Computer-Aided Design and Applications

http://www.cadanda.com

Reducing Curvature by Deviating CAM Tool Paths within a Tolerance Band

G. B. Naseath[1] and W. E. Red[2]

[1]Brigham Young University, naseath@gmail.com
[2] Brigham Young University, ered@et.byu.edu

ABSTRACT

The feedrate along a tool path is directly related to the path curvature. High-curvature sections in tool paths are caused by complex part geometry, data noise, and discontinuities in the model. High-curvature sections cause the NC machine to reduce the feedrate along the tool path due to acceleration and jerk limits. Reduced feedrates increase machining time and decrease production rates. This paper presents new methods to decrease curvature on tool paths represented by splined Bezier curves, within a tolerance band, thereby increasing production rates for complex surface machining. In sample cases the algorithm decreased machining times by 1% to 9% for sections of high curvature and by 16% to 75% for CAM induced ripples.

Keywords: tool path planning efficiency, increased feedrate, reduced curvature.
DOI: 10.3722/cadaps.2008.921-931

1. INTRODUCTION

NC machines are used to manufacture complex part geometries because they efficiently and accurately machine parts. Tool paths are now being represented by complex curves such as B-Splines and NURBS because they permit smoother and faster motion, as compared to the conventional approach of passing large blocks of tessellated move increments. High-curvature sections exist in B-Spline tool paths that cause the NC controller to decrease the tool feedrate. High-curvature sections can be created by the designer or result from anomalies in the computer aided design (CAD) model.

When a user plans a production process using a CAM system, a surface tolerance is defined according to the quality specifications. A path tolerance radius is specified that limits the tool path to a tolerance band that is dependent on the surface tolerance. A larger path tolerance allows the path to deviate further and produces a smoother path at the expense of surface accuracy. Deviating the tool path within this tolerance band can smooth high-curvature ripples, resulting in an overall reduced-curvature tool path. As a simple example, see the red line in Fig. 1 with reduced curvature as compared to the blue line, both falling within the black tolerance bands. The new path allows higher feedrates, resulting in reduced machining times and improved profits. The sum of the path tolerance and the machine tool's accuracy must be less than the allowable surface tolerance to ensure that the part will remain within finishing tolerances. The tool path is allowed to touch the edge of the tool path tolerance band.

1.1 Related Research

Several researchers have proposed methods of interpolating data using various types of splines that are at least C^2 continuous and that allow the tool to move smoothly along the path - see [1], [3–6], [8-9], [14]. Unfortunately, undesirable high-curvature oscillations (ripples) may develop in these curves during the interpolation of the tool path. Even though the mathematical representation of the tool path is C^2 continuous, these ripples cause sections of high curvature that slow the desired feedrate.

The main cause for high-curvature ripples is discontinuities in the position, tangency, and curvature of the model part surface. Discontinuities that cause high-curvature sections can be created in a model in many ways. International

TechneGroup Incorporated [7] explains how gaps and overlaps in CAD models caused by accuracy constraints create inefficiencies in NC programming. An inexperienced designer may create a model with tangency and curvature discontinuities. Bohez [2] says that tangency discontinuities in CAD models cause these high-curvature sections.
Even when a B-Spline is interpolated to fit a curvature-continuous surface, small ripples can form because interpolation points represent discrete values along a continuous surface. Path position, tangency, and curvature of the surface will change discretely between interpolation points, creating the same affect as a surface with minute discontinuities. These minute discontinuities create continuity ripples in the tool path. The amplitude of the ripples correlates directly to the severity of the discontinuity.

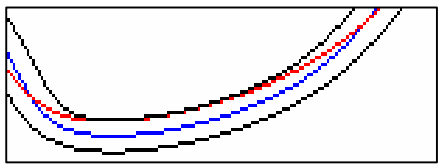

Fig. 1: Curvature smoothing in tolerance band.

Fig. 2: High-curvature toy molds.

Fig. 3: Tire complex geometry.

Not all high-curvature sections are created by anomalies. Many are created by the designer to satisfy a design requirement. Complex parts may contain unnecessarily small radii with high-curvatures. The designer may not realize the effect that these small radii have on machining times. If there is enough surface tolerance on the part, smoothing can decrease the curvature in these high-curvature regions. Examples of parts with unnecessarily high-curvature are the molds for children's toys and tire molds as shown in Fig. 2. and Fig. 3.
To decrease the curvature of splines, researchers effectively filter noisy models ([2], [10], [13]), but none minimize curvature and limit the tool path to a path tolerance band. Eilers [3] minimizes curvature by a penalty on sections with high curvature, but does not consider NC tool paths constrained to a path tolerance.
Langeron [9] limits tool paths within a tolerance tube, by interpolating random points along the desired part surface without regard to curvature. However, Langeron's splines can introduce even more high-curvature sections as the path wiggles back and forth within the tolerance band.
Although significant work has been done to generate optimal tool paths, none smooth paths without unnecessary high-curvature sections, nor do any deviate the tool path within a path tolerance band to minimize curvature.

