

Solution of inverse dynamics problems for contour error minimization in CNC machines

Charlie A. Ernesto and Rida T. Farouki
Department of Mechanical and Aeronautical Engineering,
University of California, Davis, CA 95616, USA.

Abstract

For CNC machines governed by typical feedback controllers, the problem of compensating for inertia and damping of the machine axes is solved by *a priori* modifications to the commanded path geometry. Standard second-order models of axis dynamics are expressed in terms of the path parameter ξ rather than the time t as independent variable, incurring ordinary differential equations with polynomial coefficients. For a commanded path specified as a Pythagorean-hodograph curve $\mathbf{R}(\xi)$ and a P controller, a modified path $\hat{\mathbf{R}}(\xi)$ can be determined as a rational Bézier curve, that precisely compensates for the axis inertia and damping and thus (theoretically) achieves zero contour error. For PI, PID, or P-PI controllers, exact closed-form solutions for $\hat{\mathbf{R}}(\xi)$ are no longer possible, but polynomial approximations may be computed in the numerically-stable Bernstein basis on $\xi \in [0, 1]$. The inverse-dynamics path modification procedure is applicable to both constant feedrates and variable feedrates defined by polynomial functions $V(\xi)$ of the curve parameter. The method is described in the general context of PID controllers, and its implementation is then demonstrated for both P and PI controllers, governing motion along paths with extreme variations of curvature and/or parametric speed.

Keywords: CNC machine, PID controller, inverse dynamics, feedrate, contour error, path modification, Pythagorean-hodograph curve.

e-mail addresses: caernesto@ucdavis.edu, farouki@ucdavis.edu

1 Introduction

CNC machines employ feedback control systems to independently drive each machine axis in order to achieve a given speed of the tool along a given path, relative to the workpiece. Due to the inherent machine/controller dynamics, it is impossible to respond instantaneously to variations in commanded path geometry and speed. Consequently, the actual machine motion deviates from the desired motion in both path geometry (*contour error*) and speed along it (*feedrate error*). Contour error incurs appreciable inaccuracy of the machined part shape, but feedrate error has the less-serious consequence of altering the overall machining time. The focus of this study is on the “inverse dynamics” problem of identifying *a priori* modifications to the commanded path, for a given feedrate, that will cause the physical trajectory generated by the CNC machine to conform more closely to the original commanded path.

For brevity, we focus here on planar paths (the extension to spatial paths is elementary) and we consider only the intrinsic machine dynamics, without reference to cutting forces, external disturbances, etc.¹ (one can, in principle, also compensate for such effects if quantitative models of them are available). A standard second-order model [26] for the axis dynamics is used, together with a P, PI, PID, or P-PI controller. For a commanded path specified as a parametric curve $\mathbf{R}(\xi)$, the independent variable in the differential equations governing the machine/controller dynamics is transformed from the time t to the curve parameter ξ in accordance with a given (possibly constant) feedrate function $V(\xi)$. By reverting these differential equations, a modified path $\hat{\mathbf{R}}(\xi)$ is sought, such that the actual executed path — under the influence of the machine/controller dynamics — agrees with the desired path $\mathbf{R}(\xi)$.

When expressed in terms of the curve parameter ξ , the machine/controller dynamical equations have non-constant coefficients. It is advantageous to use *Pythagorean-hodograph (PH) curves* [7] to specify the path $\mathbf{R}(\xi)$, because the coefficients of these differential equations are then *polynomials* in ξ . For a P controller, the modified path $\hat{\mathbf{R}}(\xi)$ can be exactly determined as a higher-order rational curve. For more sophisticated controllers, exact descriptions of $\hat{\mathbf{R}}(\xi)$ are no longer possible, but algorithms can be formulated to compute a convergent sequence of polynomial approximations to it.

The focus of this paper is the inverse dynamics path-modification problem

¹The intrinsic machine dynamics may dominate in *high-speed machining* [4, 19, 21, 22] of complicated shapes, involving very large speeds and acceleration/deceleration rates.

for smooth analytic curves. A number of authors have recently investigated path-modification procedures for CNC machines, primarily to reduce contour error in the vicinity of sharp path corners through an “over-corner” approach [5, 17, 18]. These methods invoke an *ad hoc* replacement of sharp corners by smooth curve segments, empirically optimized to reduce contour error for a given corner angle, rather than solving an inverse dynamics problem.

The plan for the remainder of this paper is as follows. Section 2 describes the basic machine/controller dynamic models employed herein, expressed in the time domain. These differential equations are transformed in Section 3 so that the time t is replaced by the curve parameter ξ as independent variable, and the inverse dynamics path-modification problem is formulated through a reversion of the resulting equations (which have non-constant coefficients). Section 4 presents an exact solution for the modified path in the case of a P controller, and an approximate solution (allowing refinement to any desired accuracy) in the case of a PI controller. Computed examples are presented to illustrate the ability of the *a priori* path modification procedure to minimize contour error along paths with strong curvature variations. Finally, Section 5 summarizes and assesses the results of the present study, and identifies other promising directions for further investigation.

2 Machine/controller dynamics

Let $(X(t), Y(t))$ denote the commanded path and $(x(t), y(t))$ the actual path executed by a CNC machine, both parameterized by time t . Standard models [1, 5] for the dynamics of CNC machines involve a determination of $(x(t), y(t))$ from $(X(t), Y(t))$ through differential equations of the form

$$\begin{aligned} a_x \ddot{x} + b_x \dot{x} + c_x x &= d_x \ddot{X} + e_x \dot{X} + X, \\ a_y \ddot{y} + b_y \dot{y} + c_y y &= d_y \ddot{Y} + e_y \dot{Y} + Y, \end{aligned} \tag{1}$$

where dots indicate time derivatives, and the constant coefficients a_x, b_x, \dots depend on the machine/controller physical parameters. Figures 1 and 2 show block diagrams of typical physical systems giving rise to the equations (1).

For brevity, we discuss only the x -axis dynamics: similar principles apply to the y -axis, with possibly different values for the physical parameters. The system variables (and their dimensions) are as follows — u (V) is the control variable; k_a (A/V) and k_t (N m/A) are the current amplifier and motor torque gains; J (kg m²) and B (kg m²/s) are the x -axis inertia and viscous damping;

T (Nm), ω (rad/s), and θ (rad) are the motor torque, angular speed, and position; and r_g (m/rad) is the transmission ratio — i.e., the translation of the axis for unit rotation of the motor shaft.

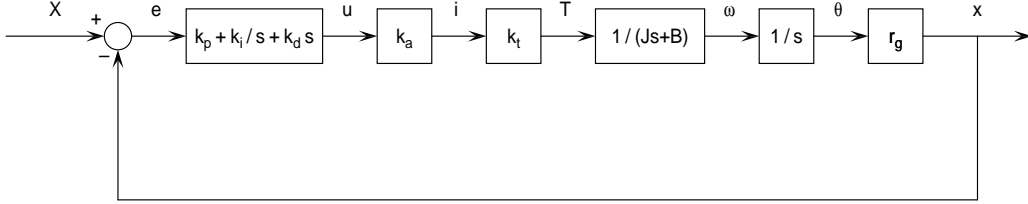


Figure 1: Block diagram for the x -axis drive — the proportional, integral, and derivative gains of the PID controller are k_p , k_i , and k_d , while $e = X - x$ is the difference between the commanded and actual axis locations. The power amplifier converts the actuating signal u into a current i to the motor, which produces a torque T that determines the angular speed ω through the system inertia J and damping B . The motor shaft angle θ , obtained by integration of ω , determines the axis linear position x through the transmission ratio r_g .

We consider here several commonly-used forms for the controller transfer function: PID control with proportional, integral, derivative gains k_p , k_i , k_d as shown in Figure 1 (with P and PI control as the special cases $k_i = k_d = 0$ and $k_d = 0$) and P-PI control as employed in [17, 18]. The latter, illustrated in Figure 2, employs feedback of the motor angular speed ω as well as the axis position x . For brevity we set $K = k_a k_t r_g$ henceforth, since the parameters k_a , k_t , r_g often occur in the form of this product. For each model, the system transfer function and the coefficients in the governing differential equations (1) will be derived below.

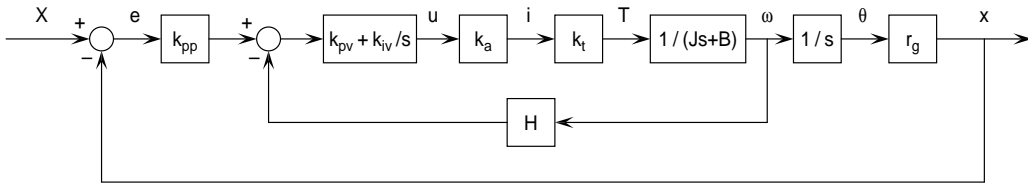


Figure 2: Block diagram for x -axis drive with P-PI controller. The position loop is closed by the P controller, with proportional gain k_{pp} , and the velocity loop is closed by the PI controller, with proportional gain k_{pv} and integral gain k_{iv} . The remainder of the block diagram is identical to that in Figure 1.

2.1 PID controller

For the general PID controller in Figure 1, the transfer function relating the Laplace transforms of the output x and the input X can be written as

$$\frac{x}{X} = \frac{K(k_d s^2 + k_p s + k_i)}{J s^3 + (B + K k_d) s^2 + K k_p s + K k_i}.$$

This defines a third-order system, with three poles and two (real or complex conjugate) zeros. The coefficients of the differential equations (1) are

$$a_x = \frac{J}{K k_i}, \quad b_x = \frac{B + K k_d}{K k_i}, \quad c_x = \frac{k_p}{k_i}, \quad d_x = \frac{k_d}{k_i}, \quad e_x = \frac{k_p}{k_i}.$$

Analogous results hold for the y -axis dynamics, using appropriate values for the physical parameters associated with that axis.

