Reference trajectory modification based on spatial iterative learning for contour control of 2-axis NC systems

Article  (Accepted Version)


This version is available from Sussex Research Online: http://sro.sussex.ac.uk/id/eprint/89979/

This document is made available in accordance with publisher policies and may differ from the published version or from the version of record. If you wish to cite this item you are advised to consult the publisher’s version. Please see the URL above for details on accessing the published version.

Copyright and reuse:
Sussex Research Online is a digital repository of the research output of the University.

Copyright and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable, the material made available in SRO has been checked for eligibility before being made available.

Copies of full text items generally can be reproduced, displayed or performed and given to third parties in any format or medium for personal research or study, educational, or not-for-profit purposes without prior permission or charge, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

http://sro.sussex.ac.uk
Reference Trajectory Modification based on Spatial Iterative Learning for Contour Control of 2-axis NC Systems

Jiangang Li, Member, IEEE, Yiming Wang, Yanan Li*, Member, IEEE, and Wenshu Luo

Abstract—Contour error is a main factor that affects the quality of products in numerical control (NC) machining. This paper presents a contour control strategy based on digital curves for high-precision control of computer numerical control (CNC) machines. A contour error estimation algorithm is presented for digital curves based on a geometrical method. The dynamic model of the motion control system is transformed from time domain to space domain because the contour error is dependent on space instead of time. Spatial iterative learning control (sILC) is developed to reduce the contour error, by modifying the reference trajectory in the form of G code. This allows system improvement without interference of low-level controllers so it is applicable to many commercial controllers where interpolators and feed-drive controllers cannot be altered. The effectiveness of this method is verified by experiments on a NC machine, which have shown good performance not only for smooth trajectories but also for large curvature trajectories.

Index Terms—Contour error control, iterative learning control, CNC machines.

I. INTRODUCTION

In CNC machining, the machining path is a result of coordinated multi-axis motion and the machining quality of a product is usually evaluated by the contour error (the geometric deviation of the actual contour from the desired one). Therefore, reducing the contour error is an important task of multi-axis motion control systems [1], [2].

In order to improve the accuracy of contour tracking in CNC machining, a lot of research effort is devoted to feed-forward/feedback control strategies to improve the accuracy of single-axis tracking, thereby indirectly improving the accuracy of motion control systems. For example, zero phase error tracking controller proposed in [3] achieved tracking control by eliminating the zero poles in the system transfer function. However, the main cause of the contour error is the dynamic mismatch between the axes, so simply improving the tracking accuracy of a single axis does not necessarily lead to a reduction in the contour error [4], [5]. In order to solve this problem for multi-axis motion systems, cross-coupling control (CCC) was proposed in [6], [7] where the coupling effects of each axis on others was studied. Researchers have since then followed this pioneering work and improved its performance by focusing on two major issues: contour error estimation (CEE) and contour error control (CEC) [8], [9].

CEE is a trivial task for a linear trajectory, as one can compute the real contour error. However, a trajectory in CNC machining is usually a free curve, for which various methods were developed to estimate the contour error. A free curve contour error estimation model based on tangent was proposed in [10]. A local approximation method was proposed in [11], which has a large error when the tracking error is large. In [12], a contour error estimation method was proposed based on position backtracking, which is in the form of small line segments. However, the outputs of a CNC system are usually digital curves, namely a series of discrete data points, so extensive curve-based analysis is infeasible. Moreover, it is not reasonable to estimate the contour error of the curve by linear interpolation. In [13], a contour error prediction and compensation method was developed for five-axis systems.

A lot of effort has been also made for CEC. In [14], the contour error was reduced by modifying feedback in combination with integral sliding mode control. In [15], the traditional CCC scheme was improved for CEC. In [16], position-loop feedforward control was studied to reduce the contour error. In [17], a multi-input multi-output linear parameter-varying feedback controller was proposed for contour error minimization in CNC machines. In [18], a discrete-time fractional-order sliding-mode CEC was proposed to improve the contour tracking accuracy. However, these methods at the control level are not easy to implement, especially for commercial controllers which are not open to users. Moreover, contour control at the control level may have adverse effects on the controller and may change the original speed planning. In [19], [20], trajectory planning was modified to achieve reduction of the contour error, but the modification of the trajectory planning is limited by many constraints. In [21], an error compensation approach of parametric programming was developed that deals with the deformation of the machining system caused by radial cutting force in turning, and the generated error was compensated to the tool path. This approach works well with many commercial controllers for which the interpolators and the feed-drive controllers are not allowed to be altered, and it is simple to implement. In order to further improve the contour tracking of CCC, especially in mass production, the strategy of cross iteration learning coordination control (CCILC) was developed, which combines iterative learning.
control (ILC) and CCC [22]. A novel algorithm that integrates ILC with empirical mode decomposition (EMD) was proposed to improve learning process [23].