1.2 Paper Objectives

This paper focuses on the feasibility of smoothing B-Spline tool paths. It smoothes tool paths with high-curvature sections that are created by the designer or by anomalies in the CAD model. The methods are extendable to any degree B-Spline. Only paths for 3-axis end mills are considered, meaning that only the position and not the orientation of the tool is considered. Only 2-D planar paths are created, smoothed, and tested; still, many practical contour machining methods fall within these assumptions.
The methods use C++ algorithms to parse CAM-produced spline data and reduce the curvature in CAM-produced cubic B-Spline tool paths, modifying the splines within a path tolerance. Verification used test data and criteria to ensure that the algorithms improve the tool path, by comparing curvature values of the tool paths before and after the application of the smoothing algorithm. Path time evaluations used S-curve velocity profiles through the high curvature regions to compare the time to move across the original and smoothed paths; see Red [12].
We will first provide some background information on the use of control point perturbations to change B-Spline curvature properties, followed by the algorithmic exposition, and concluding with several test examples and related smoothing data.

2. B-SPLINE CONTROL POINT BACKGROUND

Bezier curves are created using a control polygon made up of control points as in Fig. 4. Dr. Pierre Bezier designed the Bezier curve (red line in Fig. 4) so that it would mimic the shape of its control polygon, by passing through the first and last point in its control polygon, and also tangent to the control polygon at the endpoints.
The equation for the Bezier Curve is similar to the equation for the center of mass of point masses. If there are four equal masses distributed for the cubic curve shown in Fig. 4, then their center of mass is

$$\overline{\mathbf{P}} = \frac{m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3}{m_0 + m_1 + m_2 + m_3} \tag{2.1}$$

Bezier curves are created by using parametric equations to vary the masses of each point instead of using equal constant values. The equations for the masses (the blending functions) in Fig. 4 become

$$m_0(t) = (1-t)^3; \quad m_1(t) = 3t(1-t)^2; \quad m_2(t) = 3t^2(1-t); \quad m_3(t) = t^3 \tag{2.2}$$

As t changes from zero to one, the center of mass also changes. The Bezier curve is the path that the center of mass follows as t changes from zero to one. In a more general form, the blending functions for Bezier curves are referred to as Bernstein polynomials and represented by the following equations for an n degree Bezier curve with n+1 control points:

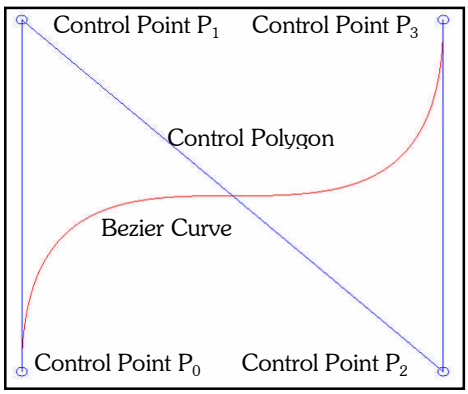

Fig. 4: Bezier curve (red), control polygon.

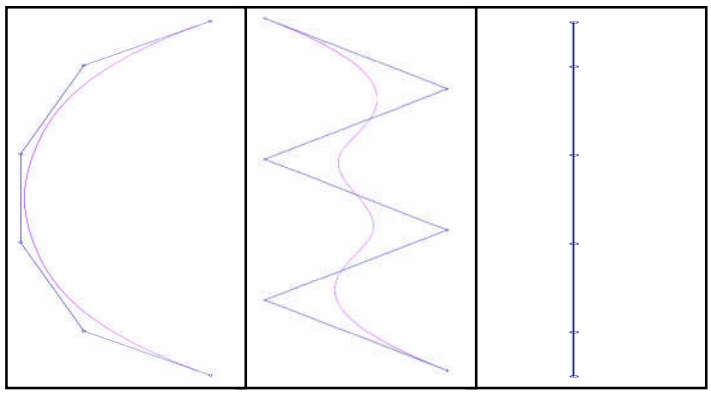

Fig. 5: B-Splines follow the control polygon shapes.

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \qquad \text{where } \binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{2.3}$$

Equation (2.3) leads to the general equation for a Bezier curve:

$$\mathbf{P}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{P}_i = \sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^i\mathbf{P}_i \tag{2.4}$$

A B-Spline is formed by splining multiple Bezier curves together. B-Splines connect Bezier curves with C^{n-1} continuity. This means that a cubic B-Spline will be C^2 continuous where the Bezier curves meet. The B-Spline uses a control polygon like the Bezier curve, but it also has a knot vector. The knot vector is a list of parameter values, or knots, used to determine the parameter values at which a Bezier curve begins and ends. Like Bezier curves, B-Splines follow the general shape of their control polygon. If the control polygon is shaped like a semicircle, a saw-tooth, or a flat straight line, the B-Spline will be shaped similarly, Fig. 5.

