Least–squares tool path approximation with Pythagorean–hodograph curves for high–speed CNC machining

Rida T. Farouki, Kazuhiro Saitou, and Yi–Feng Tsai

University of Michigan, Ann Arbor

Abstract

Pythagorean–hodograph (PH) curves admit the formulation of real–time CNC interpolators that are extremely accurate, flexible, and robust. Motivated by the practical benefits of these algorithms in high–speed machining applications, we study the approximation of "traditional" (piecewise–linear/circular) G code part programs by PH curve tool paths. A least–squares fitting approach, entailing the solution of a non–linear system of equations in four variables, is employed to accomplish this approximation. We discuss both the Newton–Raphson and simulated annealing methods for solving this system. We also address issues of tolerance computation, footpoint parameter refinement, penalization of the objective function by the absolute rotation numbers or bending energies of PH curves, and extension of the fitting procedure to 3D tool paths.

1 Introduction

Continuing efforts to reduce manufacturing costs and improve production efficiency have prompted great interest in *high–speed machining* processes, which employ feedrates and spindle speeds an order–of–magnitude or more higher than those of conventional CNC practice [3, 22, 24, 32]. Although there are currently no universally–acknowledged (machine–independent) standards for what constitutes "high–speed machining" (HSM), spindle speeds in the range 10,000–50,000 rpm and feedrates of $\sim$ 1200 in/min ($\sim$ 50 cm/sec) are not uncommon.

Dynamical instability is of central concern in selecting HSM spindle speeds and feedrates. Indeed, some authors *define* the high–speed regime in terms of approach of the cutter tooth engagement frequency with the workpiece to the natural frequency of the machine tool system, which may incur resonant vibration (i.e., chatter) of the cutter [31]. Our present

concern is not with this problem, which has been studied in some depth, but rather with communicating tool path descriptions to the CNC system in a concise manner that is sufficiently flexible to accommodate smooth accelerations to high feedrates; to suppress machining force variations by continuous feedrate variation; and to subdue feedrate fluctuations that arise from the discrete nature of traditional (i.e., piecewise–linear/circular) G code part programs.

Specifically, we shall advocate the use of real–time CNC interpolators for *Pythagorean–hodograph* (PH) *curves* [18] in HSM applications. The PH curves are polynomial parametric curves, compatible with the standard Bézier/B–spline representation [7] of modern CAD systems, whose special algebraic structure makes them uniquely suited to the problem of real–time interpolation at either constant or arc–length/curvature–dependent feedrates. A variety of PH curve interpolators, together with experimental results from their implementation on an open–architecture CNC milling machine, are described in [13, 14, 20, 21] (other authors [5, 6, 26, 29, 30, 34] have also formulated free–form curve interpolators, but their algorithms are based on first–order Taylor approximations and thus lack the accuracy, robustness, and flexibility of the PH curve interpolators).

Real–time PH curve CNC interpolators provide many advantages over "traditional" (linear/circular) interpolators, that are especially valuable in the HSM context — specifically:

- the ability to specify *smooth* accelerations and decelerations along curved tool paths, through the use of feedrate functions with linear or quadratic dependence on the arc length [20];
- the use of curvature–dependent feedrates to reduce machining–force variations due to varying material removal rates at fixed depth of cut along curved tool paths [13];
- the ability to directly interpolate the *offset* at a given fixed distance d from a curved path, for tool–radius compensation [12, 20];
- suppression of feedrate fluctuations incurred by the incompatibility of discrete (i.e., linear/circular) tool path descriptions with smooth realization of high speeds, and hence improved surface finish [14].

Note that, for traditional G code part programs, there is a fundamental conflict between the *accuracy* with which a curved path is specified, and the *smoothness* with which it is traversed at high feedrate. The accuracy of a piecewise–linear/circular tool path approximation may only be improved by increasing the number (and reducing the size) of the approximating segments. However, if ℓ is a typical segment length, V is the feedrate, and

Δt is the system sampling time, feedrate accuracy is maintained only if $V\Delta t \ll \ell$ (i.e., many sampling intervals elapse in traversing a segment).

When $\ell = 1$ mm and $\Delta t = 0.01$ sec, for example, one may expect poor feedrate performance as V approaches 10 cm/sec. This can have a variety of deleterious consequences — including degradation of surface finish due to "jerkiness" of the tool motion, and execution times longer than those nominally expected from the specified paths and speeds. The PH curve interpolators circumvent these problems by generating the timed reference points, required by the control algorithm, directly from the exact *analytic* definitions of the curved tool paths.

A system of G codes for communicating PH tool paths and feedrate functions to CNC machines has been developed [14]. This was designed for compatibility with the existing conventions [1] for linear/circular G codes, allowing the latter to be combined with PH curves in part programs. We encourage the reader to consult the references cited above for background on PH curve interpolators; see also [2, 8, 9, 10, 11, 15, 19] for details on basic properties of, and construction procedures for, PH curves.

2 Tool path approximation

Ideally, tool paths are generated *ab initio* in terms of PH curves in CAD systems, and downloaded to CNC machines incorporating the appropriate software interpolators. However, although PH curves are fully compatible with the infrastructure of existing CAD systems, and intuitive methods can be provided for users to construct and manipulate them, most CAD vendors are entrenched trying to make current functionality work robustly, and exhibit reluctance to venture into uncharted territory.

Thus, as a practical means of taking advantage of the capabilities of PH curve interpolators in HSM, we focus on the problem of approximating "standard" (linear/circular) G code part programs by PH curves to a given tolerance. Our intent is to make use of a least–squares fitting procedure to replace numerous linear/circular motions by relatively few PH segments. Once the geometrical approximation of the tool paths is accomplished, feedrate functions may be imposed on the PH curves to suit requirements of the particular HSM application. A program serving these purposes can then be used as a universally–compatible "post–processing" package (this approach was suggested by Bruce Nourse of Manufacturing Data Systems Inc., Ann Arbor, MI — a supplier of "unbundled" open–architecture software CNC controls). It is expected that the least–squares PH curve fitting methods will be useful in other contexts — e.g., in "reverse engineering."

2.1 Pre–processing steps

We wish to perform least–squares fits of planar PH quintic curves to an ordered sequence of $N+1$ points, $\mathbf{q}_0, \ldots, \mathbf{q}_N$. The first step is to identify subsequences of the point data that have "sufficiently simple" shape to admit accurate fits. One possible strategy is to require these subsequences to be monotone with respect to both the coordinate axes — we say that $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$ is a monotone subsequence if

$$(x_{k+1} - x_k)(x_k - x_{k-1}) > 0 \quad \text{and} \quad (y_{k+1} - y_k)(y_k - y_{k-1}) > 0$$

for $k = m+1, \ldots, m+n-1$, where $\mathbf{q}_k = (x_k, y_k)$. At first, we attempt to fit a single PH quintic $\mathbf{r}(t)$ to each monotone subsequence. To ensure continuity, we require the PH curve to interpolate the first and last points: $\mathbf{r}(0) = \mathbf{q}_m$ and $\mathbf{r}(1) = \mathbf{q}_{m+n}$ (end–tangents might also be fixed to ensure G^1 continuity). If the fit does not satisfy the specified tolerance, we split $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$ into two subsequences, and fit a PH curve to each — this process can be repeated recursively until the desired tolerance is attained.