As acknowledged by some of aforementioned works, ILC is a suitable method to eliminate the effects caused by repetitive interference in the CNC machining. It works in an iterative manner to gradually counteract the effects of repetitive disturbances. In [24], ILC was used to update control signals based on path errors and the time elapsed along a trajectory. While the ILC used in most of the existing works was based on a time period, the contour error is naturally defined in space. Therefore, any delay in one axis will introduce an error to the contour error compensation. In the field of ILC, spatial iterative learning control (sILC) was introduced for systems with spatial periodicity. In [25], [26], the standard temporal ILC algorithm was formulated with spatial variables and a sILC framework was developed based on 2D spatial convolution. In [27], sILC was developed for a class of high-order nonlinear motion systems in the presence of both parametric and nonparametric uncertainties, but it was also implemented in the control level.

Based on above discussions, this paper will introduce a new method to compute the contour error and to modify the reference trajectory without modification of the controller. We will use Hermite spline interpolation to get a cubic spline that is close to a free curve and then develop a geometrical method to estimate its contour error. This method does not require complex computation and is easy to implement, while it ensures a high estimation accuracy even for large curvature trajectories. We will develop sILC with modification of reference trajectory achieved by modifying G code. This method allows system improvement without interference of the low-level controllers. This property is favourable as for many commercial controllers the interpolators and feed-drive controllers are not allowed to be altered. We will show that this learning guarantees convergence of the contour error when the iteration number is large enough, by using a composite energy function. Finally, we will present experimental results to verify the validity of our idea.

In the rest of the paper, we first introduce an improved contour error estimation method in Section II. In Section III, we propose a contour control method for modifying the reference trajectory based on sILC. The convergence of contour error is proved in theory. In Section IV, we present the results of the experiment under different conditions for comparison. At last, we draw conclusions of this paper and discuss possible future works.

II. CONTOUR ERROR ESTIMATION

In this section, the approach of contour error estimation for digital curves will be described. First, the Hermite spline interpolation method will be used to deal with a digital curve, so that a better approximation of the free curve can be achieved and mathematical analysis could be carried out. Then, the contour error is estimated based on closest point search and geometrical analysis.

A. Digital curve processing

The original reference trajectory used to calculate the contour error is obtained by discretization of a parametric curve, and the actual trajectory is obtained by machine tool sampling. Therefore, both of them are digital curves.

In order to estimate the contour error more accurately, splines are used to construct these curves. In particular, cubic splines are used as they make a reasonable compromise between the flexibility and the computing speed [28]. According to different methods to determine the coefficient, cubic splines have three forms: cubic natural splines, Hermite splines and Cardinal splines. In this work, the Hermite spline method is selected as any part of the curve only depends on the local control point.

A general Hermite spline can be described as

$$P(t) = A_t t^3 + B_t t^2 + C_t t + D_t \quad (l \in [0,1])$$

where $P(l) = [x(l) \ y(l) \ z(l)]$ represents position vector with $x(l), y(l), z(l)$ as coordinates and $l$ as spline parameter. $A_t, B_t, C_t$ and $D_t$ are coefficients to be determined in the following.

Suppose the starting point and ending point of a curve segment are respectively $P_0$, $P_1$, and the tangent vectors of the curve segment at the two points are respectively $P'_0$, $P'_1$. Then, the following equations can be obtained that provide solutions of the four coefficients in Eq. (1):

$$\begin{align*}
P_0 &= A_t l^3 + B_t l^2 + C_t l + D_t \quad (l = 0) \\
P_1 &= A_t l^3 + B_t l^2 + C_t l + D_t \quad (l = 1) \\
P'_0 &= C_t + 2B_t l + 3A_t l^2 = C_t \quad (l = 0) \\
P'_1 &= C_t + 2B_t l + 3A_t l^2 = C_t + 2B_t + 3A_t \quad (l = 1)
\end{align*}$$

Let us write Eq. (2) in a form of matrices, as below

$$\begin{bmatrix} P_0 \\ P_1 \\ P'_0 \\ P'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} A_t \\ B_t \\ C_t \\ D_t \end{bmatrix}$$