2.1 Distance Between Two Splines

The distance between points of equal parameter value on two splines can be represented as a new spline with a control polygon calculated as the difference between the two splines' control polygons. The two splines must have an equal number of control points and parameterization to use this distance formula. Equation (2.5) is the spline that represents the distance between the two splines:

$$\mathbf{D}(t) = \mathbf{P}(t) - \mathbf{Q}(t) = \sum_{i=0}^{n}(\mathbf{P}_i - \mathbf{Q}_i)B_i^n(t) = \sum_{i=0}^{n}\mathbf{D}_iB_i^n(t), \tag{2.5}$$

where $\mathbf{P}(t) = \sum_{i=0}^{n}\mathbf{P}_iB_i^n(t)$ and $\mathbf{Q}(t) = \sum_{i=0}^{n}\mathbf{Q}_iB_i^n(t)$.

This equation compares the difference between the original and perturbed spline to the tolerance band and is used in the control point perturbation algorithms of this paper. Note that this equation will work for both Bezier curves and B-Splines. Fig. 6 shows a B-Spline that is the difference of two other B-Splines.

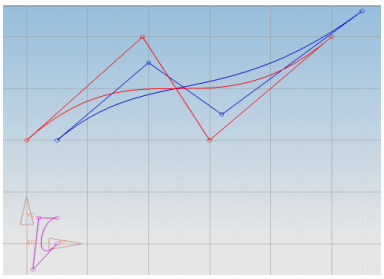

Fig. 6: B-Spline difference is purple line (lower left figure).

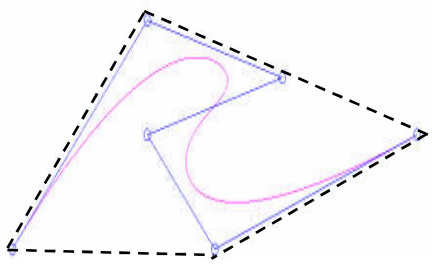

Fig. 7: Convex hull property (dashed line).

2.2 Convex Hull Property

Both Bezier curves and B-Splines remain within the convex hull of their control points; see Fig. 7. The polygon formed is the convex hull. The center of mass analogy for Bezier curves and B-Splines ensures that the spline will always remain within the convex hull. All of the control points are either inside of or on the boundary of the convex hull so it is impossible for the center of mass (the spline) to lie outside of the convex hull.

The convex hull property guarantees that the distance between two curves is bounded by the largest distance from the origin to any of the control points of the difference B-Spline. This also means that if a spline's control points are moved, then the spline is guaranteed to not move a distance greater than the largest distance that any control point moved. This is important to the control point perturbations used in our algorithm.

2.3 Continuity

A B-Spline of degree *n* is guaranteed to join Bezier curves with C^{n-1} continuity. Two curves are C^k continuous if

$$\mathbf{P}(t_1) = \mathbf{Q}(t_1), \mathbf{P}'(t_1) = \mathbf{Q}'(t_1), \ldots, \mathbf{P}^{(k)}(t_1) = \mathbf{Q}^{(k)}(t_1) \tag{2.6}$$

Parametric continuity C^k depends on the parameterization of the B-Spline (t), but geometric continuity (G^k) does not. If a spline is C^k continuous then it is also G^k continuous. G^0 means that the two curves have a common endpoint but not necessarily the same parameter value. Tangency continuity or G^1 continuity is when the control polygons are tangent where they meet. Curvature continuity or G^2 continuity is when the curvature is equal for both curves where they meet. This is considered sufficiently smooth for most tool paths.

3. SMOOTHING ALGORITHM

This section will present a smoothing algorithm that uses control point perturbations to decrease spline curvature within a tolerance tube. First note that in high-curvature sections, a NC machine reduces the feedrate to remain within acceleration and jerk limits. Total acceleration for motion in Cartesian space is the vector sum of the tangential and centripetal acceleration:

$$\mathbf{A}_{\text{total}} = \mathbf{A}_{\text{centripital}} + \mathbf{A}_{\text{tangential}} \tag{3.1}$$

where $A_{centripetal} = kV^2_{feedrate}$ and k is the path curvature. High curvature increases centripetal acceleration. As centripetal acceleration increases with high curvature, the allowable tangential acceleration must decrease.

The smoothing algorithm in this thesis reduces the curvature of all ripples even if the curvature is below the value that would reduce the feedrate, because it takes longer to calculate the curvature to distinguish between negligible and significant curvature values than it does to smooth the path. The algorithm thus reduces the curvature in tool paths within a tolerance band without calculating the curvature.

3.1 Path Tolerance

The path tolerance band is formed by offsetting the given tool path in two directions by the path tolerance radius $\pm\rho$. The offset band is formed by the set of all points (P_b) that lay a perpendicular distance ρ from a given curve point P, as shown in (3.2), where x' and y' represent the derivatives of the spline at P with respect to the spline parameter t, Fig. 8.

$$P_b(\rho, x(t), y(t)) = P(x(t), y(t)) \pm \rho \frac{(y'(t), -x'(t))}{\sqrt{x'^2(t) + y'^2(t)}} \tag{3.2}$$

3.2 Algorithmic Basis

When all of the control points of a B-spline form a single straight line, the spline becomes a straight line with zero curvature. Thus moving the control points towards a straight line, considering the tolerance tube limit, will reduce curvature. This observation forms the basis of the smoothing algorithm. The algorithm incrementally perturbs the various control points to flatten the control point polygon, comparing the perturbed spline to the offset splines, exiting the iterations when the perturbed spline is sufficiently close to the tolerance band. During each iteration: 1) the control points are re-categorized; 2) the control points are moved in new directions; 3) the minimum distance between the spline and tolerance is calculated; and 4) all of the control point move distances are rescaled.