2.2 Pythagorean–hodograph curves

It is convenient to introduce the *complex representation* [9] for PH curves, in which the x and y components of a plane parametric curve are regarded as real and imaginary parts of a complex function $\mathbf{r}(t) = x(t) + \mathrm{i}\, y(t)$ of a real variable t. The square of any complex polynomial $\mathbf{w}(t) = u(t) + \mathrm{i}\, v(t)$ then yields a hodograph $\mathbf{r}'(t) = x'(t) + \mathrm{i}\, y'(t) = \mathbf{w}^2(t)$ such that

$$x'(t) \;=\; u^2(t) - v^2(t)\,, \quad y'(t) \;=\; 2u(t)v(t)\,, \quad \sigma(t) \;=\; u^2(t) + v^2(t)$$

are three polynomials satisfying the Pythagorean condition

$$x'^2(t) \;+\; y'^2(t) \;\equiv\; \sigma^2(t)\,.$$

Thus, for example, the Bernstein–form complex quadratic

$$\mathbf{w}(t) \;=\; \mathbf{w}_0(1-t)^2 \;+\; \mathbf{w}_1 2(1-t)t \;+\; \mathbf{w}_2 t^2$$

yields, upon integrating $\mathbf{r}'(t) = \mathbf{w}^2(t)$, the Bézier representation

$$\mathbf{r}(t) \;=\; \sum_{i=0}^{5} \mathbf{p}_i b_i^5(t)\,, \qquad b_i^5(t) = \binom{5}{i}(1-t)^{5-i}t^i \tag{2.1}$$

of a planar PH quintic, with control points

$$\begin{aligned}
\mathbf{p}_1 \;&=\; \mathbf{p}_0 \;+\; \mathbf{w}_0^2 \,/\, 5\,,\\
\mathbf{p}_2 \;&=\; \mathbf{p}_1 \;+\; \mathbf{w}_0\mathbf{w}_1 \,/\, 5\,,\\
\mathbf{p}_3 \;&=\; \mathbf{p}_2 \;+\; (2\mathbf{w}_1^2 + \mathbf{w}_2\mathbf{w}_0) \,/\, 15\,,\\
\mathbf{p}_4 \;&=\; \mathbf{p}_3 \;+\; \mathbf{w}_1\mathbf{w}_2 \,/\, 5\,,\\
\mathbf{p}_5 \;&=\; \mathbf{p}_4 \;+\; \mathbf{w}_2^2 \,/\, 5\,,
\end{aligned} \tag{2.2}$$

$\mathbf{p}_0$ being an arbitrary integration constant. We also write the given point data in complex form as $\mathbf{q}_k = x_k + \mathrm{i}\, y_k$ for $k = 0, \ldots, N$.

2.3 Non–linear least–squares fits

Our problem of fitting a PH quintic $\mathbf{r}(t)$ to the points $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$ differs from "ordinary" least–squares fits in two respects. In the latter context, one finds a function $f(t)$ approximating scalar values $f_0, \ldots, f_n$ at given independent–variable values $t_0, \ldots, t_n$. Further, the "fitting parameters," with respect to which the sum of the squared deviations $[\, f(t_k) - f_k \,]^2$ will be minimized, appear *linearly* in the fitting function $f(t)$.

In fitting a parametric curve to point data $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$, the parameter t is the independent variable, and we must identify a "target value" t_k for each point $\mathbf{q}_k$. Ideally, t_k would correspond to the *footpoint* of $\mathbf{q}_k$ on $\mathbf{r}(t)$ — i.e., the value that minimizes $|\, \mathbf{r}(t) - \mathbf{q}_k |$ for $t \in [\, 0, 1]$. In general, this footpoint parameter value is a root of the polynomial equation

$$P_\perp(t) \;=\; \mathbf{r}'(t) \cdot [\, \mathbf{r}(t) - \mathbf{q}_k \,] \;=\; 0\,, \tag{2.3}$$

of degree $2n-1$ if $\mathbf{r}(t)$ is of degree n. Since the coefficients of $\mathbf{r}(t)$ are not known *a priori*, and t_k does not admit a closed–form expression in terms of them if $n \geq 3$, we must rely on fixed *estimates* of $t_m, \ldots, t_{m+n}$ in the fitting procedure. For monotone data, a reasonable choice is

$$t_k \;=\; \frac{1}{2}\left[\, \frac{|\, x_k - x_m|}{|\, x_{m+n} - x_m|} \;+\; \frac{|\, y_k - y_m|}{|\, y_{m+n} - y_m|} \,\right] \tag{2.4}$$

for $k = m, \ldots, m+n$, i.e., the average of the fractional x and y distances of $\mathbf{q}_k = (x_k, y_k)$ along the overall x and y extents of the data $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$.

Given these choices for $t_m, \ldots, t_{m+n}$, our problem is to minimize

$$\Delta(\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2) \;=\; \sum_{k=m}^{m+n} |\, \mathbf{r}(t_k) - \mathbf{q}_k|^2 \tag{2.5}$$

with respect to $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$, subject to the constraint

$$\mathbf{w}_0^2 + \mathbf{w}_0\mathbf{w}_1 + \frac{2\mathbf{w}_1^2 + \mathbf{w}_2\mathbf{w}_0}{3} + \mathbf{w}_1\mathbf{w}_2 + \mathbf{w}_2^2 \;=\; 5\,(\mathbf{q}_{m+n} - \mathbf{q}_m)\,, \tag{2.6}$$

which ensures that $\mathbf{p}_5 = \mathbf{r}(1) = \mathbf{q}_{m+n}$ on taking $\mathbf{p}_0 = \mathbf{r}(0) = \mathbf{q}_m$ in (2.2).

The second difference between our problem and an "ordinary" least–squares fit is now apparent — the fitting function $\mathbf{r}(t)$ defined by equations (2.1) and (2.2) depends *non–linearly* on the fitting parameters $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$. Hence, we cannot expect to obtain a linear system by equating to zero the derivatives of (2.5) with respect to these parameters.