(3)

Then, with solved $A_t$, $B_t$, $C_t$ and $D_t$, we can obtain the matrix expression for the Hermite spline curve as below

$$P(t) = \begin{bmatrix} t^3 & t^2 & t \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 \\ -3 & 3 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P'_0 \\ P'_1 \end{bmatrix}$$

(4)

Finally, we can obtain the expression of the k-th segment curve as

$$P(l) = P_k(2l^3 - 3l^2 + 1) + P_{k+1}(-2l^3 + 3l^2) + P'_k(l^3 - 2l^2 + l) + P'_{k+1}(l^3 - l^2)$$

(5)

B. Estimation algorithm

The estimation of contour error is based on digital curve processing in the previous subsection, followed by searching the closest point and Frenet coordinate. The detailed process is given as follows.

Consider a trajectory $\Gamma$ shown in Fig. 1, where $Q(k)$ represents the actual tool position at time $k$, $P(k)$ represents
In a process of repetitive iterations, \( P(k) \) is invariant, so the contour error is defined based on the deviation from \( P(k) \). The idea is to compute the contour error at the same position in different iterations, so that we can implement contour control at the right place. When we compute the contour error at the position \( P(k) \), the first step is to find the actual position point closest to \( P(k) \). Then, we can use the following equation to compute the distance between \( P(k) \) and actual position, and determines the point \( Q(k + n) \) that is closest to \( P(k) \):

\[
L = \sqrt{(x_{pm} - x_{qk})^2 + (y_{pm} - y_{qk})^2}
\]  

(6)

Note that the distance range of search is within that between the last pedal point \( P_c(k - 1) \) and the ending point. Therefore, there are two cases to find the closest point. When \( P_c(k - 1) \) is the closest point, we have one segment \( L_1 \) to compute the pedal point. When \( Q(k + n) \) is the closest point, we have two segments \( L_1 \) and \( L_2 \) to compute the pedal point. In the next step, we use the closest point to compute the new pedal point based on trigonometry as shown in Fig. 2. The calculation process is divided into the following three cases:

1. If \( \alpha_1 < 90^\circ \) or \( \alpha_2 < 90^\circ \), the pedal point is on the segment. We can compute \( P_c(k) \) by the following two equations in segment \( L_1 \):

\[
\left\{ \begin{array}{l}
|P(k)P_c(k)| = |P(k)Q(k + n)| \sin \alpha_1 \\
|Q(k + n)P_c(k)| = |P(k)Q(k + n)| \cos \alpha_1
\end{array} \right.
\]  

(7)

\[
P_c(k) = Q(k + n) + \frac{|Q(k + n)P_c(k)|}{|Q(k + n)Q(k + n + 1)|} Q(k + n)Q(k + n + 1)
\]  

(8)

(2) If \( \alpha_1 = 90^\circ \) or \( \alpha_2 = 90^\circ \), we can get \( Q(k + n) \) or \( Q(k + n + 1) \) as the pedal point \( P_c(k) \).

(3) If \( \alpha_1 > 90^\circ \) or \( \alpha_2 > 90^\circ \), there is no pedal point in segment \( L_1 \), so we need to compute the pedal point in \( L_2 \); if there is no pedal point in \( L_2 \), the point \( Q(k + n) \) can be regarded as the contour error point.

Based on the method of curve interpolation, we can get the equation of a curve between two points as below:

\[
\left\{ \begin{array}{l}
x_i(l) = A_{i1}l^3 + B_{i1}l^2 + C_{i1}l + D_{i1} & l \in [0, 1] \\
y_i(l) = A_{i2}l^3 + B_{i2}l^2 + C_{i2}l + D_{i2} & l \in [0, 1]
\end{array} \right.
\]

(9)

Moreover, we compute the spline parameter \( l \) as below:

\[
l = \left| \frac{Q(k + n)P_c(k)}{Q(k + n)Q(k + n + 1)} \right|
\]

(10)

From Eqs. (9) and (10), we can get the coordinate \( P_m(k) = (x_{mk}, y_{mk}) \) in the spline curve, and then a coordinate system is established with this point as the origin, the tangent as X-axis and normal as Y-axis, and the contour error as the Y-coordinate.

From Eq. (9), we can get the tangent vector \( \vec{\tau} \) at the point \( P_m(k) \), then the contour error \( e = ||P_m(k)Q(k)|| \cdot \vec{\tau} \). At the same time, we can get the contour error point \( P_e(k) \).