To test the curvature improvement we must determine the curvature, but the algorithm has the advantage of not requiring curvature calculations. Equation (3.3) is the curvature calculation for point P_0 in Fig. 9 formed by splitting the Bezier at the point of curvature interest. The Bezier curve containing the point where the curvature value is to be calculated is extracted from the B-Spline. Then the Bezier is divided at the point of interest. This creates an endpoint where the curvature is to be calculated. The first three points of the Bezier control polygon are shown in Figure 9. P_0 is the point where the curvature is being calculated and the first point in the Bezier control polygon. The value h is the distance from P_2 to the line formed by P_0 and P_1. If two line segments are flattened, h is decreased and the curvature k is also decreased.

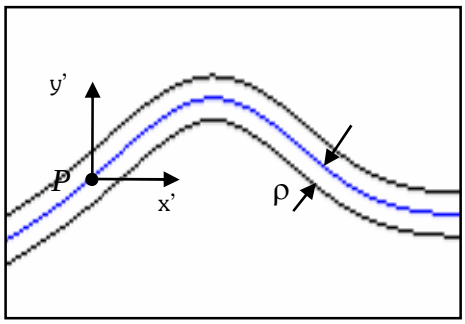

Fig. 8: Path tolerance tube.

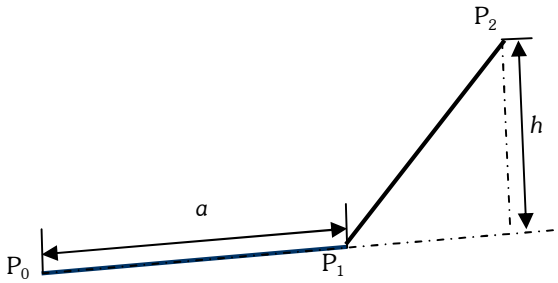

Fig. 9: Curvature from control point polygon.

$$k = \frac{n-1}{n}\frac{h}{a^2} \tag{3.3}$$

where:

n = degree

$$a = \left|\mathbf{P}_1 - \mathbf{P}_0\right|$$

$$h = \frac{\left(Y_{P_0} - Y_{P_1}\right) * X_{P_2} + \left(X_{P_1} - X_{P_0}\right) * Y_{P_2} + \left(X_{P_0} * Y_{P_1} - X_{P_1} * X_{P_0}\right)}{\sqrt{\left(Y_{P_0} - Y_{P_1}\right)^2 + \left(X_{P_1} - X_{P_0}\right)^2}}$$

3.3 Direction Vectors for Control Point Perturbation

To reduce the curvature, first, we determine a direction vector for each control point according to its geometry and category. The direction vector is the direction that the control point will move to form a straight line with its neighboring control points. Control points are placed in one of three categories, depending on how they must be perturbed to form a straight line. The first category contains the *endpoints* of the control polygon. The next category consists of *ripple points*. These points are recognized because the control polygon forms a saw-tooth pattern, i.e., each point in the control polygon lies in the opposite direction, Fig. 5. The direction vector that smoothes the ripple point is the normalized bisector of the triangle formed by the ripple point and the two adjacent control points in the control polygon. *Smooth control points* make up the final category. These are recognized because the control polygon continues in the same direction along an arc at these points and the spline is smooth.

All adjacent smooth control points are placed in groups that contain a beginning point, one or several middle points, one or two apex points, and an ending point. The apex point(s) are the median point(s) in each group. If there are an odd number of points in a group then there is only one apex point. If there is an even number of points then there are

two apex points. The rest of the points in the group are middle points. For example, in Fig. 10 points 4 and 9 are the beginning and ending points of a smooth section, respectively. Points 5 and 8 are middle points and points 6 and 7 are the apex points.

The algorithm begins by determining perturbation directions based on the control point properties described in the following sections. All control points are initially perturbed by the tolerance ρ to establish control point perturbation sensitivity. The movement of the path spline is then measured, establishing a non-uniform set of control point sensitivity perturbations. The sensitivity value for each control point is the ratio of the spline move distance to the control point's move distance from the previous iteration. The maximum allowable move distance is the minimum distance between the spline and the tolerance tube remaining after the last iteration. This step is repeated iteratively to reduce path curvature until the perturbed spline approaches sufficiently close to the tolerance band. See reference [11] for more detail.

