. *International Journal of Machine Tools and Manufacture* 48 (2008) 733-745.

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

[14] Dong JY, Ferreiraa PM, Stori JA. Feedrate optimization with jerk constraints for generating minimum-time trajectories. *International Journal of Machine Tools and Manufacture* 2007; 47:1941-1955.

[15] Gasparetto A, Lanzutti A, Vidoni R, Zanotto V. Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning. *Robotics and Computer-Integrated Manufacturing* 2012; 28:164-181.

[16] Yeh SS, Hsu PL. Adaptive-feedrate interpolation for parametric curves with a confined chord error. *Computer-Aided Design* 34(2002)229-237.

[17] Ernesto CA, Farouki RT. High-speed cornering by CNC machines under prescribed bounds on axis accelerations and toolpath contour error. *Int J Adv Manuf Technol* 2012; 58:327-338.

[18] Feng J, Li Y, Wang Y, Chen M. Design of a real-time adaptive NURBS interpolator with axis acceleration limit. *Int J Adv Manuf Technol* 2010; 48:227C241.

[19] Sun Y, Jia Z, Ren F, Guo D. Adaptive feedrate scheduling for NC machining along curvilinear paths with improved kinematic and geometric properties. *Int J Adv Manuf Technol* 2008; (36):60C68.

[20] Nam SH, Yang MY. A study on a generalized parametric interpolator with real-time jerk-limited acceleration. *Computer-Aided Design* 36, 27-36, 2004.

[21] Lai JY, Lin KY, Tseng SJ, Ueng WD. On the development of a parametric interpolator with confined chord error, feedrate, acceleration and jerk. *Int J Adv Manuf Technol* 2008; 37:104-121.

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

[23] Lee AC, Lin MT, Pan YR, Lin WY. The feedrate scheduling of NURBS interpolator for CNC machine tools. *Computer-Aided Design* 2011; 43:612-628.

[24] Fan W, Gao XS, Yan W, Yuan CM. Interpolation of parametric CNC machining path under confined jounce. *Int J Adv Manuf Technol* 2012; DOI 10.1007/s00170-011-3842-0.

[25] Yau HT, Lin MT, Tsai MS. Real-time NURBS interpolation using FPGA for high speed motion control. *Computer-Aided Design* 2006; 38:1123-1133.

[26] Wu J, Zhou H, Tang X, Chen J. A NURBS interpolation algorithm with continuous feedrate. *Int J Adv Manuf Technol* 2012; 59:623C632.

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

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

[29] Akima MH. A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the Association for Computing Machinery* 1970; 17(4):589-602.