C. Verification of contour error estimation

In order to verify the correctness of our contour error estimation method, we use a circular trajectory with radius of 1.5mm to conduct an experiment. The real contour error of the circular trajectory can be computed as follows:

\[
e_r = |\vec{x}|-r
\]

(11)

where \( \vec{x} \) is the actual position vector with the center of circle as the origin and \( r \) is the radius of circle.

The results with the above analytic method and our computation method are shown in Fig. 3 (a), where we can find that the contour errors computed by two methods almost overlap. The difference between them is shown in Fig. 3 (b), with the maximum of the difference as \( 5.9 \times 10^{-5} \)mm, much smaller than the contour error itself. These results indicate that our contour error computation method achieves accurate estimation.
III. DYNAMICS ANALYSIS AND CONTOUR ERROR CONTROL

In this section, the control structure is established, the dynamics model of a CNC system is analyzed and the approach of adjusting the reference trajectory is explained. For a motion control system with two or three axes, there are four parts in the control framework, as shown in Fig. 4. The first part represents two-axis or three-axis plant and inner controllers of individual axes that guarantee closed-loop stability. This part receives G-code generated by digital curves in G01 lines. The inner controller would not affect the design of CEE and CEC as long as the stability of $P_i$ is guaranteed. The second part is CEE, which computes the contour error based on the desired trajectory and actual trajectory which are represented by discrete data points through sampling. The third part is error analysis where the root mean square error and maximum error are used to observe the error variation in the iterative process and verify whether the machining requirements have been met. The last part is CEC to determine the error compensation and generate a new reference trajectory. By going through these four parts, a discrete reference trajectory is obtained to generate new G-code based on G01.

A. CNC system model

Fig. 5(a) shows a schematics of the dynamics model of the motion control system. In order to analyze this system, it can be simplified as a mechanical model shown in Fig. 5(b), which can be written as:

$$M \ddot{x}(t) + B \dot{x}(t) + F = u + d$$

(12)

where $x$ is the position, $u$ is the control input and $d$ is the unknown disturbance. $M$, $B$, $F$ are the inertia, coefficient of viscous friction and coulomb friction, respectively.

For the following ILC design, we assume that the disturbance model is

$$d = -\theta_i$$

(13)

where $\theta_i(s) = \theta_{i-1}(s)$ with $s$ representing the displacement that will be explained later. This model indicates that the disturbance is repetitive in the displacement.

We design a controller including two parts, i.e., $u = u_1 + u_2$ where $u_1$ is used to compensate for the system dynamics, as below

$$u_1 = M \ddot{x}_e(t) + B \dot{x}_e(t) + F$$

(14)

where $x_d$ is the desired trajectory, $e$ is tracking error and $L$ is a positive definite matrix. $u_2$ is used to introduce feedback, as below

$$u_2 = -K[\dot{e} + L(x - x_r)] = -K[e_v + L(x_d - x_r)]$$

(16)

where $e_v = \dot{e} + Le$, $x_r$ is the reference trajectory and $K$ is the gain of position control loop.

By substituting (14) and (16) into (12), we obtain the error dynamics

$$M \ddot{e}_v(t) + (B + K)e_v(t) = -KL(x_d - x_r) + d$$

(17)

From the above equation, we find that if the disturbance $d = 0$, choosing the reference trajectory $x_r = x_d$ will lead to $e_v = 0$ and thus $e = 0$ according to the definition of $e_v$. Since $d \neq 0$, we need to design $x_r$ so that $e = 0$ is still guaranteed.

B. Transformation from time domain to spatial domain

As we assume that the disturbance $d$ is repetitive in displacement, we consider transferring the system dynamics from time domain to spatial domain. First, let us discuss the relationship between the spatial and temporal coordinates. As have been defined before, $t$ is the time, $s$ is the displacement of the system along the path and $v = ds/dt$ is the speed of the system in this direction. Therefore, we can obtain

$$\frac{d}{dt} = \frac{d}{ds} \frac{ds}{dt} = \frac{d}{ds}v$$

(18)

The spatial differentiator, or the $\nabla$ operator [29], is defined below and linked to the temporal differentiator:

$$\nabla = \frac{d}{ds}$$

(19)
In order to facilitate the conversion between $t$ and $s$, let us analyze the relationship between the temporal coordinate $t$ and spatial coordinate $s$. From $ds = vd t$ we have $s = \int^t_0 v(\tau)d\tau$. When the system’s speed $v > 0$, $s$ is a strictly increasing function of $t$, hence the relationship between $t$ and $s$ is bijective. The function $s = f(t)$ is analytic and the inverse function $t = f^{-1}(s)$ exists globally. Therefore as a variable, $v(t)$ can also be expressed as a spatial function $v(f^{-1}(s))$.