2.2 P controller

As a special case of the PID controller shown in Figure 1, consider the case of a simple P controller, specified by the choices $k_i = k_d = 0$. The transfer function then reduces to

$$\frac{x}{X} = \frac{K k_p}{J s^2 + B s + K k_p}.$$

This describes a second-order system, with two (real or complex conjugate) poles and no zeros. The differential equations (1) then have the coefficients

$$a_x = 0, \quad b_x = \frac{J}{K k_p}, \quad c_x = \frac{B}{K k_p}, \quad d_x = 0, \quad e_x = 0,$$

with analogous results for the y -axis dynamics.

2.3 PI controller

Since a P controller cannot ensure zero steady-state error, it is often desirable to upgrade to a PI controller (specified by $k_d = 0$ in the PID controller). In this case, the transfer function becomes

$$\frac{x}{X} = \frac{K(k_p s + k_i)}{J s^3 + B s^2 + K k_p s + K k_i}.$$

This defines a third-order system, with three poles and one (real) zero. The differential equations (1) then have the coefficients

$$a_x = \frac{J}{Kk_i}, \quad b_x = \frac{B}{Kk_i}, \quad c_x = \frac{k_p}{k_i}, \quad d_x = 0, \quad e_x = \frac{k_p}{k_i},$$

and analogous results hold for the y -axis dynamics.

2.4 P–PI controller

The P–PI controller, shown in Figure 2, involves feedback of the position x through a P controller, and the angular velocity ω through a PI controller, yielding the transfer function

$$\frac{x}{X} = \frac{Kk_{pp}(k_{pv}s + k_{iv})}{Js^3 + (B + Hk_a k_t k_{pv})s^2 + k_a k_t (Hk_{iv} + k_{pp}k_{pv}r_g)s + Kk_{pp}k_{iv}}.$$

This describes a third-order system with three poles and one real zero — the transfer function has the same form as the PI controller, but the coefficients (and the physical parameters that determine them) differ. In the differential equations (1), the coefficients are

$$a_x = \frac{J}{Kk_{pp}k_{iv}}, \quad b_x = \frac{B + Hk_a k_t k_{pv}}{Kk_{pp}k_{iv}}, \quad c_x = \frac{k_{pv}}{k_{iv}} + \frac{H}{k_{pp}r_g},$$

$$d_x = 0, \quad e_x = \frac{k_{pv}}{k_{iv}},$$

with analogous results for the y -axis dynamics.

3 Modelling in curve parameter domain

The time domain is conventionally used to study machine dynamics, but in practice the desired path $\mathbf{R}(\xi) = (X(\xi), Y(\xi))$ — generated by a CAD/CAM system — is not parameterized by time, but rather by polynomial or rational functions of a general parameter ξ . The *parametric speed*

$$\sigma(\xi) = |\mathbf{R}'(\xi)| = \sqrt{X'^2(\xi) + Y'^2(\xi)} = \frac{ds}{d\xi}$$

of $\mathbf{R}(\xi)$ is the function specifying the rate of change of its arc length s with respect to the parameter ξ . Ideally, one would like to have $\sigma \equiv 1$ (i.e., $s \equiv \xi$), but the only curves that admit parameterization by rational functions of their arc length are straight lines [12, 13]. For the Pythagorean–hodograph (PH) curves, $\sigma(\xi)$ is a *polynomial* function of the curve parameter [7].

For general parametric curves, the (constant or variable) speed or *feedrate* $V = ds/dt$ must be specified along the path to fix the relation between the curve parameter ξ and elapsed time t . If the curve is *regular* — i.e., it satisfies $\sigma(\xi) \neq 0$ for all ξ — and the feedrate V is everywhere positive, there will be a one-to-one relation between the variables ξ , s , t describing progress along the path, but it is the curve parameter ξ that the real-time interpolator algorithm employs to compute path reference points $(X_k, Y_k) = (X(\xi_k), Y(\xi_k))$ at each servo sampling instant $k\Delta t$, $k = 0, 1, 2, \dots$ [14, 15, 20, 24, 25].

3.1 Transformation of equations

To formulate the inverse dynamics problem, the equations (1) are re-cast in terms of the parameter ξ , rather than time t , as the independent variable. This is accomplished by invoking the chain rule to observe that derivatives with respect t and ξ are related by

$$\frac{d}{dt} = \frac{ds}{dt} \frac{d\xi}{ds} \frac{d}{d\xi} = \frac{V}{\sigma} \frac{d}{d\xi}. \quad (2)$$

Suppose that, for any given path and feedrate, the function $\xi(t)$ specifies the variation of the curve parameter with time. Applying (2) repeatedly, we can express the derivatives of $\xi(t)$ recursively² as

$$\begin{aligned} \dot{\xi} &= \frac{V}{\sigma}, & \ddot{\xi} &= \frac{\sigma V' - \sigma' V}{\sigma^2} \dot{\xi}, \\ \ddot{\xi} &= \frac{\sigma V' - 3\sigma' V}{\sigma^2} \ddot{\xi} + \frac{\sigma V'' - \sigma'' V}{\sigma^2} \dot{\xi}^2, & \text{etc.}, \end{aligned} \quad (3)$$

where primes indicate derivatives with respect to ξ . The parametric speed σ and its derivatives, required in (3), can also be expressed recursively as

$$\begin{aligned} \sigma &= |\mathbf{R}'|, & \sigma' &= \frac{\mathbf{R}' \cdot \mathbf{R}''}{\sigma}, \\ \sigma'' &= \frac{\mathbf{R}' \cdot \mathbf{R}''' + |\mathbf{R}''|^2 - \sigma'^2}{\sigma}, & \text{etc.} \end{aligned} \quad (4)$$

²These relations were first developed in [15], in the context of formulating real-time CNC interpolators for variable feedrate functions.

In (3), the feedrate is assumed to be specified as a function $V(\xi)$ of the curve parameter (if $V = \text{constant}$, we have $V' = V'' = \dots = 0$). Since $\sigma, \sigma', \sigma'', \dots$ are known functions of ξ , the expressions (3) are all known functions of ξ .

Writing V/σ in (2) as $\dot{\xi}$ and applying it repeatedly, time derivatives can be converted to ξ derivatives through

$$\begin{aligned}\frac{d}{dt} &= \dot{\xi} \frac{d}{d\xi}, & \frac{d^2}{dt^2} &= \dot{\xi}^2 \frac{d^2}{d\xi^2} + \ddot{\xi} \frac{d}{d\xi}, \\ \frac{d^3}{dt^3} &= \dot{\xi}^3 \frac{d^3}{d\xi^3} + 3\dot{\xi}\ddot{\xi} \frac{d^2}{d\xi^2} + \ddot{\xi}\ddot{\xi} \frac{d}{d\xi}, & \text{etc.}\end{aligned}$$

Transforming the differential equations (1) in this manner, we obtain

$$\begin{aligned}a_x \dot{\xi}^3 x''' + (3a_x \ddot{\xi} + b_x \dot{\xi}) \dot{\xi} x'' + (a_x \ddot{\xi} + b_x \ddot{\xi} + c_x \dot{\xi}) x' + x \\ = d_x \dot{\xi}^2 X'' + (d_x \ddot{\xi} + e_x \dot{\xi}) X' + X, \\ a_y \dot{\xi}^3 y''' + (3a_y \ddot{\xi} + b_y \dot{\xi}) \dot{\xi} y'' + (a_y \ddot{\xi} + b_y \ddot{\xi} + c_y \dot{\xi}) y' + y \\ = d_y \dot{\xi}^2 Y'' + (d_y \ddot{\xi} + e_y \dot{\xi}) Y' + Y,\end{aligned}$$

where again primes indicate derivatives with respect to ξ . Substituting from (3) and multiplying through by σ^5 , these equations can be written as

$$\begin{aligned}\alpha_x x''' + \beta_x x'' + \gamma_x x' + \delta_x x &= \lambda_x X'' + \mu_x X' + \nu_x X, \\ \alpha_y y''' + \beta_y y'' + \gamma_y y' + \delta_y y &= \lambda_y Y'' + \mu_y Y' + \nu_y Y,\end{aligned}\tag{5}$$

where α_x, β_x, \dots are functions of ξ , defined by

$$\begin{aligned}\alpha_x &= a_x \sigma^2 V^3, & \alpha_y &= a_y \sigma^2 V^3, \\ \beta_x &= \sigma V^2 [3a_x (\sigma V' - \sigma' V) + b_x \sigma^2], \\ \beta_y &= \sigma V^2 [3a_y (\sigma V' - \sigma' V) + b_y \sigma^2], \\ \gamma_x &= V [(a_x (\sigma V' - 3\sigma' V) + b_x \sigma^2) (\sigma V' - \sigma' V) \\ &\quad + a_x \sigma V (\sigma V'' - \sigma'' V) + c_x \sigma^4], \\ \gamma_y &= V [(a_y (\sigma V' - 3\sigma' V) + b_y \sigma^2) (\sigma V' - \sigma' V) \\ &\quad + a_y \sigma V (\sigma V'' - \sigma'' V) + c_y \sigma^4], \\ \delta_x &= \delta_y = \nu_x = \nu_y = \sigma^5, \\ \lambda_x &= d_x \sigma^3 V^2, & \lambda_y &= d_y \sigma^3 V^2, \\ \mu_x &= \sigma^2 V [d_x (\sigma V' - \sigma' V) + e_x \sigma^2], \\ \mu_y &= \sigma^2 V [d_y (\sigma V' - \sigma' V) + e_y \sigma^2].\end{aligned}\tag{6}$$

If the commanded path $\mathbf{R}(\xi) = (X(\xi), Y(\xi))$ is a PH curve, so that $\sigma(\xi)$ is a polynomial, and the feedrate is a specified polynomial function $V(\xi)$ of the curve parameter, the executed path $\mathbf{r}(\xi) = (x(\xi), y(\xi))$ is the solution of the differential equations (5), with the polynomials (6) in ξ as coefficients.