Equation (2.6) defines a quadric surface — specifically, an ellipsoid — in the complex three–dimensional space $\mathbf{C}^3$ with coordinates $(\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2)$. In order to eliminate this constraint, we parameterize the quadric in terms of complex–angle variables $\boldsymbol{\theta} = \theta + \mathrm{i}\,\vartheta$ and $\boldsymbol{\phi} = \phi + \mathrm{i}\,\varphi$ as

$$\begin{aligned}
\mathbf{w}_0 &= \sqrt{\mathbf{p}_f - \mathbf{p}_s}\,\left(\sin\boldsymbol{\theta} \;+\; \sqrt{5}\cos\boldsymbol{\theta}\cos\boldsymbol{\phi} \;-\; \sqrt{3}\cos\boldsymbol{\theta}\sin\boldsymbol{\phi}\right)\,, \\
\mathbf{w}_1 &= \sqrt{\mathbf{p}_f - \mathbf{p}_s}\,\left(\sin\boldsymbol{\theta} \;-\; 2\sqrt{5}\cos\boldsymbol{\theta}\cos\boldsymbol{\phi}\right)\,, \\
\mathbf{w}_2 &= \sqrt{\mathbf{p}_f - \mathbf{p}_s}\,\left(\sin\boldsymbol{\theta} \;+\; \sqrt{5}\cos\boldsymbol{\theta}\cos\boldsymbol{\phi} \;+\; \sqrt{3}\cos\boldsymbol{\theta}\sin\boldsymbol{\phi}\right)\,. \qquad (2.7)
\end{aligned}$$

where we write $\mathbf{p}_s = \mathbf{q}_m$ and $\mathbf{p}_f = \mathbf{q}_{m+n}$. Here, the complex trigonometric functions can be written explicitly in terms of real and imaginary parts as

$$\begin{aligned}
\sin\boldsymbol{\theta} &= \sin(\theta + \mathrm{i}\,\vartheta) = \sin\theta\cosh\vartheta \;+\; \mathrm{i}\,\cos\theta\sinh\vartheta\,, \\
\cos\boldsymbol{\theta} &= \cos(\theta + \mathrm{i}\,\vartheta) = \cos\theta\cosh\vartheta \;-\; \mathrm{i}\,\sin\theta\sinh\vartheta\,, \qquad (2.8)
\end{aligned}$$

and similarly for $\boldsymbol{\phi} = \phi + \mathrm{i}\,\varphi$ (note that these functions are periodic with respect to the real part, but not the imaginary part, of the argument).

The parameterization (2.7) of the "constraint quadric" arises from an orthogonal transformation that diagonalizes equation (2.6) — it is easily verified through direct substitution. Finally, we simplify the control point expressions (2.2) by using (2.6) to obtain

$$\begin{aligned}
\mathbf{p}_0 &= \mathbf{p}_s\,, \\
\mathbf{p}_1 &= \mathbf{p}_s \;+\; \mathbf{w}_0^2\,/\,5\,, \\
\mathbf{p}_2 &= \mathbf{p}_s \;+\; (\mathbf{w}_0^2 + \mathbf{w}_0\mathbf{w}_1)\,/\,5\,, \\
\mathbf{p}_3 &= \mathbf{p}_f \;-\; (\mathbf{w}_1\mathbf{w}_2 + \mathbf{w}_2^2)\,/\,5\,, \\
\mathbf{p}_4 &= \mathbf{p}_f \;-\; \mathbf{w}_2^2\,/\,5\,, \\
\mathbf{p}_5 &= \mathbf{p}_f\,. \qquad (2.9)
\end{aligned}$$

We can thus re–formulate the least–squares fitting problem in terms of an *unconstrained* minimization of the objective function

$$\Delta(\boldsymbol{\theta}, \boldsymbol{\phi}) \;=\; \sum_{k=m}^{m+n} |\,\mathbf{r}(t_k) - \mathbf{q}_k\,|^2\,, \qquad (2.10)$$

where the dependence of $\mathbf{r}(t)$ on the complex fitting parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ is defined sequentially through equations (2.1), (2.9), and (2.7).

3 Newton–Raphson scheme

We can minimize (2.10) by setting its derivatives with respect to the fitting parameters equal to zero, and solving the resulting system of non–linear

equations. However, since the modulus $|\mathbf{z}|$ of a complex variable $\mathbf{z}$ is not an analytic function of that variable, we cannot formulate this minimization directly in terms of the complex unknowns $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ — we have to express (2.10) explicitly in terms of their real and imaginary parts (θ, ϑ) and (ϕ, φ). In other words, we re–write (2.10) in the form

$$\Delta(\theta, \vartheta, \phi, \varphi) \;=\; \sum_{k=m}^{m+n} [\, x(t_k) - x_k \,]^2 \;+\; [\, y(t_k) - y_k \,]^2 \,, \tag{3.1}$$

where $x(t)$ and $y(t)$ are obtained by substituting (2.8), and the analogous expressions for $\boldsymbol{\phi}$, into (2.7), (2.9), and (2.1), and separating the result into real and imaginary parts. Since the resulting expressions are somewhat cumbersome, we refrain from explicitly enumerating them.

For convenience of notation, we shall now write

$$\mathbf{v} \;=\; [\; v_1 \; v_2 \; v_3 \; v_4 \;]^T \;=\; [\; \theta \; \vartheta \; \phi \; \varphi \;]^T \,, \tag{3.2}$$

and

$$\mathbf{f} \;=\; [\; f_1 \; f_2 \; f_3 \; f_4 \;]^T \;=\; \left[\; \frac{\partial \Delta}{\partial v_1} \; \frac{\partial \Delta}{\partial v_2} \; \frac{\partial \Delta}{\partial v_3} \; \frac{\partial \Delta}{\partial v_4} \;\right]^T .$$

Thus, the least–squares fitting problem amounts to solving the non–linear system of equations $\mathbf{f}(\mathbf{v}) = \mathbf{0}$, where the component functions $f_1, \ldots, f_4$ of the variables $v_1, \ldots, v_4$ are defined by

$$f_i \;=\; \frac{\partial \Delta}{\partial v_i} \;=\; 2 \sum_{k=m}^{m+n} [\, x(t_k) - x_k \,] \frac{\partial x}{\partial v_i}(t_k) \;+\; [\, y(t_k) - y_k \,] \frac{\partial y}{\partial v_i}(t_k) \;. \tag{3.3}$$

In general, the non–linear system $\mathbf{f}(\mathbf{v}) = \mathbf{0}$ admits no simple closed–form solution, and we must resort to the use of numerical methods.

In this section we discuss a multivariate Newton–Raphson scheme that is efficient, but requires an initial "guess" for the solution values. However, since there is in fact a multiplicity of solutions, the computed solution can be sensitive to the initial guess, and may not identify the global minimum of (2.10). The simulated annealing method, described in the next section, is guaranteed (under suitable conditions) to find the global minimum.

The *Hessian matrix* $\mathbf{M}$ for the function Δ has elements

$$M_{ij} \;=\; \frac{\partial^2 \Delta}{\partial v_i \partial v_j} \qquad \text{for } \; 1 \le i, j \le 4$$

— in terms of $x(t)$ and $y(t)$, these elements are given explicitly by

$$\begin{aligned} M_{ij} \;=\; 2 \sum_{k=m}^{m+n} \Big\{ & [\, x(t_k) - x_k \,] \frac{\partial^2 x}{\partial v_i \partial v_j}(t_k) \;+\; [\, y(t_k) - y_k \,] \frac{\partial^2 y}{\partial v_i \partial v_j}(t_k) \\ & + \; \frac{\partial x}{\partial v_i}(t_k) \, \frac{\partial x}{\partial v_j}(t_k) \;+\; \frac{\partial y}{\partial v_i}(t_k) \, \frac{\partial y}{\partial v_j}(t_k) \Big\} \,. \end{aligned} \tag{3.4}$$

The Newton–Raphson scheme provides a sequence of approximations $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$, ... from a given starting approximation $\mathbf{v}^{(0)}$, that converges rapidly to an exact solution of $\mathbf{f}(\mathbf{v}) = \mathbf{0}$ under suitable conditions. The $(k+1)$–th approximation is given in terms of the k–th approximation by

$$\mathbf{v}^{(k+1)} \;=\; \mathbf{v}^{(k)} \;-\; \mathbf{M}^{-1}(\mathbf{v}^{(k)})\,\mathbf{f}(\mathbf{v}^{(k)}) \quad \text{for} \;\; k = 0, 1, 2, \ldots \tag{3.5}$$

where, on the right–hand side, the inverse $\mathbf{M}^{-1}$ of the Hessian and the vector $\mathbf{f}$ of function values are evaluated at the k–th iteration, $\mathbf{v}^{(k)}$.