Based on above discussions, we can transform the error dynamics in Eq. (17) into the spatial domain, as below

$$
Mv \bigtriangledown e_v(s) + (B + K)v = -KL[x_d(s) - x_r(s)] + d
$$

(20)

where $x_d(s)$ is the desired position and $x_r(s)$ is the reference position in space and $e_v(s)$ is the error in space domain, including the contour error $e(s) = x(s) - x_d(s)$.

### C. Reference modification

According to (20), we can make the contour error $e(s)$ converge to 0 if $-KL[x_r - x_d] = d$. Therefore, according to (13), in the $i$-th iteration we design $x_r$ as below

$$
x_r,i(s) = x_d(s) + \frac{1}{KL} \hat{\theta}_i
$$

(21)

which is equivalent to

$$
x_r,i(s) = x_{r,i-1}(s) + \delta \hat{\theta}_i
$$

(22)

where $\hat{\theta}_i$ is the estimate of $\theta_i$ and is updated iteratively as below

$$
\delta \hat{\theta}_i = \hat{\theta}_i - \hat{\theta}_{i-1} = Qe_v,i(s)
$$

(23)

where $Q$ is a positive definite gain matrix.

According to (13), (20) and (21), in the $i$-th iteration we have

$$
Mv \bigtriangledown e_v,i + (B + K)e_v,i = \tilde{\theta}_i
$$

(24)

where $\tilde{\theta}_i = \hat{\theta}_i - \hat{\theta}_i$. Furthermore, we can get

$$
M \bigtriangledown e_v,i + \frac{1}{v} Be_v,i + \frac{1}{v} Ke_v,i = \frac{1}{v} \hat{\theta}_i
$$

(25)

From this equation, we can easily see that if $\tilde{\theta}_i = 0$ then $e_v,i \to 0$ and thus $e_i \to 0$, indicating that the contour error converges to 0. Therefore, the objective of the adaptation of $\hat{\theta}_i$ is to minimize $\tilde{\theta}_i$, which can be carried out through minimizing the cost function as below

$$
J_c(s) = \frac{1}{2} \int^s_{s-S} \frac{1}{v} (\tilde{\theta}_i^T Q^{-1} \tilde{\theta}_i) \, dt
$$

(26)

where $S$ is the displacement of the system in one iteration. Correspondingly, the reduction of the contour error can be obtained by minimizing the cost function

$$
J_c(s) = \frac{1}{2} \hat{\theta}_i^T Me_v,i
$$

(27)

Consequently, we consider a combined cost function

$$
J = J_c + J_e
$$

(28)

that will be shown to decrease as the iteration number increases.

### D. Convergence analysis

In particular, we consider the space derivative of $J_e$, as below

$$
\nabla J_e(s) = e_v^T M \nabla e_v,i + e_v,i \frac{1}{v} Me_v,i
$$

$$
\nabla J_e(s) = e_v^T M \nabla e_v,i + e_v,i \frac{1}{v} Be_v,i
$$

(29)

where we use the skew-symmetric property [30], i.e.

$$
e_v^T \hat{B} e_v,i = 2e_v^T Be_v,i
$$

(30)

Considering Eq. (25), the above equation can be rewritten as

$$
\nabla J_e(s) = e_v^T [M \nabla e_v,i + \frac{1}{v} Be_v,i]
$$

$$
\nabla J_e(s) = e_v^T [\frac{1}{v} \hat{\theta}_i - \frac{1}{v} Ke_v,i]
$$

(31)

Let us then consider the difference between $J_e$ of two consecutive iterations as below

$$
\Delta J_e = J_e(s) - J_e(s - S) = \frac{1}{2} \int^s_{s-S} \frac{1}{v} \hat{\theta}_i^T Q^{-1} \hat{\theta}_i - \frac{1}{v} \hat{\theta}_i^T (\tau - S)Q^{-1} \hat{\theta}_i (\tau - S) \, dt
$$