3.2 Inverse dynamics problem

Instead of computing the actual path from the commanded path, however, we wish to use equations (5) to solve an “inverse” problem — we seek a *modified* commanded path $(\hat{X}(\xi), \hat{Y}(\xi))$ that, subject to the machine dynamics, yields an executed path exactly coincident with the original commanded path, i.e., $(x(\xi), y(\xi)) \equiv (X(\xi), Y(\xi))$. Substituting $(\hat{X}(\xi), \hat{Y}(\xi))$ for $(X(\xi), Y(\xi))$ and $(X(\xi), Y(\xi))$ for $(x(\xi), y(\xi))$ in (5), we obtain

$$\begin{aligned}\lambda_x \hat{X}'' + \mu_x \hat{X}' + \nu_x \hat{X} &= \alpha_x X''' + \beta_x X'' + \gamma_x X' + \delta_x X, \\ \lambda_y \hat{Y}''' + \mu_y \hat{Y}' + \nu_y \hat{Y} &= \alpha_y Y''' + \beta_y Y'' + \gamma_y Y' + \delta_y Y.\end{aligned}\tag{7}$$

These are linear differential equations for $\hat{X}(\xi), \hat{Y}(\xi)$ in which the coefficients and right-hand sides are known *polynomial* functions of ξ . We are interested in finding solutions to these equations over the parameter domain $\xi \in [0, 1]$.

In the case of a P controller, we have $d_x = e_x = d_y = e_y = 0$ and hence $\lambda_x(\xi) = \mu_x(\xi) = \lambda_y(\xi) = \mu_y(\xi) \equiv 0$, so $\hat{X}(\xi), \hat{Y}(\xi)$ can be determined exactly from (7) as *rational functions* (see Section 4.1). For PI control, $d_x = d_y = 0$ so $\lambda_x(\xi) = \lambda_y(\xi) \equiv 0$ and (7) are first-order differential equations for $\hat{X}(\xi), \hat{Y}(\xi)$ requiring an initial condition for a unique solution. Finally, for PID control, all the left-hand side coefficients in (7) are non-vanishing, and hence they are second-order equations, requiring two initial conditions for a unique solution. Note that, since $\sigma(\xi)$ and $V(\xi)$ are positive for all ξ , the differential equations (1) have no *singular points* — i.e., there are no real values of ξ at which the coefficient of the highest-order term vanishes [2].

In general, linear differential equations with polynomial coefficients do not admit “simple” (polynomial or rational) solutions: see the Appendix. As an alternative, one may seek infinite power series solutions — but the truncation error and radius of convergence is often difficult to assess. In the case of PI or PID control, we therefore seek *approximate* polynomial solutions to the equations, represented in the numerically-stable Bernstein basis [8, 10, 11].

If the path $\mathbf{R}(\xi) = (X(\xi), Y(\xi))$ is a PH curve of (odd) degree n , and the feedrate function $V(\xi)$ is a polynomial of degree m , we have $\deg(\sigma) = n - 1$

and (assuming for simplicity that $m < n$):

$$\begin{aligned}\deg(\alpha_x, \alpha_y) &= 3m + 2n - 2, \\ \deg(\beta_x, \beta_y) &= \deg(\lambda_x, \lambda_y) = 2m + 3n - 3, \\ \deg(\gamma_x, \gamma_y) &= \deg(\mu_x, \mu_y) = m + 4n - 4, \\ \deg(\delta_x, \delta_y) &= \deg(\nu_x, \nu_y) = 5n - 5.\end{aligned}$$

All the coefficients (6) are then of degree $\leq 5n - 5$, and the right-hand sides in (7) are known polynomials of degree $6n - 5$ in ξ , that can be expressed [23] in the numerically-stable Bernstein basis on $\xi \in [0, 1]$.

3.3 Interpretation of modified path

Before proceeding to a detailed description of the path modification and the “induced feedrate” associated with it, one should have a clear picture of their significance and their role in compensating for the machine dynamics.

For any given commanded path $\mathbf{R}(\xi) = (X(\xi), Y(\xi))$ and feedrate $V(\xi)$, let $\hat{\mathbf{R}}(\xi) = (\hat{X}(\xi), \hat{Y}(\xi))$ be the modified path, obtained by solving equations (7). For each $\xi \in [0, 1]$ the points $\mathbf{R}(\xi)$ and $\hat{\mathbf{R}}(\xi)$ of the original and modified paths correspond to the same instant in time, namely

$$t(\xi) = \int_0^\xi \frac{\sigma}{V} d\xi. \quad (8)$$

Note, in particular, that the total traversal time $T = t(1)$ is the same for the modified path $\hat{\mathbf{R}}(\xi)$ as for the original path $\mathbf{R}(\xi)$.

When using the modified path, the real-time interpolator should compute reference-point parameter values $\xi_1, \xi_2, \xi_3, \dots$ corresponding to the sampling times $\Delta t, 2\Delta t, 3\Delta t, \dots$ based upon the original commanded path $\mathbf{R}(\xi)$, its parametric speed $\sigma(\xi)$, and the original feedrate function $V(\xi)$. The output of the real-time interpolator, however, will be the sequence of reference points $\hat{\mathbf{R}}(\xi_1), \hat{\mathbf{R}}(\xi_2), \hat{\mathbf{R}}(\xi_3), \dots$ on the *modified* path — rather than the points $\mathbf{R}(\xi_1), \mathbf{R}(\xi_2), \mathbf{R}(\xi_3), \dots$ on the *original* commanded path. In other words, the real-time interpolator algorithm is modified only in the final step, to compute the reference points from $\xi_1, \xi_2, \xi_3, \dots$ using $\hat{\mathbf{R}}(\xi)$ instead of $\mathbf{R}(\xi)$.

The parametric speed $\hat{\sigma}(\xi)$ and cumulative arc length $\hat{s}(\xi)$ of the modified path $\hat{\mathbf{R}}(\xi)$ are defined by

$$\hat{\sigma}(\xi) = |\hat{\mathbf{R}}(\xi)| = \sqrt{\hat{X}'^2(\xi) + \hat{Y}'^2(\xi)}$$

and

$$\hat{s}(\xi) = \int_0^\xi \hat{\sigma} \, d\xi.$$

Now the modified path $\hat{\mathbf{R}}(\xi)$ and time function (8) define an *induced feedrate* $\hat{V}(\xi)$ that differs from the feedrate $V(\xi)$ specified for the original path $\mathbf{R}(\xi)$. This can be seen by noting that

$$V = \sigma \frac{d\xi}{dt} \quad \text{and} \quad \hat{V} = \hat{\sigma} \frac{d\xi}{dt}, \quad (9)$$

where the function $\xi(t)$ in (9) is *unique* — it identifies corresponding points on $\mathbf{R}(\xi)$ and $\hat{\mathbf{R}}(\xi)$ at each time t . Since $\sigma(\xi)$ and $\hat{\sigma}(\xi)$ are in general different, $\hat{V}(\xi)$ and $V(\xi)$ also differ: from (9) see that $\hat{V}/V = \hat{\sigma}/\sigma$. Note, in particular, that a constant feedrate V specified for $\mathbf{R}(\xi)$ will generally incur a *variable* induced feedrate \hat{V} along the modified path $\hat{\mathbf{R}}(\xi)$.

There is no need to explicitly compute the induced feedrate: the real-time interpolator algorithm generates it “implicitly” by a non-uniform spacing of the reference points $\hat{\mathbf{R}}(\xi_1), \hat{\mathbf{R}}(\xi_2), \hat{\mathbf{R}}(\xi_3), \dots$ at sampling times $\Delta t, 2\Delta t, 3\Delta t, \dots$ on the modified path. In the examples presented below, we shall see that the modified path often exhibits tight “loops” with corresponding “spikes” in the induced feedrate. This extreme behavior in the commanded motion is necessary to overcome the inherent smoothing effects of the machine inertia and damping. In combination, the modified path $\hat{\mathbf{R}}(\xi)$ and induced feedrate $\hat{V}(\xi)$ compensate for the machine/controller dynamics, to ensure a physical output motion corresponding to accurate traversal of the original commanded path $\mathbf{R}(\xi)$ at the original commanded feedrate $V(\xi)$.

4 Path modification procedure

We now discuss implementation of the path modification procedure, for the case of P and PI controllers. The P controller permits *exact* definition of the modified path as a rational Bézier curve. For the PI controller, polynomial approximations to the modified path are computed. Since the P–PI controller is functionally equivalent to the PI controller, we shall not consider it further here. Also, since the path modification scheme for a PID controller is more complicated than for a PI controller in the implementation details, but not in basic methodology, we omit full treatment of it here.