The convergence of the Newton–Raphson iterations can be monitored by a suitable measure of fractional changes in the solution values, such as

$$\epsilon_{k+1} \;=\; \frac{\|\mathbf{v}^{(k+1)} - \mathbf{v}^{(k)}\|_2}{\|\mathbf{v}^{(k)}\|_2}\,,$$

where $\|\mathbf{v}\|_2 = \sqrt{v_1^2 + \cdots + v_6^2}$, or of the residual function values, such as

$$\eta_{k+1} \;=\; \|\mathbf{f}^{(k+1)}\|_2\,.$$

To start the Newton–Raphson iterations (3.5), we require initial values for the unknowns (3.2) — i.e., for the complex values $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. We use the PH quintic Hermite interpolation algorithm [15], which requires two end points $\mathbf{r}(0)$, $\mathbf{r}(1)$ and derivatives $\mathbf{r}'(0)$, $\mathbf{r}'(1)$ as input, to estimate these initial values. For end points we have $\mathbf{r}(0) = \mathbf{p}_s$ and $\mathbf{r}(1) = \mathbf{p}_f$, while for the end derivatives we estimate their orientations by linear least–squares fits to a few points $\mathbf{q}_{m-r}, \ldots, \mathbf{q}_{m+r}$ and $\mathbf{q}_{m+n-r}, \ldots, \mathbf{q}_{m+n+r}$ neighboring $\mathbf{p}_s = \mathbf{q}_m$ and $\mathbf{p}_f = \mathbf{q}_{m+n}$, and we assign their magnitudes to be $\lambda\,|\mathbf{p}_f - \mathbf{p}_s|$, where λ is a numerical scale factor of order unity.

Note that the PH quintic Hermite interpolation problem actually has *four* distinct solutions, and it is essential to select the "good" interpolant — having the smallest *absolute rotation number* [15] — from among them. The values $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$ are computed directly by the Hermite interpolation algorithm; $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can then be obtained through the relations

$$\sin\boldsymbol{\theta} \;=\; \frac{\mathbf{w}_0 + \mathbf{w}_1 + \mathbf{w}_2}{3\,(\mathbf{p}_f - \mathbf{p}_s)} \qquad \text{and} \qquad \tan\boldsymbol{\phi} \;=\; \sqrt{15}\,\frac{\mathbf{w}_2 - \mathbf{w}_0}{\mathbf{w}_2 - 2\mathbf{w}_1 + \mathbf{w}_0}\,.$$

Tests with the Newton–Raphson method reveal that the function (2.10) has many distinct local minima, which do not all identify acceptable least–squares fits to the data. We have used this method only for exploratory purposes — because of the difficulty of guaranteeing identification of the global minimum (or even just a "reasonable" least–squares fit), we do not recommend this approach for practical use. We now turn our attention to the simulated annealing method, which offers much better performance in this respect — although at greater computational cost.

4 Simulated annealing method

The Newton–Raphson scheme is an example of a 2nd–order "local descent" minimization method — it uses the gradient vector $\partial\Delta/\partial v_i$ and Hessian matrix $\partial^2\Delta/\partial v_i \partial v_j$ of the objective function (2.10). A basic defect of this method is that it cannot guarantee convergence to the *global* minimum of (2.10): it typically converges to a "local" minimum, determined entirely by the choice of the initial guess for the variables $\mathbf{v}$. To address this problem, we have investigated the application of the *simulated annealing algorithm* [23] — a global heuristic search scheme inspired by thermal annealing of critically heated solids, to the minimization of the function (2.10).

The basic idea of the simulated annealing algorithm is motivated by the physical process of *slowly* cooling a molten material from an initial high temperature to well below its freezing point, so that it solidifies into a near–perfect crystal structure — a process called *annealing* [23]. A perfect crystal corresponds to an absolute minimum of the energy associated with the configuration of its constituent particles. Departures from this ideal — i.e., "crystal defects" — will correspond to higher energies, which may nevertheless be *local* minima (compared to "neighboring" states) of the energy. The random thermal motions associated with each temperature allow the system to "seek" the lowest energy state, and the cooling must proceed at a sufficiently slow rate to prevent "trapping" in local energy minima (corresponding to defective crystal structures).

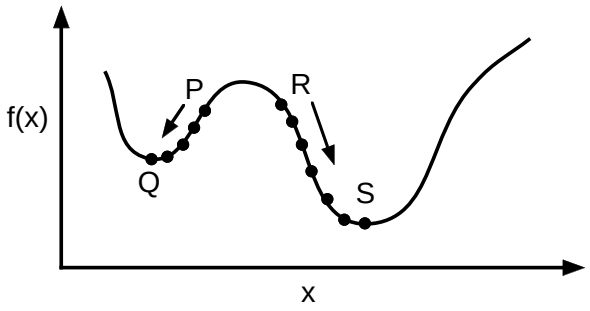

FIGURE 1. Local descent method applied to a univariate function $f(x)$.

The simulated annealing method interprets the value of a function, whose global minimum is sought, as an "energy" value. Random changes in the variables of that function correspond to thermal motions of particles, and a "temperature" variable controls their magnitude. The temperature slowly decreases as the algorithm proceeds, thereby gradually inhibiting

these random changes. Under suitable conditions, it is guaranteed that the final "frozen" state of the system will identify the global minimum.

In local descent methods, the variables are always changed in a manner that attempts to reduce the function value. As noted above, such methods often result in convergence to a local (rather than the global) minimum, and the final result is completely dependent on the chosen starting point. Figure 1 illustrates this for a univariate function $f(x)$. Starting from point P yields convergence to the local minimum at Q, while starting at R will give convergence to S (which happens to be the global minimum).

Clearly, to achieve convergence to the global minimum from *arbitrary* starting points, an algorithm must be capable of executing "uphill" steps, i.e., movements from the current point to one with a higher function value (a "worse" point). However, this must be done in a "controlled" manner, that does not jeopardize eventual convergence to the global minimum; see Figure 2. The simulated annealing algorithm incorporates this property.

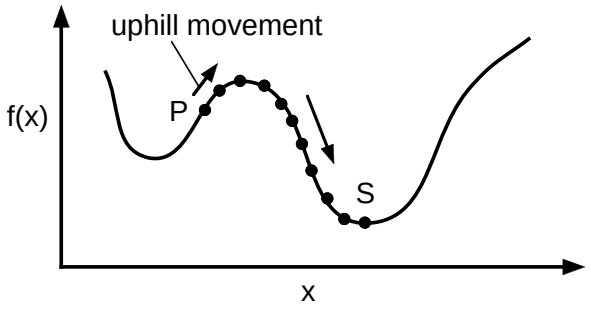

FIGURE 2. Uphill motion required for convergence to global minimum.