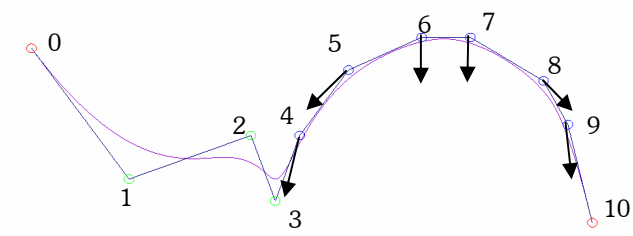

Fig. 10: Curvature from control point polygon.

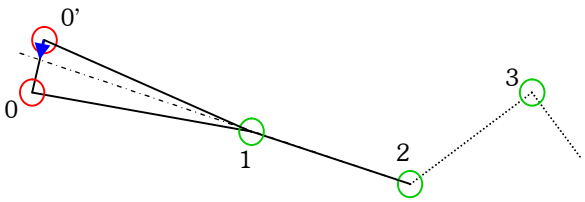

Fig. 11: Endpoint 0 perturbed to 0', requiring direction vector (blue) reversal.

3.3.1 Endpoint Perturbations

Each endpoint has a direction vector that is perpendicular to both its control polygon line segment and the tolerance tube. The control polygon is initially parallel to the tolerance tube at both ends. Each endpoint is smoothed by perturbing it towards the line formed by the next two or previous two control points in the control polygon in Fig. 11 and in this direction:

$$\mathbf{DV}'_0 = \begin{bmatrix} \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{-\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \tag{3.4}$$

where $\Delta X = X_1 - X_0$ and $\Delta Y = Y_1 - Y_0$. The actual perturbation distance depends on the sensitivity determined in each perturbation.

The direction vectors for endpoints are perpendicular to the initial control polygon. As the control polygon moves with each perturbation, it will no longer be parallel to the tolerance tube at the ends. However, the endpoint direction vector is always forced to be perpendicular to the tolerance tube so that the endpoint remains at the end of the tolerance tube to force tool path continuity. It is possible to overshoot the target line in a given iteration which may require a reversing of the direction vector to again point towards the line. If the minimum distance to the line is negative, the direction vector is reversed. If the distance to the line is zero the direction vector is set to null. The direction vector for the last control point is calculated in a similar manner to the first point.

3.3.2 Ripple points

The direction vector for a ripple control point directs the control point towards a straight line formed by its two adjacent control points. The procedure to find the red normalized direction vector for ripple control points, Fig. 12, uses the green normalized direction vectors from point 2 to point 1 and 3, respectively. The generalized equations are:

$$\mathbf{A}_i = \begin{bmatrix} X \\ Y \end{bmatrix}_i = \begin{bmatrix} \dfrac{\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix}; \qquad \mathbf{B}_i = \begin{bmatrix} X \\ Y \end{bmatrix}_i = \begin{bmatrix} \dfrac{\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \tag{3.5}$$

where $\Delta X = X_{i-1} - X_i$ and $\Delta Y = Y_{i-1} - Y_i$ define $\mathbf{A}_i$, and $\Delta X = X_{i+1} - X_i$ and $\Delta Y = Y_{i+1} - Y_i$ define $\mathbf{B}_i$. These define the ripple perturbation vector $\mathbf{DV}_i = \|\mathbf{A}_i + \mathbf{B}_i\|$.

3.3.3 Smooth Points

The procedure to calculate the direction vectors for smooth sections combines multiple vectors as shown in Fig. 13. Direction **A** vectors for the apex points are calculated first. Then preliminary direction **B** vectors are calculated for the middle and ending points. Each preliminary vector is then combined with a scaled version of the apex direction vectors (**C** vectors, black) to create the non-apex final direction vectors **A**.

The apex direction vectors are calculated in one of two ways. If a single apex, the direction vector is calculated using the same method used for a ripple point's direction vector in (3.5). If a double apex, each apex is assigned the same direction vector that is calculated using Equation (3.6). It is perpendicular to the line that goes through both apex points (points 6 and 7) as illustrated by line D in Fig. 13.

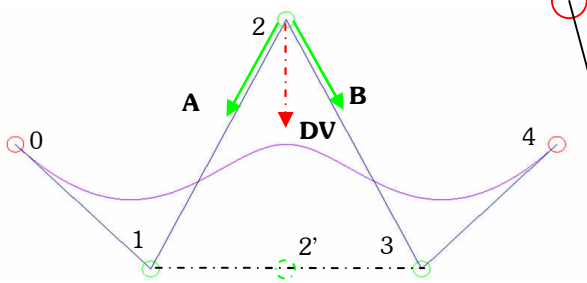

Fig. 12: Direction vector (red) for ripple point defined as normalized sum of vectors **A** and **B.**

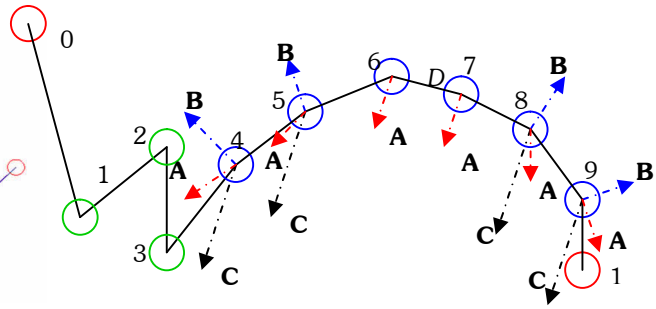

Fig. 13: Smooth section control point perturbations.