To compare machine performance in response to the original commanded path $\mathbf{R}(\xi)$ and the modified path $\hat{\mathbf{R}}(\xi)$ as input, the machine dynamics were simulated in MATLAB using the `lsim` function. As input to `lsim`, a sequence of reference points (corresponding to time increments equal to the sampling interval Δt) was computed by a real-time interpolator algorithm, according to the prescribed path and feedrate. In the simulations, a nominal sampling frequency of $f = 1$ kHz was employed, corresponding to a sampling interval of $\Delta t = 0.001$ seconds. For cases where a constant feedrate V was employed with a P controller, the simulation was started at time $t = -T$ so the output during the interval $t \in [0, T]$ is representative of the steady-state behavior.

4.1 P controller

In the case of a P controller, we have $a_x = d_x = e_x = 0$ and $a_y = d_y = e_y = 0$, and hence the coefficients (6) reduce to

$$\begin{aligned}\alpha_x &= \alpha_y = \lambda_x = \lambda_y = \mu_x = \mu_y = 0, \\ \beta_x &= b_x \sigma^3 V^2, \quad \beta_y = b_y \sigma^3 V^2, \\ \gamma_x &= \sigma^2 V [b_x(\sigma V' - \sigma' V) + c_x \sigma^2], \\ \gamma_y &= \sigma^2 V [b_y(\sigma V' - \sigma' V) + c_y \sigma^2], \\ \delta_x &= \delta_y = \nu_x = \nu_y = \sigma^5.\end{aligned}$$

Cancelling out the common factor σ^2 , equations (7) then become

$$\begin{aligned}\sigma^3 \hat{X} &= b_x \sigma V^2 X'' + V [b_x(\sigma V' - \sigma' V) + c_x \sigma^2] X' + \sigma^3 X, \\ \sigma^3 \hat{Y} &= b_y \sigma V^2 Y'' + V [b_y(\sigma V' - \sigma' V) + c_y \sigma^2] Y' + \sigma^3 Y.\end{aligned}\quad (10)$$

This allows us to determine the modified path explicitly as

$$\begin{aligned}\hat{X} &= \frac{b_x \sigma V^2 X'' + V [b_x(\sigma V' - \sigma' V) + c_x \sigma^2] X' + \sigma^3 X}{\sigma^3}, \\ \hat{Y} &= \frac{b_y \sigma V^2 Y'' + V [b_y(\sigma V' - \sigma' V) + c_y \sigma^2] Y' + \sigma^3 Y}{\sigma^3}.\end{aligned}\quad (11)$$

If $\mathbf{R}(\xi) = (X(\xi), Y(\xi))$ is a PH curve³ of degree n , and the feedrate $V(\xi)$ is of degree $< n$, expressions (11) specify the modified path $\hat{\mathbf{R}}(\xi) = (\hat{X}(\xi), \hat{Y}(\xi))$

³Although we emphasize the PH curves here on account of their flexible and essentially exact real-time interpolator algorithms, and the desire to obtain a *rational* modified curve $\hat{\mathbf{R}}(\xi)$, it should be noted that expressions (11) apply to *any* analytic path $\mathbf{R}(\xi)$ for which we can compute the parametric speed $\sigma = |\mathbf{R}'|$ and its derivative σ' .

as a rational curve of degree $4n - 3$. When $\mathbf{R}(\xi)$ is a PH quintic, for example, $\hat{\mathbf{R}}(\xi)$ can be exactly represented as a degree 17 rational Bézier curve — whose control points and weights can be readily derived from (11).

It is more convenient to regard the modified path $\hat{\mathbf{R}}(\xi)$ as a *variable offset*

$$\hat{\mathbf{R}}(\xi) = \mathbf{R}(\xi) + \Delta\mathbf{R}(\xi) \quad (12)$$

from the original path $\mathbf{R}(\xi)$, where the components of the vector displacement function $\Delta\mathbf{R}(\xi) = (\Delta X(\xi), \Delta Y(\xi))$ are defined by

$$\begin{aligned} \Delta X &= b_x(V/\sigma)^2 X'' + (V/\sigma) [b_x(V/\sigma)' + c_x] X', \\ \Delta Y &= b_y(V/\sigma)^2 Y'' + (V/\sigma) [b_y(V/\sigma)' + c_y] Y', \end{aligned}$$

with $(V/\sigma)' = (\sigma V' - \sigma' V)/\sigma^2$. Expression (12) offers a more efficient means of evaluating the modified path than explicit representation as a high-degree rational Bézier curve. In general, the displacement $\Delta\mathbf{R}(\xi)$ will vary in both *magnitude* and *direction* along the original path $\mathbf{R}(\xi)$. Note that for identical axes — $b_x = b_y (= b, \text{ say})$ and $c_x = c_y (= c, \text{ say})$ — it can be expressed as

$$\Delta\mathbf{R}(\xi) = b \dot{\xi}^2 \mathbf{R}''(\xi) + (b \ddot{\xi} + c \dot{\xi}) \mathbf{R}'(\xi),$$

where $\dot{\xi}$, $\ddot{\xi}$ are given as functions of ξ by (3).

The simplest case is that of a constant feedrate V , i.e., $V' \equiv 0$. It should be noted that, in this case, the expression for $\Delta\mathbf{R}(\xi) = (\Delta X(\xi), \Delta Y(\xi))$ is non-vanishing at $\xi = 0$ and 1 , i.e., there is a discrepancy between the start and end points of the original path $\mathbf{R}(\xi)$ and the modified path $\hat{\mathbf{R}}(\xi)$. This is a consequence of the fact that the constant feedrate V applies to the entire curve, $-\infty < \xi < +\infty$, not just $\xi \in [0, 1]$. Finite displacements at $\xi = 0$ and 1 are needed to compensate for the machine/controller dynamics on passing through those points with the constant speed V . To obtain a modified path satisfying $\hat{\mathbf{R}}(0) = \mathbf{R}(0)$ and $\hat{\mathbf{R}}(1) = \mathbf{R}(1)$, one must use a feedrate function $V(\xi)$ with $V(0) = V(1) = 0$, i.e., the motion must start and end at rest.

Example 1 As test curve for the case of a P controller we use the PH quintic interpolant [9] to the Hermite data (with dimensions in meters):

$$\begin{aligned} \mathbf{r}(0) &= (0.0, 0.0), & \mathbf{r}'(0) &= (3.0, 2.5), \\ \mathbf{r}(1) &= (0.7, 0.1), & \mathbf{r}'(1) &= (2.5, -3.0), \end{aligned}$$

shown in Figure 3. In terms of the complex model [6] for planar PH curves, the hodograph $\mathbf{r}'(\xi) = [\mathbf{w}_0(1 - \xi)^2 + \mathbf{w}_1 2(1 - \xi)\xi + \mathbf{w}_2 \xi^2]^2$ of this curve is the square of the complex quadratic polynomial with Bernstein coefficients

$$\begin{aligned}\mathbf{w}_0 &= 1.85810721 + 0.67272760i, \\ \mathbf{w}_1 &= -1.11728797 + 0.43234437i, \\ \mathbf{w}_2 &= 1.78957046 - 0.83818997i.\end{aligned}$$

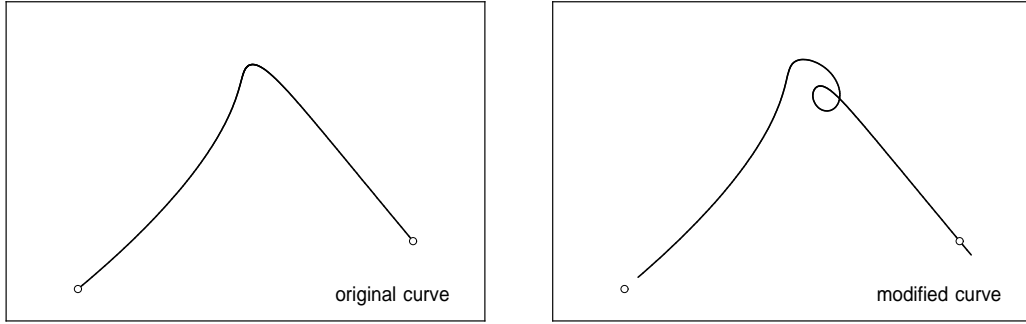


Figure 3: Left: the PH quintic test curve for Example 1. Right: the modified curve defined by (11), that compensates for the machine/controller dynamics.

Correspondingly, the parametric speed $\sigma(\xi)$ is a quartic polynomial with the Bernstein coefficients

$$\begin{aligned}\sigma_0 &= |\mathbf{w}_0|^2, & \sigma_1 &= \text{Re}(\mathbf{w}_0 \bar{\mathbf{w}}_1), \\ \sigma_2 &= \frac{1}{3} [2|\mathbf{w}_1|^2 + \text{Re}(\mathbf{w}_2 \bar{\mathbf{w}}_0)], \\ \sigma_3 &= \text{Re}(\mathbf{w}_1 \bar{\mathbf{w}}_2), & \sigma_4 &= |\mathbf{w}_2|^2,\end{aligned}$$

and the arc length $s(\xi)$ is the quintic polynomial with Bernstein coefficients

$$s_0 = 0 \quad \text{and} \quad s_k = \frac{1}{5} \sum_{j=0}^{k-1} \sigma_j \quad \text{for } k = 1, \dots, 5.$$

The total arc length of this curve is $S = s(1) = s_5 = 1.108098$ m.