(32)

where we compute

$$
\frac{1}{v} \hat{\theta}_i^T (\tau - S) Q^{-1} \hat{\theta}_i (\tau - S) - \frac{1}{v} \hat{\theta}_i^T (\tau) Q^{-1} \hat{\theta}_i (\tau - S)
$$

$$
= (\frac{1}{v} - \frac{1}{v} \hat{\theta}_i^T (\tau - S) Q^{-1} \hat{\theta}_i (\tau - S)) + (\frac{1}{v} \hat{\theta}_i^T (\tau) Q^{-1} \hat{\theta}_i (\tau - S) - \frac{1}{v} \hat{\theta}_i^T (\tau - S) Q^{-1} \hat{\theta}_i (\tau - S))
$$

$$
= (\frac{1}{v} - \frac{1}{v} \hat{\theta}_i^T (\tau) Q^{-1} \hat{\theta}_i (\tau - S)) + (\frac{1}{v} \hat{\theta}_i^T (\tau) Q^{-1} \hat{\theta}_i (\tau - S) - \frac{1}{v} \hat{\theta}_i^T (\tau - S) Q^{-1} \hat{\theta}_i (\tau - S))
$$

$$
= -2 \frac{1}{v} \hat{\theta}_i^T (\tau) e_v,i (\tau - S)
$$

$$
\leq -2 \frac{1}{v} \hat{\theta}_i^T (\tau) e_v,i (\tau - S)
$$

(33)

Thus, we have

$$
\Delta J_e \leq - \int^s_{s-S} \frac{1}{v} e_v,i^T \hat{\theta}_i (\tau) \, d\tau
$$

(34)

From (31), we can get the variation of $J_e$ as follows

$$
\Delta J_e = \int^s_{s-S} e_v,i^T [\frac{1}{v} \hat{\theta}_i (\tau) - \frac{1}{v} Ke_v,i (\tau)] \, d\tau
$$

By considering Eqs. (34) and (35), we have

$$
\Delta J = \Delta J_e + \Delta J_e \leq - \int^s_{s-S} \frac{1}{v} e_v,i^T (\tau) Ke_v,i (\tau) \, d\tau
$$

(36)

Because $K$ is positive-definite, we can get $\Delta J \leq 0$.

By Ineq. (36), we have shown that the function $J$ does not increase when the iteration number increases for $s \in [0, S]$. Then, we will have the boundedness of $J$ if we show that $J$ in the first iteration is bounded, i.e., $J_{i=1} < \infty$.

Considering the spatial derivative of $J_{i=1}$ as below

$$
\nabla J_{i=1} = \nabla J_c + \nabla J_e
$$

(37)
According to Ineq. (36), we have
\[ \nabla J_{i=1} \leq - \frac{1}{v} e_{v,i} K e_{v,i} \leq 0 \] (38)

Therefore, by integrating \( \nabla J_{i=1} \) from 0 to \( s \), we obtain
\[ J_{i=1} - J_{i=1}(0) \leq 0 \] (39)

Since \( e_{v,1}(0) \) and \( M \) are bounded, \( J_e(0) \) is bounded according to its definition. Since the displacement \( S \) and true values of parameters \( \theta \) are bounded and \( \tilde{\theta} \) is initialized as zero before the first iteration, \( J_e(0) \) is bounded. Therefore, \( J_{i=1}(0) \) is bounded, and thus \( J_{i=1} \) is bounded.

Finally, by Ineq. (36), we have
\[ \Delta J \leq - \int_{s-S}^{s} \frac{1}{v} e_{v,i} K e_{v,i} d\tau \] (40)

From the above inequality, we obtain
\[ J - J_{i=1} \leq - \sum_{j=1}^{i-1} \int_{s-S}^{s} \frac{1}{v} e_{v,i} K e_{v,i} d\tau \] (41)

which leads to
\[ J_{i=1} \geq - \sum_{j=1}^{i-1} \int_{s-S}^{s} \frac{1}{v} e_{v,i} K e_{v,i} d\tau \] (42)

Since \( J_{i=1} \) is bounded, we can conclude that \( \| e_{v,i} \| \to 0 \) when the iteration number \( i \to \infty \).