$$\mathbf{DV}'_i = \begin{bmatrix} X \\ Y \end{bmatrix}_{m-1} = \begin{bmatrix} \dfrac{-\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \tag{3.6}$$

where $\Delta X = X_{i+1} - X_i$ and $\Delta Y = Y_{i+1} - Y_i$. Note that (3.6) does not guarantee that the direction of the apex vectors is correct. To determine if the apex direction vectors must be reversed, the minimum distance from the control point immediately following the second apex point to the line through the two apex points is calculated. If the distance to the line is negative, the preliminary direction vector is reversed.

Preliminary vectors B are calculated the same way as ripple points in (3.5) except that they are reversed to point outwards as shown in Fig. 13. The preliminary direction B vectors move the control points out, widen the path, and lower the curvature.

A scaled vector C is created at each non-apex point by scaling the apex direction vector by some factor α using Equation (3.7) Scale values of 9 – 10 produced the best combination of widening (vector B) and flattening (vector C). Combining vectors B and C with the given weights was found to best reduce the curvature in smooth sections in the most cases. Vectors A align the movement of the non-apex points with the apex points, so that the entire curve moves together as it flattens. The normalized sum of vector B and vector C in Equation (3.8) is the direction vector for each of the smooth points.

$$\mathbf{C}_i = \alpha \mathbf{A}_i^{apex} \tag{3.7}$$

$$\mathbf{A}_i^{non-apex} = \left\| \mathbf{B}_i + \mathbf{C}_i \right\| \tag{3.8}$$

By summing the two vectors together and normalizing the result, the final direction vector obtains attributes from both of the vectors. The curve is widened by vector B and flattened by vector C. The final direction vectors for the control points in a smooth section cause the curve to move to the inside of the tolerance tube at the apex and to the outside of the tolerance tube at the beginning and ending points as shown in Fig. 14.

4. RESULTS

Five sample tool paths shown in Fig. 15 (not to scale) were created to test the smoothing algorithm. The tool paths represent a single pass across a surface. Each tool path is composed of a single B-spline and contains a different set of high-curvature ripples.

The Case 1 tool path represents a tool path with low curvature radii (maximum curvature is 0.175 mm^{-1}). The tool feedrate will approach the commanded value. The tool path of Case 2 is mostly low curvature with two high-curvature sections. The path is 376 mm long with a maximum curvature of 9.0 mm^{-1}, where the tool feedrate will slow significantly. The tool path for Case 3 is complex with multiple high-curvature ripples. The tested path is 210 mm long and has a maximum curvature of 28 mm^{-1}. This path represents an extreme case in which the feedrate will be very slow. The tool path shown in Case 4 is generated for factory siping on a snow tire mold with complex geometry. The final case (Case 5) was designed as a straight line with zero curvature but two of the data points used to interpolate the tool path were designed as noisy discontinuity bumps. These noisy points create ripples in the tool path that significantly increase the curvature and cause the spline to deviate from the desired tool path by up to 0.4 mm.

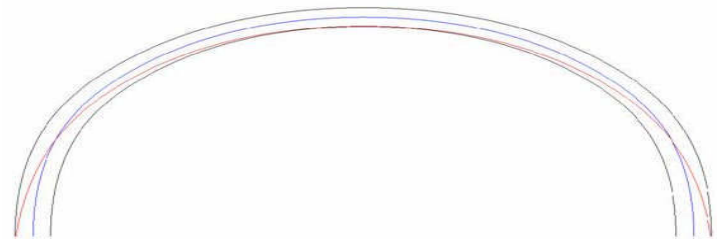

Fig. 14: Smooth section widened and flattened to reduce curvature.

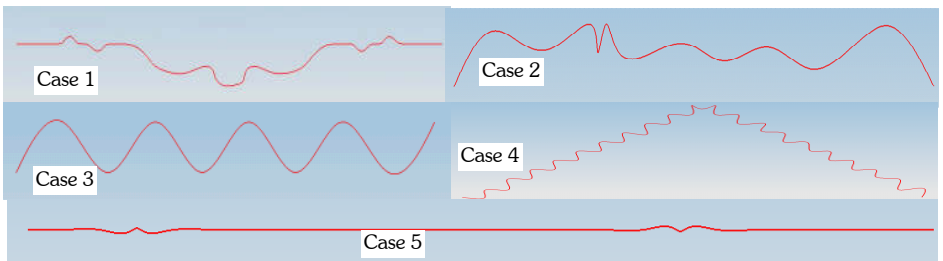

Fig. 15: Test cases.