Figure 4 plots the parametric speed σ and curvature κ of this test curve, as functions of the arc length $s(\xi)$ rather than the parameter ξ . Because of the uneven parameter flow along the curve, plotting σ and κ as functions of ξ

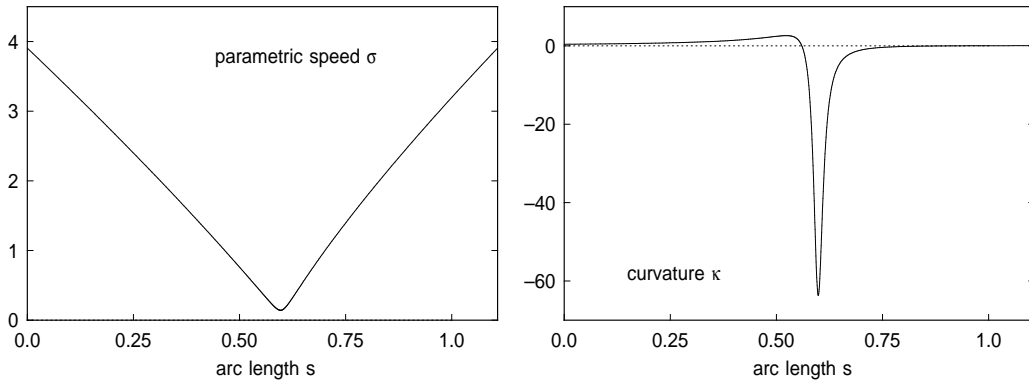


Figure 4: The parametric speed σ (left) and curvature κ (right) of the PH quintic curve in Example 1, plotted as functions of the curve arc length s .

fails to adequately portray the severity of their variation, that challenges the ability of the controller to faithfully execute the commanded path.

The same physical parameters are assumed for both the x and y axes, namely $k_a = 8$ A/V, $k_t = 0.5$ N m/A, $r_g = 0.002$ m/rad, $J = 0.01$ kg m², $B = 0.025$ kg m²/s, and P controller gain $k_p = 10$, so that $K = k_a k_t r_g = 0.008$, $b_x = b_y = J/Kk_p = 0.125$, and $c_x = c_y = B/Kk_p = 0.3125$. With these values, the closed-loop system is stable for all positive k_p (it is over-damped for $k_p < 1.953$ and under-damped for $k_p > 1.953$). Also, we choose a constant feedrate $V = 0.12$ m/s, giving a nominal traversal time $T = S/V \approx 9.23$ s.

For the above parameters, Figure 3 shows the modification of the commanded path defined by (11). A dramatic difference between the original and modified commanded paths is apparent near the region of high curvature. The peculiar rapid “looping” of the modified path is needed to compensate for the inability of the machine/controller dynamics to respond sufficiently fast to accurately track the sharply-curved region of the original path. By means of this looping behavior, the modified path “tricks” the machine/controller into faithfully executing the original commanded path.

The arc length $\hat{S} = 1.301524$ of the modified path exceeds that of the original path by $\sim 17.5\%$ and, since the traversal time must be equal (see Section 3.3) to that for the unmodified path, the average induced feedrate on the modified path must be proportionately higher. When $b_x = b_y = b$, $c_x = c_y = c$, and

$V = \text{constant}$, one can express the derivative of $\hat{\mathbf{R}}(\xi)$ as

$$\begin{aligned} \hat{\mathbf{R}}' = & \left[1 + b \frac{(3\sigma'^2 - \sigma\sigma'')V^2}{\sigma^4} - c \frac{V\sigma'}{\sigma^2} \right] \mathbf{R}' \\ & + \left[c - 3b \frac{V\sigma'}{\sigma^2} \right] \mathbf{R}'' + b \frac{V^2}{\sigma^2} \mathbf{R}''' , \end{aligned}$$

and its parametric speed is $\hat{\sigma} = |\hat{\mathbf{R}}'|$. As observed in Section 3.3, the induced feedrate along the modified path is $\hat{V} = (\hat{\sigma}/\sigma)V$ and this is shown in Figure 5. \hat{V} conforms closely to the nominal value $V = 0.12$ m/s over most of $\hat{\mathbf{R}}(\xi)$ except the high curvature region, which incurs⁴ a 9-fold “spike” in \hat{V} . This is a consequence of the fact that $\hat{\sigma}$ does not decrease as dramatically as σ , and hence the ratio $\hat{\sigma}/\sigma$ becomes quite large. Figure 5 also shows the induced feedrate for the case $V = 0.06$ m/s.

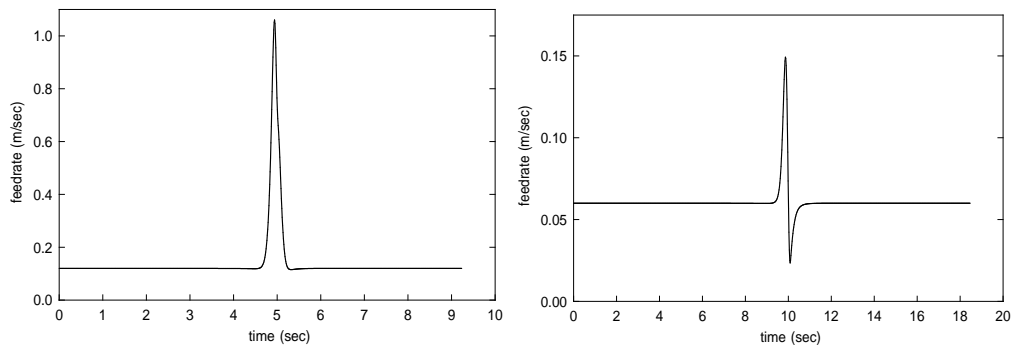


Figure 5: Left: induced feedrate \hat{V} along the modified path $\hat{\mathbf{R}}(\xi)$ in Figure 3, in the case of a constant feedrate $V = 0.12$ m/s along the original path $\mathbf{R}(\xi)$. Right: induced feedrate in the case $V = 0.06$ m/s (note the different scales).

For fixed values of all the other parameters, it is interesting to see how the modified path $\hat{\mathbf{R}}(\xi)$ depends on the constant feedrate V and controller gain k_p . The effect of decreasing and increasing V is illustrated in Figure 6, while Figure 7 shows that of decreasing and increasing k_p . For simplicity, a constant feedrate V is employed in this example. As observed above, this implies that the modified path $\hat{\mathbf{R}}(\xi)$ does not coincide exactly with the original path $\mathbf{R}(\xi)$ at $\xi = 0$ and 1 — this discrepancy is apparent in Figures 3, 6, and 7.

⁴This spike in feedrate, together with the looped path in Figure 3, comprise the severe commanded behavior necessary to compensate for the machine inertia and damping, in executing the high-curvature region of the original curve at a fixed feedrate $V = 0.12$ m/s.

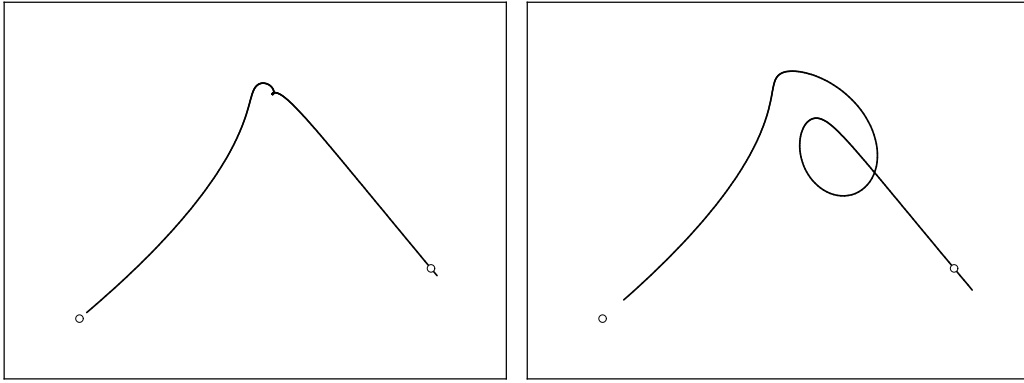


Figure 6: Modified path for constant feedrates V different from the nominal value 0.12 m/s used in Figure 3 — left: $V = 0.06$ m/s, right: $V = 0.18$ m/s.

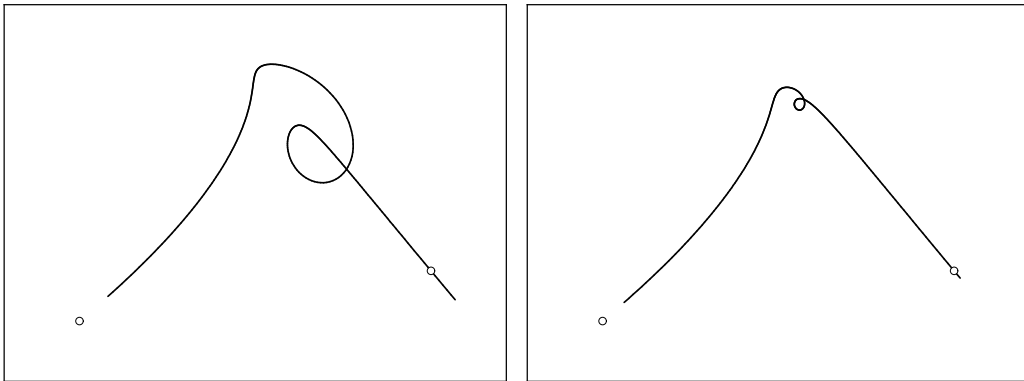


Figure 7: Modified path for feedrate $V = 0.12$ m/s and controller gains k_p different from the nominal value 10 in Figure 3 — left: $k_p = 5$, right: $k_p = 20$.