In simulated annealing, the occurrence of uphill motion is controlled in a probabilistic manner, motivated by concepts from statistical mechanics. A step involving a *decrease* in the function value is always taken, whereas a step that involves an increase by Δf is only taken with probability

$$\textit{probability}\,(\text{uphill movement}) \;=\; \exp\left(-\frac{\Delta f}{kT}\right), \tag{4.1}$$

where k is a positive constant — analogous to the Boltzmann constant of thermodynamics — and $T > 0$ is the current "temperature." Note that, according to (4.1), uphill movements are more likely for small increases Δf in the function value, and at higher temperatures T. A step governed by these principles is known as a *Metropolis step* of the algorithm, after an author of pioneering simulations [28] of statistical mechanics.

A typical implementation commences with a random start point and a very high temperature. A "neighboring" point is randomly sampled, and selected with certainty as the new point if the function value has decreased. If the function value has increased, it is accepted with the probability (4.1). These Metropolis steps are repeated at each temperature, until a suitable termination criterion is satisfied: either the system "equilibriates" at that temperature, or a prescribed bound on the number of steps is attained. The temperature is then decreased according to the *temperature schedule* — a pre–determined sequence of monotonically decreasing values.

The choice of temperature schedule, and the size of the neighborhoods used to select new points, have a significant impact on the performance of the algorithm. The best choices are generally dependent on the properties of the function being minimized. The examples shown here have employed a modified version of the temperature schedule suggested in [27]:

$$T^{\,\mathrm{new}} \;=\; \frac{T}{1+\beta\, n}\,, \tag{4.2}$$

where T and $T^{\,\mathrm{new}}$ are the current and new temperatures, $\beta \ll 1$ is a positive constant, and n counts the number of temperatures employed thus far. The neighborhoods from which new points are selected are defined by

$$v_i^{\,\mathrm{new}} \;=\; v_i \,+\, \rho\, T\, \tan(r) \qquad \text{for } \; i = 1, \ldots, 6\,, \tag{4.3}$$

where v_i and $v_i^{\,\mathrm{new}}$ are the i-th coordinates of the current and new points, r is a uniformly–distributed random number between $-\pi/2$ and $\pi/2$, and ρ is a positive constant. Note that the probability density function for the random variable $\epsilon = \rho\, T\, \tan(r)$ corresponds to the Cauchy distribution:

$$f(\epsilon) \;=\; \frac{1}{\pi}\,\frac{\rho\, T}{(\rho\, T)^2 + \epsilon^2}\,. \tag{4.4}$$

Figure 3 illustrates the shape of $f(\epsilon)$ for $\rho = 2$ and $T = 10,\, 5,\, 1$. As shown in this figure, the sampling of a new points is biased toward $\epsilon = 0$, and this bias increases rapidly as the temperature T decreases — in fact $f(\epsilon)$ coincides, in the limit $\rho\, T \to 0$, with the Dirac δ–function.

Figure 4 shows (on a logarithmic scale) the variation of the objective function (2.10) with the temperature counter n in equation (4.2) during a typical simulated annealing run. Two examples of PH quintics computed by the simulated annealing method (fits to 10 data points) are shown in Figure 5. These illustrate the ability of PH quintics to accurately model both convex and inflected smooth data — the quality of fit is excellent in both cases. The point data in the second case is actually from a circular arc, and the fit is clearly very accurate even though circular arcs cannot be represented *exactly* as (polynomial) PH curves.

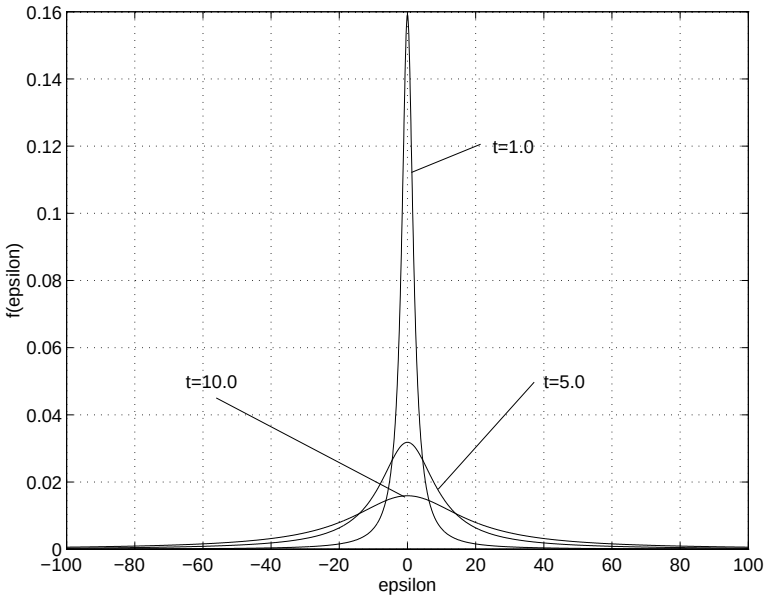

FIGURE 3. Shape of the distribution $f(\epsilon)$ for various temperatures T.

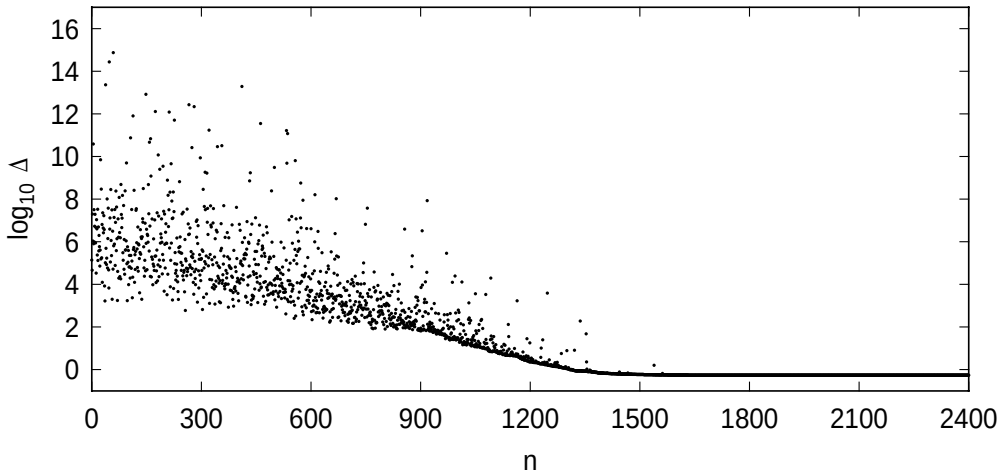

FIGURE 4. Value of (2.10) during a typical simulated annealing run.

In Figure 6 we have added random "noise" to the data points for the examples in Figure 5, and then computed new least–squares fits based on this noisy data. The resulting fits are remarkably similar to those obtained from the original "smooth" data — indicating that the fitting process is quite insensitive to (small) random perturbations of the input.