IV. EXPERIMENTS

A. Experimental setup

The proposed method of contour control is tested on our NC machine shown in Fig. 6, which uses a Googol numerical control system with actuators of Sanyo. The rotary encoders serve as position sensors with a resolution of 32768p/\( \tau \), and the pitch of motor is 5mm. The controller board executes algorithms at a sampling frequency of 500Hz. Note that a different sampling time or resolution will lead to different data points, and thus will affect the error, but the proposed method achieves better performance than other methods, regardless of the sampling time. In the experiment, the NC machine receives the G code based on G01, and it collects data at the same sampling time. The displacement \( s \) is the distance traveled so far, i.e. increment of displacement is related to the sampling time and is the distance between two adjacent points.

The following two performance indexes are employed to evaluate the quality of the contour error control strategy [31]:

1) \( e_{rms}(s) = \sqrt{\frac{1}{s} \int_{0}^{s} |e(s)|^2 ds} \) is the root-mean-square (RMS) value of the contour error \( e(s) \) defined in the proposed method and \( S \) represents the total length of a trajectory.

2) \( e_{max}(s) = \max|e(s)| \) is the maximum absolute (MAX) value of the contour error over the whole length of the trajectory.

B. Experimental results

The experiments are conducted to verify the CEE and CEC methods based on modification of the reference trajectory in the spatial domain. For comparison, the method of modifying the reference trajectory based on the tracking error in the time domain is also tested. In addition, how the reference trajectory affects the contour error is investigated, and the influence of learning rate on experimental results is analyzed. In the experiment, we give limits on maximum velocity and maximum acceleration in G code, with the maximum feed rate 1000mm/min and the maximum feed acceleration 1000mm/min².

Fig. 6: Multi-axis CNC machine

Fig. 7: A fourth-order ellipse trajectory

1) Modification of reference trajectory based on tracking error

Before verifying the method proposed in this paper, first we use the tracking error, i.e. the error in time domain, to modify the reference trajectory and study its performance. A fourth-order ellipse is used as the reference trajectory (see Fig. 7), which is defined as

\[
\begin{align*}
    x(t) &= \frac{P_{cost}}{4-k\cos(4t)} \\
    y(t) &= \frac{P_{cost}}{4-k\cos(4t)} \\
    P &= A(1-k)
\end{align*}
\] (43)

where \( A = 65 \) and \( k = 0.3 \), and the sampling interval of parameter \( t \) is 5ms. Likewise, a different interval will lead to different data points, and thus will affect the error, but the better performance is achieved by the proposed method, regardless of the interval. The trajectory is discretized with a sampling time to obtain a digital curve and generate the G code of the original trajectory based on G01.
By computing the tracking error $e(t) = x(t) - x_d(t)$, we modify the reference trajectory with a learning rate of $Q = 0.8$. The tracking error is given in Fig. 8 (a) and the contour error is shown in Fig. 8 (b). From these figures, it is found that the tracking error varies irregularly in different regions, and the contour error significantly increases from $6\, \mu m$ to $170\, \mu m$. Therefore, we can draw a conclusion that the method based on the tracking error is not helpful in improving the contour accuracy when the tool speed changes in each iteration.

By further obtaining the G code,

\begin{equation}
\begin{aligned}
    x(t) &= a + b \cos(ct) \\
    y(t) &= r(t) \sin(t)
\end{aligned}
\end{equation}

where $a = 25$, $b = 15$ and $c = 5$, and the sampling interval of parameter $t$ is also $5\, \text{ms}$. This trajectory is shown in Fig. 11 (a), with the same method to obtain the digital curve and further obtain the G code.

Like in the previous experiment, we implement the proposed method of contour control and obtain the changes of contour error throughout the iteration process in Fig. 11 (b). From this figure, we can find that the contour error in low-curvature areas is about $5\, \mu m$ and in high-curvature areas is about $25\, \mu m$ in the first iteration. This is not surprising as due to the characteristics of CNC machine tools and speed planning, the contour error for high-curvature trajectories is usually larger than the low-curvature ones. Then, with the progress...
of iterations, it can be found that the contour error decreases gradually and significantly.

The RMS and MAX values of contour error are shown in Fig. 12. From this figure, we can find that $e_{rms}(s)$ decreases form 3.8222$\mu$m to 0.290$\mu$m after iterations, which is very close to that of the fourth-order ellipse in Fig. 10, indicating that the large contour errors at high-curvature areas have little effect on $e_{rms}(s)$ because those large contour errors are local. Conversely, they have a significant effect on $e_{max}(s)$, which is larger than that of the ellipse in Fig. 10. Despite this, $e_{max}(s)$ decreases from 25$\mu$m to 5$\mu$m when the iteration number increases to 8. Compared to the results in the first iteration, $e_{rms}(s)$ is reduced by 93.8% and $e_{max}(s)$ reduced by 80%. These results show that the proposed method is also effective for large-curvature trajectories.