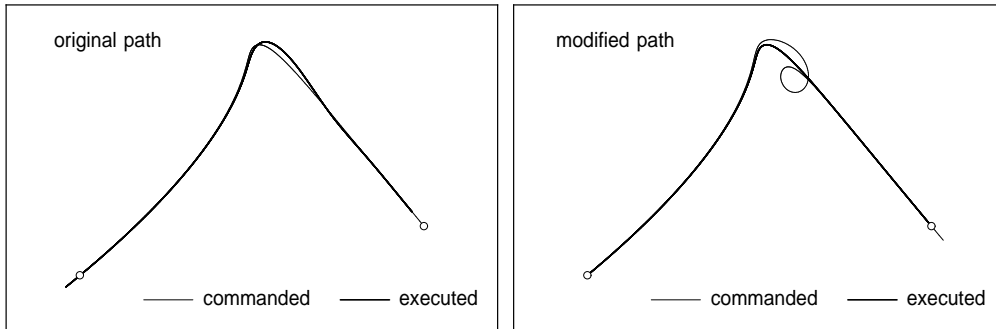


Figure 8: Comparison of commanded and executed motions for original (left) and modified (right) paths. In the former case, the executed motion deviates significantly from the desired path. In the latter case, however, the executed motion is essentially indistinguishable from the original commanded path.

Figure 8 compares commanded and executed trajectories (generated by the MATLAB simulations) for both the original path $\mathbf{R}(\xi)$ and the modified path $\hat{\mathbf{R}}(\xi)$ as input, and a fixed feedrate $V = 0.12$ m/s. For the original path, the executed trajectory deviates appreciably from the commanded path, due to the finite inertia and damping of the machine axes. When the modified path is used as input, on the other hand, the executed trajectory is essentially identical to the desired path $\mathbf{R}(\xi)$. Furthermore, the modified path $\hat{\mathbf{R}}(\xi)$ and induced feedrate⁵ $\hat{V}(\xi)$ in combination yield an executed feedrate essentially identical to the desired constant feedrate $V = 0.12$ m/s. Figure 9 compares the actual feedrates for the original and modified paths, obtained from the MATLAB simulations. The feedrate fluctuations in the former case are rather small, and they are essentially eliminated using the modified path input.

The *position error* is defined, at each instant, as the distance between the actual machine position, and the position along the specified path $\mathbf{R}(\xi)$ that corresponds to the prescribed feedrate V . Figure 10 illustrates the variation of position error obtained from the MATLAB simulations, for both the original and modified paths. This may be viewed as consisting of two components — the *feed error* in the tangent direction, and the *normal error* orthogonal to it (taken as positive on the right of the curve tangent). As noted in Section 1, the feed error amounts to a timing discrepancy along the desired path, rather than a geometrical deviation from it. Figure 10 also shows the normal error,

⁵Note that \hat{V} is generally non-constant, even when V is constant (see §3.3).

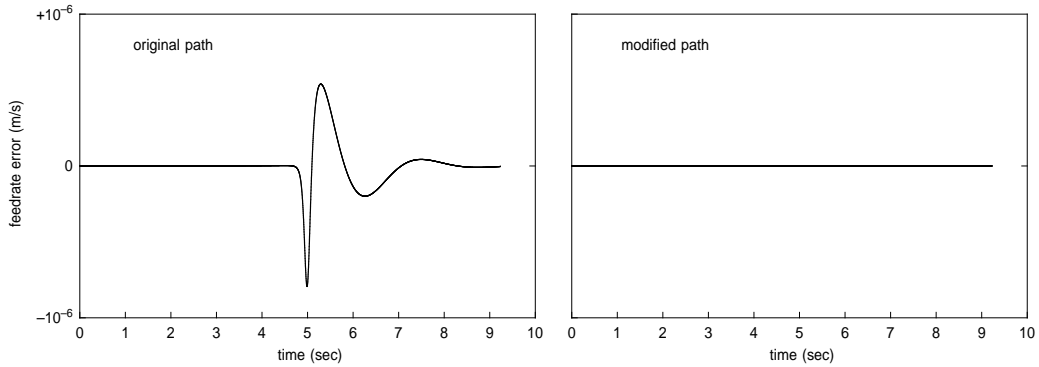


Figure 9: Measured deviation (from MATLAB simulations) of actual feedrate about desired value 0.12 m/s for original path (left) and modified path (right).

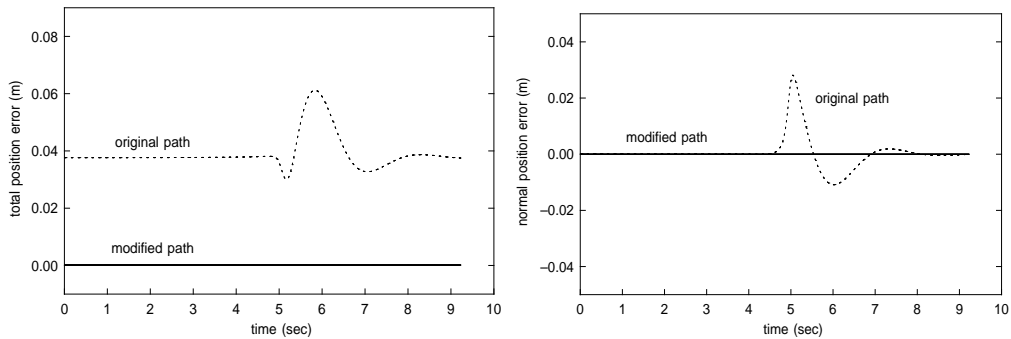


Figure 10: Left: total position error (instantaneous distance between actual and desired position) from MATLAB simulations, for the original and modified paths. Right: Normal component of the total position error for these paths.

a better measure of the actual *contour error* — i.e., the geometrical distance between the desired and executed paths.

The maximum magnitude of the normal error is 0.0377 m for the original path and 0.000139 m for the modified path. Hence, the modified path gives a normal error *more than two orders of magnitude smaller* than the original path. In fact, when using the modified path as input, the normal error of the output path was found to decrease as the sampling frequency was increased, indicating that it is primarily due to discretization of the input.

The *Hausdorff distance* [16] is a more rigorous measure of the difference

between two curves. For arbitrary point sets $S_1, S_2 \subset \mathbb{R}^n$ it is defined by

$$\text{distance}(S_1, S_2) = \max(\rho(S_1, S_2), \rho(S_2, S_1)),$$

where

$$\rho(S_j, S_k) = \max_{\mathbf{p}_j \in S_j} \min_{\mathbf{p}_k \in S_k} |\mathbf{p}_j - \mathbf{p}_k|.$$

The Hausdorff distance of the executed path from the desired path is found to be 0.0376 m for the original path, and 0.000006 m for the modified path.

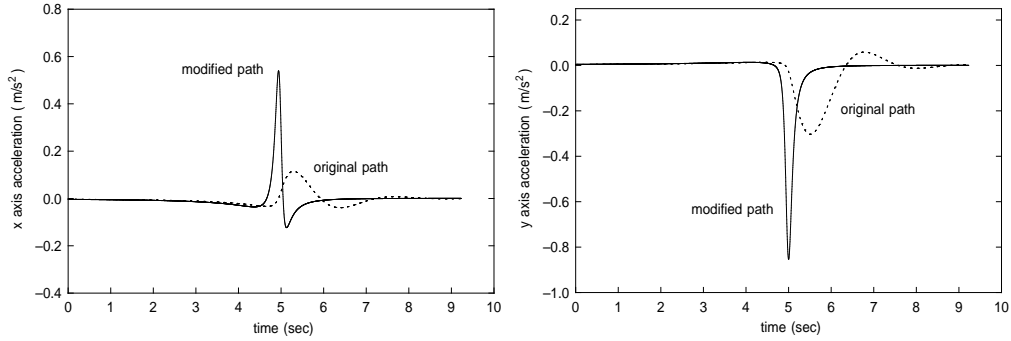


Figure 11: Left: x -axis acceleration from MATLAB simulations for original and modified paths. Right: y -axis acceleration for original and modified paths.

Finally, Figure 11 shows the x and y axis accelerations, using the original and modified paths. The acceleration magnitudes are evidently higher when using the modified path — as expected if the machine is required to traverse regions of strong curvature at high feedrates. In implementing modified paths on real CNC machines, one should verify *a priori* that such high accelerations will not exceed the torque capacity of the axis drive motors.

4.2 PI controller

For a PI controller, we have $d_x = d_y = 0$ and $(c_x, c_y) = (e_x, e_y)$ and by writing $(\hat{X}, \hat{Y}) = (X, Y) + (\Delta X, \Delta Y)$ as before, equations (7) reduce to

$$\begin{aligned} \mu_x \Delta X' + \nu_x \Delta X &= \alpha_x X''' + \beta_x X'' + \tilde{\gamma}_x X', \\ \mu_y \Delta Y' + \nu_y \Delta Y &= \alpha_y Y''' + \beta_y Y'' + \tilde{\gamma}_y Y', \end{aligned} \quad (13)$$

where $\tilde{\gamma}_x = V [(a_x(\sigma V' - 3\sigma'V) + b_x\sigma^2)(\sigma V' - \sigma'V) + a_x\sigma V(\sigma V'' - \sigma''V)]$ and $\tilde{\gamma}_y$ is analogously defined. These are linear, first-order differential equations

for $\Delta X(\xi)$, $\Delta Y(\xi)$ with polynomial coefficients. Since they do not ordinarily possess exact polynomial solutions (see the Appendix), we seek polynomial⁶ *approximations* of the solutions, expressed in the Bernstein basis on $[0, 1]$. For brevity, we shall treat only the x equation in (13).