Finally, Figure 7 shows the tendency of PH quintics to "smooth out" sharp directional changes in the discrete point data (in both examples, the points lie on straight–line segments that meet at sharp angles). This

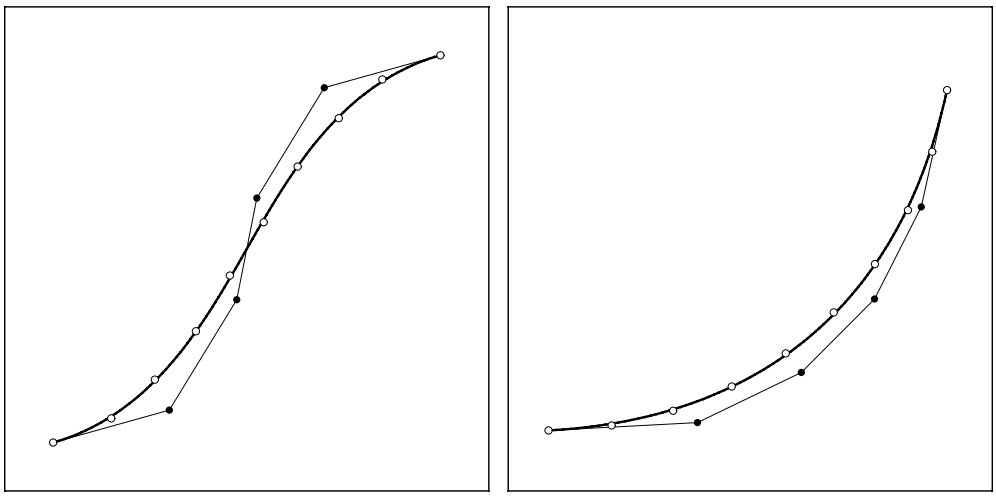

FIGURE 5. Examples of least–squares PH quintic fits to smooth point data, computed by means of the simulated annealing algorithm — open dots indicate the point data, and the solid dots are Bézier control points.

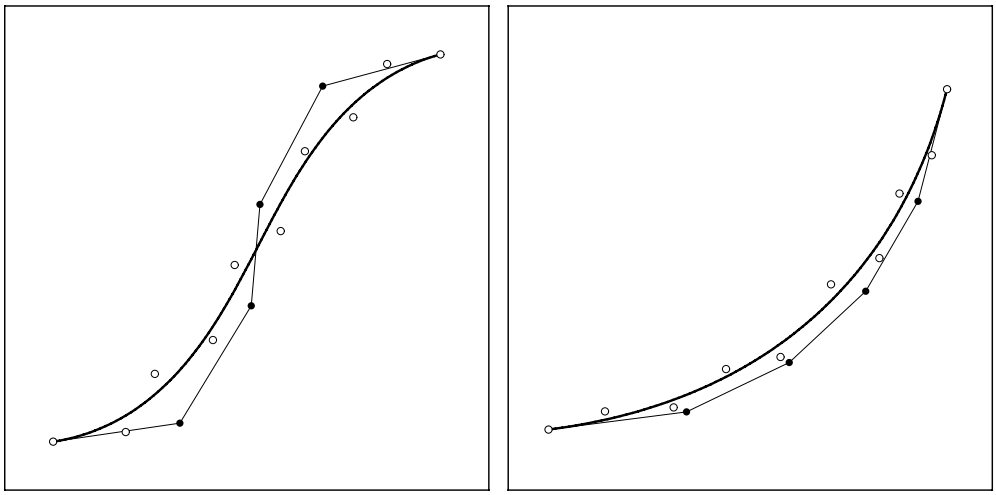

FIGURE 6. PH quintic fits to the data in Figure 5 with superimposed "noise" — note that the curves are relatively insensitive to this noise.

smoothing effect can be a desirable attribute for HSM applications, since sudden changes of direction imply very high accelerations.

An advantage of simulated annealing over the Newton–Raphson method is that it uses only *values* of the function (2.10), and not its *derivatives*. If the function evaluation is written efficiently, simulated annealing can be quite fast: the examples shown in Figure 5 required only a few seconds on an HP wokstation, although the runs involve $\sim 10^5$ evaluations.

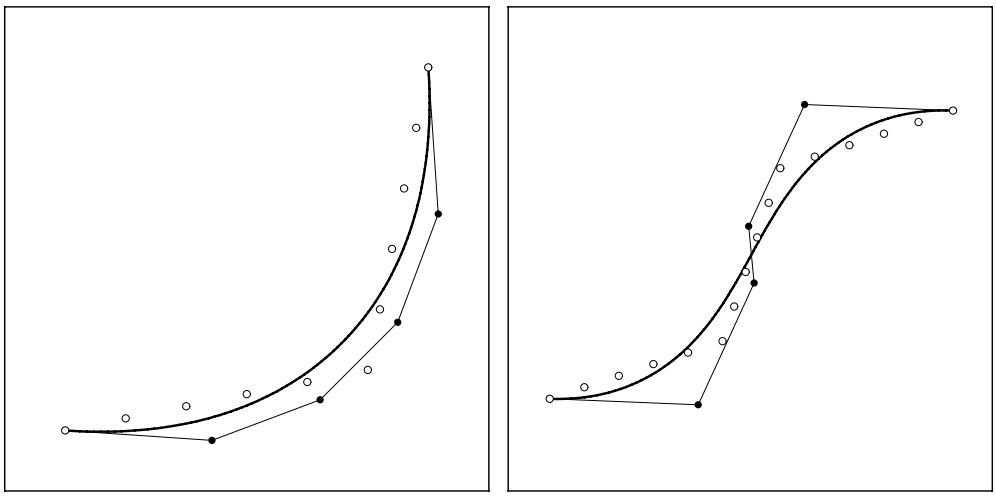

FIGURE 7. Two examples illustrating the smoothing of sharp corners by least–squares PH quintic fits to sets of piecewise–linear point data.

5 Tolerance estimation

To determine the accuracy of fit of the PH quintic $\mathbf{r}(t)$ to $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$, we need to compute the true footpoint parameter values of these data points on $\mathbf{r}(t)$. This incurs a polynomial root–solving problem, for which suitable methods [25] based upon the numerically–stable Bernstein representation [16, 17] are available. Thus, if (2.3) has $r \geq 0$ real roots $t_1, \ldots, t_r$ of odd multiplicity on $t \in [\,0, 1\,]$, and we set $t_0 = 0$ and $t_{r+1} = 1$, we may define

$$\delta_k \;=\; \min_{0 \leq i \leq r+1} |\,\mathbf{r}(t_i) - \mathbf{q}_k\,| \tag{5.1}$$

for $k = m+1, \ldots, m+n-1$ as the true distance of $\mathbf{q}_k$ from $\mathbf{r}(t)$; note that $\delta_m = \delta_{m+n} = 0$ by definition. We may then use

$$\delta_{\mathrm{rms}} \;=\; \sqrt{\frac{1}{n-1} \sum_{k=m+1}^{m+n-1} \delta_k^2} \qquad \text{or} \qquad \delta_{\mathrm{max}} \;=\; \max_{m+1 \leq k \leq m+n-1} \delta_k \tag{5.2}$$

as appropriate measures for the overall quality of fit. These can be used to ensure that the approximation satisfies a given tolerance, and to indicate a need for subdividing the data $\mathbf{q}_m, \ldots, \mathbf{q}_{m+n}$ if the tolerance is not satisfied.