Fig. 13: Trajectory modification of the fourth-order ellipse

4) Modifications of reference trajectory: Next, we analyze what happens to the reference trajectory when it is modified to improve the contour tracking accuracy. For the fourth-order ellipse, with the learning rate 0.8, the trajectory modification is shown in Fig. 13, where $x_d$, $x_r$ and $x$ respectively represent the desired trajectory, reference trajectory and actual trajectory. Figs. 14 (a) and (b) present the adjustments of the fourth-order ellipse for the first and last iterations, respectively. From these figures, we can find that as the actual trajectory is on the right hand side of the desired trajectory, the reference trajectory compensates for their difference by setting itself on the left hand side of the desired one so the actual trajectory gets closer to the desired trajectory. A similar result can be found for the flower curve in Fig. 15, with Figs. 16 (a) and (b) presenting the adjustments of the flower curve for the first and last iterations, respectively. A different result is that after several iterations, the reference trajectory becomes less smooth, although the contour error does not diverge. This is more obvious on a trajectory with a large curvature. Since the contour error can converge to a small value in the first few iterations, when the error meets the requirements we can terminate the modification of the reference trajectory which has been found to be able to effectively solve this problem.

5) Effects of learning rate: A constant learning rate of $Q = 0.8$ was used in previous experiments and ensured good results. In this subsection, the experimental results are studied by increasing and decreasing the learning rate.

The results for $Q = 0.3$ are presented in Fig. 17, where we find that $e_{rms}(s)$ and $e_{max}(s)$ decrease more slowly compared with that of $Q = 0.8$. Nevertheless, the converging values are similar, with $e_{rms}$ decreased from 3.7387$\mu$m to 0.1846$\mu$m and $e_{max}$ from 6.002$\mu$m to 0.6410$\mu$m, respectively.

The results for $Q = 1.5$ are presented in Fig. 18. Surprisingly, the decrease of $e_{rms}$ and $e_{max}$ is also slower than that of $Q = 0.8$. This may be due to overshoot of the contour error during the modification of the reference trajectory. $e_{rms}(s)$ decreases from 3.7387$\mu$m to 0.2686$\mu$m and $e_{max}(s)$ from 6.002$\mu$m to 0.9820$\mu$m, which suggest a converging performance similar to that of $Q = 0.8$.

Finally, when we increase the learning rate to 2.2, we can find that $e_{rms}(s)$ and $e_{max}(s)$ diverge as shown in Fig. 19. The reason is that when the learning rate is too large, the contour error overshoots to the opposite direction to the desired one, and its absolute value exceeds the original contour error. In the subsequent iterations, the same process continues so it diverges. Therefore, a good trade-off of fast convergence and
robustness needs to be considered when setting the learning rate.

Fig. 19: RMS and MAX errors of fourth-order ellipse for $Q = 2.2$

V. Conclusion

This paper presented a method to improve contour tracking accuracy in numerical control machining. For digital curves, we used Hermite-spline to obtain a smooth free curve and developed a contour error estimation method. The reference trajectory was modified based on the estimated contour error in spatial domain. The stability and convergence of the proposed method were proved in theory. These results were supported with experimental results on a two-axis CNC system.

From the experimental results, we pointed out that using tracking error in time domain to modify the reference trajectory could not reduce the contour error, while using the estimated contour error in spatial domain achieved significant reduction of contour error for both smooth and high-curvature trajectories. The influence of reference trajectory change and the effects of learning rate on contour tracking were investigated.

This method of modifying the reference trajectory does not interfere with the low-level controller, so it is applicable to many commercial CNC systems and is relatively easy to implement. Our future works will focus on dealing with high-curvature trajectories using the same framework but possibly with varying learning rates.

The proposed method can be used for a system of up to 3 axes in Cartesian space but not readily for a 5-axis system. While extension to 5-axis systems is in our plan, we notice that there are still many applications requiring 2-axis and 3-axis systems, e.g. laser cutting and positioning tables. Moreover, the proposed iterative learning is useful for mass production, e.g. mobile phone screens and cases, anti-lock brakes for cars, alloy wheels, etc. Finally, the position sensors used in this work are encoders, which do not consider the effect of the mechanical system. For a more accurate measurement, optical gratings may be used.

REFERENCES