We denote the Bernstein basis of degree n on $\xi \in [0, 1]$ by

$$b_i^n(\xi) = \binom{n}{i} (1 - \xi)^{n-i} \xi^i, \quad i = 0, \dots, n,$$

and we define $b_i^n(\xi) \equiv 0$ when $i < 0$ or $i > n$. If the commanded path $\mathbf{R}(\xi)$ is a degree- n PH curve, its parametric speed $\sigma(\xi)$ and the feedrate function $V(\xi)$ can be expressed in Bernstein form as

$$\sigma(\xi) = \sum_{i=0}^{n-1} \sigma_i b_i^{n-1}(\xi), \quad V(\xi) = \sum_{i=0}^m V_i b_i^m(\xi).$$

Approximating the x -component of the displacement $\Delta \mathbf{R}(\xi)$ by a polynomial of degree d , we write

$$\begin{aligned} \Delta X(\xi) &= \sum_{i=0}^d \Delta X_i b_i^d(\xi), \\ \Delta X'(\xi) &= \sum_{i=0}^{d-1} d(\Delta X_{i+1} - \Delta X_i) b_i^{d-1}(\xi). \end{aligned} \quad (14)$$

Since equations (13) are first order, initial conditions must be specified for a unique solution. We take $\Delta X(0) = \Delta X_0 = 0$ and $\Delta Y(0) = \Delta Y_0 = 0$, so the modified path coincides with the original commanded path at $\xi = 0$.

The polynomials (6) are constructed in the Bernstein form on $\xi \in [0, 1]$. The right-hand side of the x equation in (13) is then a known polynomial of degree $6n - 5$, which we denote $Q_x(\xi)$. We wish to choose $\Delta X_1, \dots, \Delta X_d$ in (14) to make the left-hand side of (13) agree “as close as possible” to $Q_x(\xi)$.

A number of different methods, with associated measures for closeness of approximation, are possible. By expressing the left and right hand sides in the Chebyshev basis on $[0, 1]$, for example, approximations that minimize

⁶Another possibility is to use piecewise-polynomial (spline) approximations, additional freedoms being obtained by increasing the number of knots rather than the degree.

the maximum error over that interval can be determined. Another approach is a least-squares minimization of the expression

$$F(\Delta X_1, \dots, \Delta X_d) = \int_0^1 [\mu_x(\xi)\Delta X'(\xi) + \nu_x(\xi)\Delta X(\xi) - Q_x(\xi)]^2 d\xi$$

yielding a system of linear equations for $\Delta X_1, \dots, \Delta X_d$. Or for suitable nodes ξ_1, \dots, ξ_d one can obtain these coefficients through an interpolatory process, requiring the left-hand side to agree in value with $Q_x(\xi)$ at the nodes.

Such methods incur important considerations regarding (1) the condition (sensitivity) of the coefficients with respect to changes in the input data; (2) the convergence behavior of the approximants; (3) the ability to ensure that the approximants satisfy a prescribed tolerance; and (4) computational cost of the method. A detailed analysis/comparison of all these factors is beyond our present scope, and we defer it to a later study. At present, we are content to employ a simple approximation scheme to illustrate the path modification for a PI controller, based on choosing the coefficients $\Delta X_1, \dots, \Delta X_d$ so that the left-hand side of (13) agrees with $Q_x(\xi)$ at the d Chebyshev nodes

$$\xi_k = \frac{1}{2} \left[1 - \cos \left(\frac{k\pi}{d-1} \right) \right], \quad k = 0, \dots, d-1. \quad (15)$$

It is well-known [3] that interpolation at the Chebyshev nodes suppresses the spurious oscillations that interpolants on uniform nodes are susceptible to. To avoid numerical stability issues, the linear equations for $\Delta X_1, \dots, \Delta X_d$ were constructed and solved in the MAPLE computer algebra system.

Example 2 For the PI controller, the PH quintic curve $\mathbf{R}(\xi)$ of Example 1 is chosen as the commanded path, again with a constant feedrate $V = 0.12$ m/s and the same physical parameters as before, and $k_p = k_i = 10$ as the P and I controller gains. For the chosen parameters, the closed-loop system is stable with poles at -1.242708 and $-0.628646 \pm 2.458121i$.

Note that the Chebyshev nodes (15) are more densely spaced at the ends of the interval $\xi \in [0, 1]$ than at the center, where the high-curvature region of the path occurs. To ensure accurate approximation of the modified path we choose a relatively high approximant degree, $d = 30$.

Figure 12 compares the commanded and executed motions obtained with the original and modified paths, using the PI controller (it is interesting to compare these with the results from the P controller — see Figure 8). Using

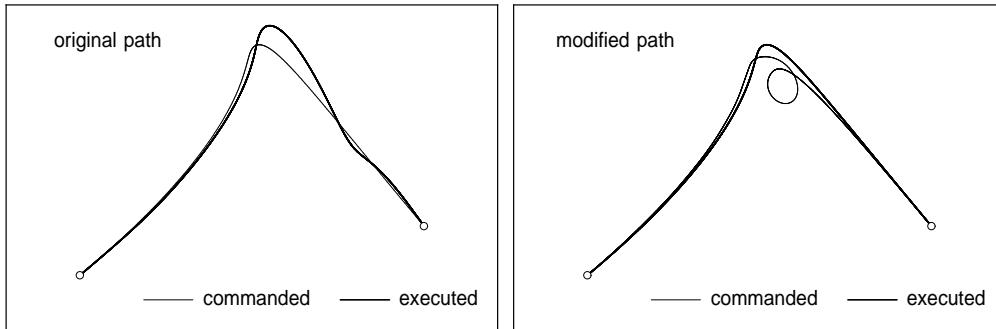


Figure 12: Commanded and executed motions for original (left) and modified (right) paths, using the PI controller — compare with P controller motions in Figure 8. With the original path, the executed motion deviates significantly from the desired path. With the modified path, however, the executed motion is essentially indistinguishable from the original commanded path.

the original path, the executed motion exhibits a very significant deviation from the commanded path, although the integrator in the controller allows the machine to rapidly “recover” from this deviation after it passes the high-curvature region. Using the modified path, on the other hand, the executed motion corresponds to an accurate traversal of the original commanded path, at the commanded feedrate $V = 0.12$ m/s.

Figure 13 compares the total and normal path errors for the original and modified paths obtained with the PI controller. Note that the adopted initial conditions $\Delta X(0) = \Delta Y(0) = 0$ for the differential equations (13) guarantee zero error at the start of the simulation. Also, the presence of an integrator in the controller transfer function effectively suppresses the error for the original path after a while. The normal error results are qualitatively similar to those obtained with the P controller: for the original path, the maximum normal error is of similar magnitude (but opposite sign); for the modified path, the normal error is essentially indistinguishable from measurement noise (caused by discretization of the input).

Figure 14 compares the x and y axis accelerations along the original and modified paths obtained with the PI controller. Because of the new controller transfer function, the accelerations along the original path differ from those for the P controller. On the other hand, the x and y axis accelerations along the modified path obtained with the P and the PI controller are essentially identical, since in both cases the modified path effectively compensates for

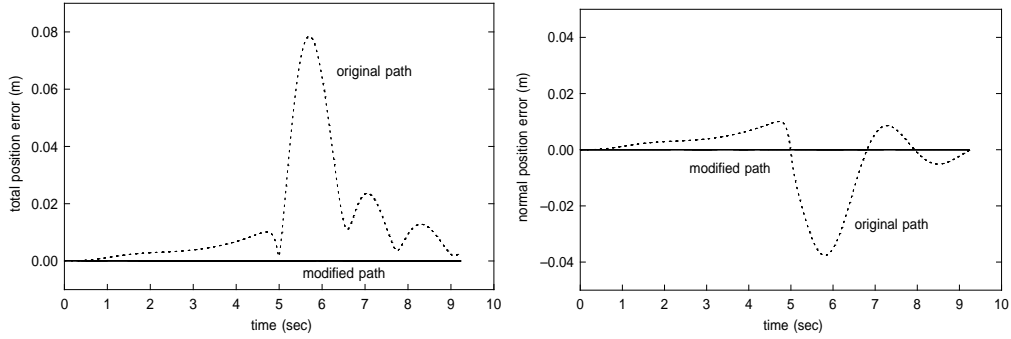


Figure 13: Measured path error (from MATLAB simulations) for the original and modified paths using a PI controller with gains $k_p = k_i = 10$ and feedrate $V = 0.12$ m/s — left: total position error, right: normal position error.

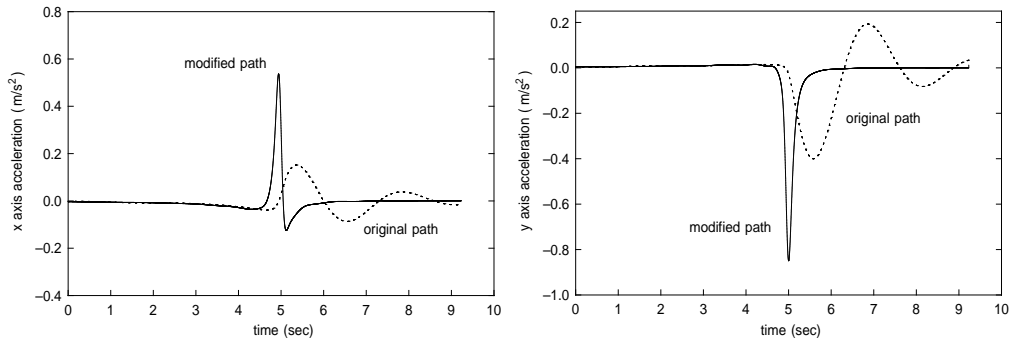


Figure 14: Axis accelerations (from MATLAB simulations) for the original and modified paths when using a PI controller with gains $k_p = k_i = 10$ and feedrate $V = 0.12$ m/s — left: x -axis acceleration, right: y -axis acceleration.

the machine/controller dynamics to yield an output motion corresponding to accurate traversal of the original path at the specified $V = 0.12$ m/s feedrate.