6 Footpoint parameter refinement

The initial estimates (2.4) of footpoint parameter values for the data points $\mathbf{q}_k$ can significantly influence the quality of the fit. Thus, as an alternative

(or in addition) to subdividing the data into smaller sequences to improve the accuracy of fit, one may also employ a procedure for refinement of these footpoint parameter values. Namely, one first performs a fit based on (2.4) and then, once the errors (5.1) and corresponding true footpoints on the fitted curve are computed, the latter can be used in lieu of (2.4) to perform a *new* fit. This process may be repeated, but further improvement in the quality of fit — as measured by (5.2) — is usually marginal.

7 Control of tangent behavior

As noted in Section 3 above, the PH quintic Hermite interpolation problem admits four distinct solutions, three of which exhibit undesired "looping" behavior. The "good" solution is distinguished by having the least *total* — i.e., without clockwise/anticlockwise cancellation — tangent rotation, as characterized by the *absolute rotation number*

$$\mathcal{R}_{\text{abs}} \;=\; \frac{1}{2\pi}\int_0^1 |\kappa(t)|\;|\mathbf{r}'(t)|\;\mathrm{d}t\,, \tag{7.1}$$

which can be calculated through an analytic reduction of the integral [15]. For PH quintics, this quantity lies in the range $0 \le \mathcal{R}_{\text{abs}} \le 2$.

To compute (7.1) it is convenient [15] to express the hodograph $\mathbf{r}'(t) = [\,\mathbf{w}_0\,(1-t)^2 + \mathbf{w}_1\,2(1-t)t + \mathbf{w}_2\,t^2\,]^2$ in the form $\mathbf{r}'(t) = \mathbf{k}\,(t-\mathbf{a})^2(t-\mathbf{b})^2$. From the output $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ of the fitting process, we obtain $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$ from (2.7), and the complex values $\mathbf{a}$ and $\mathbf{b}$ required to compute $\mathcal{R}_{\text{abs}}$ may then be written as

$$\mathbf{a} \;=\; \frac{\boldsymbol{\alpha}}{\boldsymbol{\alpha}+1} \qquad \text{and} \qquad \mathbf{b} \;=\; \frac{\boldsymbol{\beta}}{\boldsymbol{\beta}+1}\,,$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the two roots $\mathbf{z}$ of the complex quadratic equation

$$\mathbf{w}_2\,\mathbf{z}^2 \;+\; 2\,\mathbf{w}_1\,\mathbf{z} \;+\; \mathbf{w}_0 \;=\; 0\,.$$

Now the PH quintics corresponding to (local) minima of the function (2.10) do not necessarily exhibit "well–behaved" tangent variations (this outcome is less likely with simulated annealing than the Newton–Raphson method, however, since "bad" tangent behavior is usually incompatible with a low value of (2.10)). To ensure a solution with a "reasonable" $\mathcal{R}_{\text{abs}}$ value, and to help accelerate convergence toward such a solution, we have also tried the modified objective function defined by

$$\Omega(\boldsymbol{\theta},\boldsymbol{\phi}) \;=\; \frac{\Delta(\boldsymbol{\theta},\boldsymbol{\phi})}{\langle d^2\rangle} \;+\; \mu\,\mathcal{R}_{\text{abs}}(\boldsymbol{\theta},\boldsymbol{\phi})\,, \tag{7.2}$$

where μ is an adjustable "weight" factor of order unity, and we define

$$\langle d^2 \rangle \;=\; \frac{1}{n} \sum_{k=m}^{m+n-1} |\, \mathbf{q}_{k+1} - \mathbf{q}_k \,|^2 \,.$$

We introduce the division of Δ by this quantity to ensure scale–invariance of the new objective function (7.2) — note that Δ has dimension (length)^2 while $\mathcal{R}_{\rm abs}$ is evidently dimensionless.

The objective function (7.2) attempts to identify a PH quintic that is both "faithful" to the given point data, and also "smooth" in the sense of having a subdued tangent variation. In this context, the weight μ serves as a useful "shape control" parameter — when $\mu \gg 1$ the emphasis is on identifying a smooth fit at the expense of the accuracy with which it models the point data, whereas when $\mu \ll 1$ the fitted curve is allowed to "wiggle" so as to more closely approximate the data. Examples of PH quintics obtained by simulated annealing applied to the modified objective function (7.2) are shown in Figure 8.

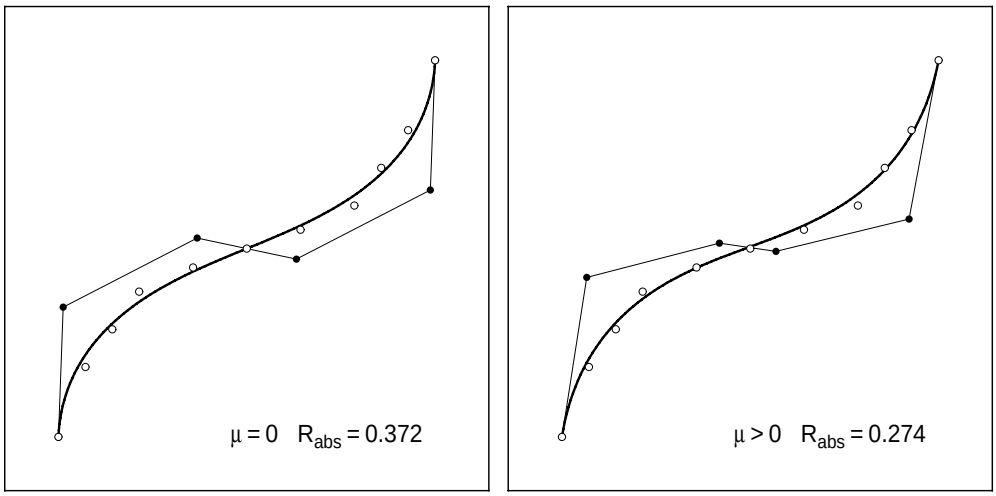

FIGURE 8. Typical examples of "smooth" PH quintic fits obtained by simulated annealing with the modified objective function defined in (7.2).

Finally, we mention another approach to the smoothing–fitting process based upon the objective function

$$\Gamma(\boldsymbol{\theta}, \boldsymbol{\phi}) \;=\; \frac{\Delta(\boldsymbol{\theta}, \boldsymbol{\phi})}{\langle d^2 \rangle} \;+\; \mu \, \frac{\langle d^2 \rangle}{S} \, \mathcal{E}(\boldsymbol{\theta}, \boldsymbol{\phi}) \,, \qquad (7.3)$$

where S denotes the total arc length of $\mathbf{r}(t)$, and its *elastic bending energy* is defined [10] by

$$\mathcal{E} \;=\; \int_0^1 \kappa^2(t) \; |\mathbf{r}'(t)| \; \mathrm{d}t \,.$$

$\mathcal{E}$ can also be evaluated in closed–form from the complex values $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$ — obtained from $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ through (2.7) — although this is rather more involved [10] than evaluation of (7.1). Again, the factors of $\langle d^2 \rangle$ in (7.3) are introduced to render this function dimensionless and scale–invariant. Note that we employ the bending energy *per unit length* — not the total bending energy — in (7.3), since the latter becomes arbitrarily small [4] for a loop of radius $\sim R$ and curvature $\sim R^{-1}$ as we let $R \to \infty$.