4.1 Smoothing Parameters

The accuracy of all tool paths was set to 1%. Therefore, the iterations terminated when the tool path closed to within 1% of the path tolerance. The precision, defined as the number of points checked on each knot interval of the B-spline per iteration, was set to 5 causing each Bezier in the B-spline to be split into 5 sections. Each of the tool paths was smoothed using three different path tolerances: 0.025 mm, 0.125 mm, and 0.25 mm.

4.2 Smoothing Results: Path Shape

Fig. 16 shows several results (not to scale) for a tolerance band of 0.25 mm with the more critical improvements expanded for visualization purposes. The results are identified for several of the selected cases shown in Fig. 15, where the red line represents the curvature flattened spline. Note how the red line approaches the tolerance band as you would expect based on the curvature properties, thereby reducing the curvature. Naseath [11] also shows that his smoothing algorithms essentially eliminate the curvature noise of Case 5. The results are similar for the tighter tolerances of 0.025 mm and 0.125 mm.

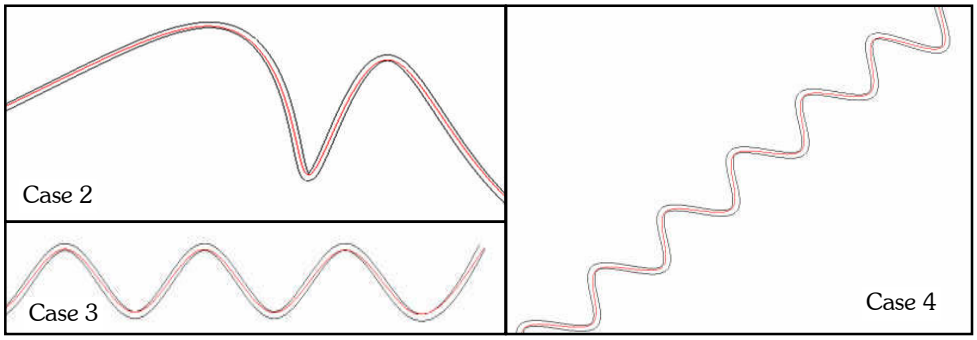

Fig. 16: Smoothing test results.

4.3 Smoothing Results: Curvature

Tab. 1 shows average curvature improvements over the splined paths in Cases 1 – 5, from small % improvements ranging to 9.5%, depending on case and path tolerance. One case of interest is the improvement of 10% to 94% for Case 5 noisy paths.

Part	*Path Tol (mm)*	*Curvature Avg (1/mm)*	*Curvature Diff (1/mm)*	*% Improvement*
Test Case 1: Simple	Original	0.040314	0	0.00%
	R = 0.025	0.040154	0.0001592	0.39%
	R = 0.125	0.039588	0.0007257	1.80%
	R = 0.250	0.038976	0.001338	3.32%
Test Case 2: Medium	Original	0.14091	0	0.00%
	R = 0.025	0.140219	0.0006906	0.49%
	R = 0.125	0.137186	0.0037242	2.64%
	R = 0.250	0.133279	0.0076309	5.42%
Test Case 3: Complex	Original	0.564505	0	0.00%
	R = 0.025	0.557309	0.0071956	1.27%
	R = 0.125	0.53213	0.0323753	5.74%
	R = 0.250	0.510896	0.0536084	9.50%
Test Case 4: Snow Tire	Original	0.296756	0	0.00%
	R = 0.025	0.295998	0.0007578	0.26%
	R = 0.125	0.292879	0.0038774	1.31%
	R = 0.250	0.288374	0.0083822	2.82%
Test Case 5: Line	Original	0.093916	0	0.00%
	R = 0.025	0.08424	0.0096766	10.30%
	R = 0.125	0.04706	0.0468563	49.89%
	R = 0.250	0.005922	0.087994	93.69%

Tab: 1: Relative curvature improvement after smoothing algorithm.

4.4 Smoothing Results: Feedrate and Time

The time to machine each of the tool paths was simulated using a trajectory generator. The trajectory generator uses S-curve velocity profiles that are limited by the curvature of the tool path. The maximum values for the trajectory generator were set to: feedrate = 400 mm/s; acceleration = 4000 mm/s^2; jerk = 20,000 mm/s^3.

An initial velocity profile was determined by assuming that the velocity was only limited by the curvature. This profile contained the maximum allowable velocities since it assumed that the velocities could change instantly between positions along the tool path and were not limited by tangential acceleration. The maximum and minimum velocities correspond to specific positions along the tool path.

All of the velocities that were not local maximum or minimum values were removed. S-curves were created to accelerate between the maximum and minimum velocities. The S-curves were required to traverse the distances between the positions on the tool path that correspond to the maximum and minimum velocities. Local maximum velocity values were decreased until the robot was capable of accelerating between the minimum and maximum velocities and traversing the required distance. More detail can be found in [11].

Tab. 2 shows feedrate improvement up to 5%, depending on tolerance, except for the case of noisy data where feedrates are improved tremendously. Of course, the improvement will be most appreciable when machining complex surfaces, where improvements of 5 – 10% can represent large manufacturing savings. Naseath [11] shows that the % time improvements are actually larger by a few % over these numbers because feedrate improvements reduce the time spent in acceleration periods.