The above approach also applies in the case of a PID controller, differing in the details for the computation of the coefficients of $\Delta X(\xi)$, $\Delta Y(\xi)$ but not in the basic methodology.

5 Closure

A novel approach to compensating for contour errors incurred by the inherent dynamic limitations of CNC machine/controller systems has been introduced, based on computing an *a priori* modification $\hat{\mathbf{R}}(\xi)$ of the commanded path $\mathbf{R}(\xi)$. For a specified feedrate variation $V(\xi)$, the independent variable in the dynamical equations is first transformed from time t to the curve parameter ξ . By reverting the equations (i.e., swapping the input/output roles) a modified path $\hat{\mathbf{R}}(\xi)$ that — modulated by the machine/controller dynamics — exactly yields the desired path $\mathbf{R}(\xi)$ as the physical output, can be characterized as the solution of linear differential equations with polynomial coefficients.

In the case of standard second-order models for machine axis dynamics, a P type controller, and a Pythagorean-hodograph (PH) curve as the original commanded path $\mathbf{R}(\xi)$, the modified path $\hat{\mathbf{R}}(\xi)$ can be exactly described as a higher-order rational curve. Exact closed-form solutions are not possible for more sophisticated controller types, but accurate polynomial approximations can be readily determined through interpolatory or least-squares procedures. These methods will be fully developed and investigated in subsequent papers.

The intent of this paper was to describe the fundamental methodology of the inverse-dynamics path-modification procedure, and to demonstrate its efficacy in reducing contour error through illustrative examples involving P or PI controllers and constant feedrates along paths with strong variations of curvature or parametric speed. More detailed results concerning the practical implementation, experimental performance analysis, and optimal choice of control parameters for the method will be presented in due course.

Appendix: ODEs with polynomial coefficients

When the machine/controller dynamical equations are transformed from the time t to the curve parameter ξ as the independent variable, their coefficients

become polynomials in ξ rather than constants. Although all derivatives of a polynomial function are polynomials of lower degree, differential equations with polynomial coefficients do not (in general) possess polynomial solutions. Since this fact is of fundamental importance to the inverse dynamics problem, but not well-known, we now briefly review from first principles the restricted conditions under which polynomial solutions may exist.

Consider an inhomogeneous linear differential equation of the form

$$a_r(x) \frac{d^r y}{dx^r} + \cdots + a_1(x) \frac{dy}{dx} + a_0(x) y = b(x), \quad (16)$$

where $a_r(x), \dots, a_0(x)$ and $b(x)$ are polynomials. To see why this does not (in general) admit a polynomial solution $y(x)$, let $\deg(a_k) = \ell_k$ for $k = 0, \dots, r$, $\deg(b) = m$, and $y(x) = c_0 + c_1 x + \cdots + c_n x^n$. Recall [2] that the most general solution of (16) is expressed in terms of arbitrary constants $\lambda_1, \dots, \lambda_r$ as

$$y(x) = \lambda_1 y_1(x) + \cdots + \lambda_r y_r(x) + y_p(x),$$

where $y_1(x), \dots, y_r(x)$ are linearly-independent solutions of the homogeneous equation, and $y_p(x)$ is a particular solution of the inhomogeneous equation. Consider first the homogeneous equation, corresponding to $b(x) \equiv 0$ in (16). If $y(x) = c_0 + c_1 x + \cdots + c_n x^n$, the left-hand side is a polynomial of degree

$$d = n + \max_{0 \leq k \leq r} (\ell_k - k) \quad (17)$$

in x , and clearly $d \geq n$. To satisfy the homogeneous equation, this polynomial must vanish identically. This corresponds to requiring the $n + 1$ coefficients c_0, \dots, c_n of $y(x)$ to satisfy $d + 1$ homogeneous linear equations (where $d \geq n$). For a non-trivial solution, $(c_0, \dots, c_n) \neq (0, \dots, 0)$, the matrix of this linear system — with entries defined by the coefficients of $a_0(x), \dots, a_r(x)$ — must be of rank $< n + 1$. Since this deficiency in rank is only achieved for special choices of the polynomials $a_0(x), \dots, a_r(x)$, the solutions $y_1(x), \dots, y_r(x)$ of the homogeneous equation are not, in general, polynomials in x .

Consider now the inhomogeneous equation. The degree (17) of the left-hand side of (16) with $y(x) = c_0 + c_1 x + \cdots + c_n x^n$ must agree with the degree of $b(x)$ — i.e., we must have

$$m = n + \max_{0 \leq k \leq r} (\ell_k - k).$$

If this is satisfied, the differential equation (16) reduces to a system of $m + 1$ linear equations for the $n + 1$ coefficients c_0, \dots, c_n of $y(x)$. Clearly, we must

have $m \leq n$ and hence $\ell_k \leq k$ for $k = 0, \dots, r$ if these equations are not to be over-determined. Specifically, this implies that (16) does not (in general) have a polynomial solution $y(x)$ if $a_0(x), a_1(x), a_2(x), \dots$ are of higher degree than constant, linear, quadratic, \dots in x . Hence, the inhomogeneous equation does not admit polynomial solutions for general choices of $a_0(x), \dots, a_r(x)$.

References

- [1] Y. Altintas (2000), *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design*, Cambridge University Press.
- [2] E. A. Coddington (1961), *An Introduction to Ordinary Differential Equations*, Dover (reprint), New York.
- [3] P. J. Davis (1975), *Interpolation and Approximation*, Dover (reprint), New York.
- [4] A. F. de Souza and R. T. Coelho (2007), Experimental investigation of feed rate limitations on high speed milling aimed at industrial applications, *International Journal of Advanced Manufacturing Technology* **32**, 1104–1114.
- [5] K. Erkorkmaz, C-H. Yeung, and Y. Altintas (2006), Virtual CNC system. Part II. High speed contouring application, *International Journal of Machine Tools and Manufacture* **46**, 1124–1138.
- [6] R. T. Farouki (1994), The conformal map $z \rightarrow z^2$ of the hodograph plane, *Computer Aided Geometric Design* **11**, 363–390.
- [7] R. T. Farouki (2008), *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*, Springer, Berlin.
- [8] R. T. Farouki and T. N. T. Goodman (1996), On the optimal stability of the Bernstein basis, *Mathematics of Computation* **65**, 1553–1566.
- [9] R. T. Farouki and C. A. Neff (1995), Hermite interpolation by Pythagorean-hodograph quintics, *Mathematics of Computation* **64**, 1589–1609.

- [10] R. T. Farouki and V. T. Rajan (1987), On the numerical condition of polynomials in Bernstein form, *Computer Aided Geometric Design* **4**, 191–216.
- [11] R. T. Farouki and V. T. Rajan (1988), Algorithms for polynomials in Bernstein form, *Computer Aided Geometric Design* **5**, 1–26.
- [12] R. T. Farouki and T. Sakkalis (1991), Real rational curves are not “unit speed,” *Computer Aided Geometric Design* **8**, 151–157.
- [13] R. T. Farouki and T. Sakkalis (2007), Rational space curves are not “unit speed,” *Computer Aided Geometric Design* **24**, 238–240.
- [14] R. T. Farouki and S. Shah (1996), Real-time CNC interpolators for Pythagorean-hodograph curves, *Computer Aided Geometric Design* **13**, 583–600.
- [15] R. T. Farouki and Y-F. Tsai (2001), Exact Taylor series coefficients for variable-feedrate CNC curve interpolators, *Computer Aided Design* **33**, 155–165.
- [16] F. Hausdorff (1957), *Set Theory* (translated by J. R. Aumann et al.), Chelsea, New York.
- [17] B. M. Imani and J. Jahanpour (2008), High-speed contouring enhanced with PH curves, *International Journal of Advanced Manufacturing Technology* **37**, 747–759.
- [18] J. Jahanpour and B. M. Imani (2008), Real-time PH curve CNC interpolators for high speed cornering, *International Journal of Advanced Manufacturing Technology* **39**, 302–316.
- [19] R. Komanduri, K. Subramanian, and B. F. von Turkovich (eds.) (1984), *High Speed Machining*, PED-Vol. 12, ASME, New York.
- [20] M. Shpitalni, Y. Koren, and C. C. Lo (1994), Realtime curve interpolators, *Computer Aided Design* **26**, 832–838.
- [21] S. Smith and J. Tlustý (1997), Current trends in high-speed machining, *ASME Journal of Manufacturing Science and Engineering* **119**, 664–666.

- [22] J. Tlusty (1993), High-speed machining, *CIRP Annals* **42**, 733–738.
- [23] Y-F. Tsai and R. T. Farouki (2001), Algorithm 812: BPOLY: An object-oriented library of numerical algorithms for polynomials in Bernstein form, *ACM Transactions on Mathematical Software* **27**, 267–296.
- [24] D. C. H. Yang and T. Kong (1994), Parametric interpolator versus linear interpolator for precision CNC machining, *Computer Aided Design* **26**, 225–234.
- [25] S-S. Yeh and P-L. Hsu (1999), The speed-controlled interpolator for machining parametric curves, *Computer Aided Design* **31**, 349–357.
- [26] C-H. Yeung, Y. Altintas, and K. Erkorkmaz (2006), Virtual CNC system. Part I. System architecture, *International Journal of Machine Tools and Manufacture* **46**, 1107–1123.