8 Extension to 3D tool paths

For brevity, we have only discussed the approximation of *planar* tool paths by PH curves. The methods can be extended to three–dimensional paths, using the PH space curves [19]. The principal difficulty in formulating this extension is the absence of a compact representation scheme for PH space curves, analogous to the complex form of planar PH curves — see, however, the quaternion methods proposed in [33]. Nevertheless, one can write down a function analogous to (3.1) in terms of *nine* unknown real quantities $\alpha_0, \beta_0, \gamma_0, \alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2$ whose minimization — subject to *three* (scalar) constraints — defines least–squares PH quintic space curve fits to sequences of spatial point data. However, reformulating this as an unconstrained minimization problem in six variables is more challenging.

9 Closure

We have described methods to perform least–squares fits of PH quintics to discrete point data, motivated by the desire to approximate "standard" (linear/circular) G code part programs by PH tool paths. The latter offer concise part programs with continuously–variable feedrate capability to ensure smooth acceleration/deceleration, control over machining forces, optimization of total machining times. Furthermore, feedrate fluctuations due to the discrete nature of standard G code part programs are subdued, offering potential improvements in surface finish. We hope to report on HSM experiments using these new capabilities in due course.

The authors gratefully acknowledge the support of the National Science Foundation, through grant CCR–9530741, for this study.

References

[1] EIA Standard RS–274–D (1979), Interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines, Electronic Industries Association, Engineering Dept., Washington, D.C.

[2] G. Albrecht and R. T. Farouki (1996), Construction of C^2 Pythagorean–hodograph interpolating splines by the homotopy method, *Advances in Computational Mathematics* **5**, 417–442.

[3] S. Ashley (1995), High–speed machining goes mainstream, *Mechanical Engineering* **117** (5), 56–61.

[4] G. Birkhoff and C. de Boor (1965), Piecewise polynomial interpolation and approximation, in *Approximation of Functions* (H. L. Garabedian, ed.) Elsevier, Amsterdam, pp. 164–190.

[5] J.–J. Chou and D. C. H. Yang (1991), Command axis generation for three–axis CNC machining, *ASME Journal of Engineering for Industry* **113** (August), 305–310.

[6] —— (1992), On the generation of coordinated motion of five–axis CNC/CMM machines, *ASME Journal of Engineering for Industry* **114** (February), 15–22.

[7] G. Farin (1993), *Curves and Surfaces for Computer Aided Geometric Design* (3rd Edition), Academic Press, Boston.

[8] R. T. Farouki (1992), Pythagorean–hodograph curves in practical use, in *Geometry Processing for Design and Manufacturing* (R. E. Barnhill, ed.), SIAM, Philadelphia, 3–33.

[9] —— (1994), The conformal map $z \rightarrow z^2$ of the hodograph plane, *Computer Aided Geometric Design* **11**, 363–390.

[10] —— (1996), The elastic bending energy of Pythagorean–hodograph curves, *Computer Aided Geometric Design* **13**, 227–241.

[11] —— (1997), Pythagorean–hodograph quintic transition curves of monotone curvature, *Computer Aided Design* **29**, 601–606.

[12] R. T. Farouki, J. Manjunathaiah, and S. Jee (1998), Design of rational cam profiles with Pythagorean–hodograph curves, *Mechanism and Machine Theory*, to appear.

[13] R. T. Farouki, J. Manjunathaiah, D. Nicholas, G.–F. Yuan, and S. Jee (1998), Variable feedrate CNC interpolators for constant material removal rates along Pythagorean–hodograph curves, *Computer Aided Design*, to appear.

[14] R. T. Farouki, J. Manjunathaiah, and G.–F. Yuan (1998), G codes for the specification of Pythagorean–hodograph tool paths and associated feedrate functions on open–architecture CNC machines, *International Journal of Machine Tools and Manufacture*, to appear.

[15] R. T. Farouki and C. A. Neff (1995), Hermite interpolation by Pythagorean–hodograph quintics, *Mathematics of Computation* **64**, 1589–1609.

[16] R. T. Farouki and V. T. Rajan (1987), On the numerical condition of polynomials in Bernstein form, *Computer Aided Geometric Design* **4**, 191–216.

[17] —— (1988), Algorithms for polynomials in Bernstein form, *Computer Aided Geometric Design* **5**, 1–26.

[18] R. T. Farouki and T. Sakkalis (1990), Pythagorean hodographs, *IBM Journal of Research and Development* **34**, 736–752.

[19] —— (1994), Pythagorean–hodograph space curves, *Advances in Computational Mathematics* **2**, 41–66.

[20] R. T. Farouki and S. Shah (1996), Real–time CNC interpolators for Pythagorean–hodograph curves, *Computer Aided Geometric Design* **13**, 583–600.

[21] R. T. Farouki, Y–F. Tsai, and G–F. Yuan (1998), Contour machining of free–form surfaces with real–time PH curve CNC interpolators, *Computer Aided Geometric Design*, to appear.

[22] R. Field and T. Beard (1996), High–speed machining of molds and dies, *Modern Machine Shop* **69** (6), 76–83.

[23] S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi (1983), Optimization by simulated annealing, *Science* **220**, 671–680.

[24] R. Komanduri, K. Subramanian, and B. F. von Turkovich (eds.) (1984), *High Speed Machining*, PED–Vol. 12, ASME, New York.

[25] J. M. Lane and R. F. Riesenfeld (1981), Bounds on a polynomial, *BIT* **21**, 112–117.

[26] R–S. Lin and Y. Koren (1996), Real–time interpolators for multi–axis CNC machine tools, *Manufacturing Systems* **25**, 145–149.

[27] M. Lundy and M. Mees (1986), Convergence of an annealing algorithm, *Mathematical Programming* **34**, 111–124.

[28] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953), Equation of state calculation by fast computing machines, *Journal of Chemical Physics* **21**, 1087–1091.

[29] A. Shima, T. Sasaki, T. Ohtsuki, and Y. Wakinotani (1996), 64–bit RISC–based Series 15 NURBS interpolation, *FANUC Technical Review* **9** (1), 23–28.

[30] M. Shpitalni, Y. Koren, and C. C. Lo (1994), Realtime curve interpolators, *Computer Aided Design* **26**, 832–838.

[31] S. Smith and J. Tlusty (1997), Current trends in high–speed machining, *ASME Journal of Manufacturing Science and Engineering* **119**, 664–666.

[32] J. Tlusty (1993), High–speed machining, *CIRP Annals* **42**, 733–738.

[33] K. Ueda (1998), Pythagorean–hodograph space curves by quaternion calculus, preprint.

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