Part	*Path Tol (mm)*	*Feedrate Avg (mm/s)*	*Feedrate Diff (mm/s)*	*% Feedrate Improvement*
Test Case 1: Simple	Original	195.0	0.0	0.00%
	R = 0.025	195.1	0.1	0.07%
	R = 0.125	195.4	0.4	0.22%
	R = 0.250	195.4	0.4	0.21%
Test Case 2: Medium	Original	180.8	0.0	0.00%
	R = 0.025	180.9	0.1	0.03%
	R = 0.125	181.1	0.3	0.14%
	R = 0.250	186.2	5.4	3.00%
Test Case 3: Complex	Original	52.3	0.0	0.00%
	R = 0.025	52.4	0.1	0.26%
	R = 0.125	53.8	1.5	2.93%
	R = 0.250	54.8	2.5	4.84%
Test Case 4: Snow Tire	Original	71.4	0.0	0.00%
	R = 0.025	72.6	1.2	1.63%
	R = 0.125	72.9	1.4	2.01%
	R = 0.250	73.0	1.5	2.17%
Test Case 5: Line	Original	73.1	0.0	0.00%
	R = 0.025	74.9	1.8	2.45%
	R = 0.125	91.9	18.8	25.74%
	R = 0.250	299.5	226.4	309.58%

Tab. 2: Feedrate measurements from before and after smoothing.

Naseath [11] also lists CPU calculation times for the cases in Tab. 2, which in most cases are less than 1 CPU second (none greater than 2 seconds). These were calculated using a Pentium Core 2 Duo with two 2.13 GHz processors and 1.98 GB of RAM. Note that these calculation times are for one tool path pass and not for an entire part. The CPU calculation times increase with the number of control points.

5. SUMMARY

Tool paths that are designed with high levels of curvature can be smoothed with the smoothing algorithms presented in this paper. In test cases both curvature and machining times were decreased by 1% to 9% based on the degree of curvature and the tolerance. Resulting savings can be significant on highly complex parts. Tool paths with high-curvature ripples caused by discontinuities can be substantially smoothed using the algorithm, as can design features with little or no filleting between intersecting edges. In Case 5 Naseath shows that machining times can be decreased by 16% to 75%.

Anomalies should be removed before creating tool paths. But if they are not, the smoothing algorithm will diminish their negative affects on machining times. High-curvature ripples in CAD models caused by the designer, bad data, or discontinuities can cause unnecessary increases in machining times. These high-curvature ripples can be smoothed with these algorithms, decreasing machining times.

The algorithm has been simplified by fundamental observations that relate control point perturbations to spline perturbations, and by applying procedures that do not require the computation of spline curvature. These methods, though iterative, are simple to apply, computationally efficient, but are not real-time.

6. REFERENCES

[1] Berglund, T.: Path-planning with obstacle-avoiding minimum curvature variation, B-splines, Licentiate Thesis, Department of Computer Science and Electrical Engineering, Luleå University of Technology, Sweden, 2003.

[2] Bohez, E. L. J.: Compensating for systematic errors in 5-Axis NC machining, Computer-Aided Design, 34 2002, 391-403.

[3] Eilers, P. H. C.; Brian D. M.: Flexible smoothing with B-Splines and penalties, Statistical Science 11, 2 1996, 89-121.

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

[5] Fleisig, R. V.; Spence, A. D.: A constant feed and reduced angular acceleration interpolation algorithm for multi-axis machining, Computer-Aided Design, 33, 2001, 1-15.

[6] Geraerts, R.; Overmars, M. H.: The corridor map method: real-time high-quality path planning, IEEE International Conference on Robotics and Automation, April 2007, 1023-1028.

[7] International TecheGroup Incorporated, “CAD Model Quality", September 2003.

[8] Jung, S.; Jang, E. S.: Collision avoidance of a mobile robot using intelligent hybrid force control technique, IEEE International Conference on Robotics and Automation, April 2005, 4418-4423.

[9] Langeron, J. M.; Duc, E.; Lartigue, C.; Bourdet, P.: A new format for 5-axis tool path computation using B-spline curves, Computer-Aided Design, 36, 2004, 1219-1229.

[10] Lee, C.-K.; Haralick, R. M.; Deguchi, K.: Estimation of curvature from sampled noisy data, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 1993, 536-541.

[11] Naseath, G. B.: Reducing curvature in complex tool paths by deviating from cam-produced tool paths within a tolerance band, M. S. Thesis, Department of Mechanical Engineering, Brigham Young University, December, 2007.

[12] Red, W. E.: A dynamic optimal trajectory generator for Cartesian path following, Robotica, 18, 2000, 451-458.

[13] Tang, C.-K.; Medioni, G.: Robust estimation of curvature information from noisy 3-D data for shape description, IEEE International Conference on Computer Vision, September 1999.

[14] Wang, F. C.; Yang, D. C. H.: Nearly arc-length parameterized quintic spline interpolation for precision machining, Computer-Aided Design, 25, 1993, 281-288.