2007-12-12

Reducing Curvature in Complex Tool Paths by Deviating from CAM-Produced Tool Paths Within a Tolerance Band

George Benjamin Naseath
Brigham Young University - Provo

REDUCING CURVATURE IN COMPLEX TOOL PATHS BY DEVIATING FROM CAM-PRODUCED TOOL PATHS WITHIN A TOLERANCE BAND

by

G. Benjamin Naseath

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

December 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

G. Benjamin Naseath

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	W. Edward Red, Chair
Date	Timothy W. McLain
Date	C. Gregory Jensen

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of G. Benjamin Naseath in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	W. Edward Red Chair, Graduate Committee

Accepted for the Department

Matthew R. Jones
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of Engineering and Technology

ABSTRACT

REDUCING CURVATURE IN COMPLEX TOOL PATHS BY DEVIATING FROM CAM-PRODUCED TOOL PATHS WITHIN A TOLERANCE BAND

G. Benjamin Naseath

Department of Mechanical Engineering

Master of Science

This thesis develops an algorithm to decrease high-curvature sections in tool paths for complex parts to achieve shorter machining times resulting in higher production rates. In the research sample cases, the algorithm decreased machining times by 1% to 9% for design-induced sections of high curvature and by 16% to 75% for CAM induced ripples using high path tolerances. High-curvature sections in tool paths are caused by complex part geometry, noise, and discontinuities in the model. The curvature is decreased by deviating the tool path within an allowable path tolerance.

The feedrate along the tool path is directly related to the curvature of the tool path. High-curvature sections cause the NC machine to reduce the feedrate along the tool path due to acceleration and jerk limits. These lower feedrates increase machining time and slow production rates. This new algorithm decreases curvature, which increases feedrates and decreases machining times, thereby increasing production rates for manufacturing companies.

The tool paths are represented by cubic B-splines. The algorithm is based on the basic principle that the curvature of a B-spline directly relates to the geometry of its control polygon. If the control polygon's geometry has many tight corners then the B-spline will have high curvature. If the control polygon's geometry is a straight line then the B-spline will be a straight line with zero curvature. The algorithm deviates the control polygon's points so that they move towards forming a straight line. The control polygon will rarely form a straight line because the spline is limited by the path tolerance. However, as the control polygon moves towards forming a straight line, the curvature decreases, which allows the feedrate to increase.

Six sample cases are explored in which the machining time is decreased. Three of the cases are tool paths that contain curvature sections with a range of unnecessary curvature from low to high. One sample is the tool path for the complex geometry in a snow tire mold. Another sample tool path contains ripples caused by noise in the CAD model. The last tool path contains ripples caused by tangency discontinuities in the CAD model. The percent of time saved directly relates to the severity of the curvature in the part.

This thesis provides a quick and efficient means to reduce curvature in complex parts, resulting in decreased machining times and increased production rates.

ACKNOWLEDGMENTS

I would like to thank my wife, Maritza Naseath, for all of her support. I would like to thank my advisor, Dr. Red, for guiding me on this journey. I would like to thank my father, Brent Naseath, for being my friend and advisor and for his endless hours of editing. I would like to thank my mother, Vicki Woodfield, for supporting and guiding me as a youth so that I could make it here today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

This thesis decreases machining times on numerically controlled (NC) machined parts, resulting in increased profits for manufacturing companies. This was accomplished through developing an algorithm that deviates cubic B-spline tool paths produced by computer-aided manufacturing (CAM) within a path tolerance to reduce tool path curvature in high-curvature sections.

Manufacturers constantly look for ways to cost-effectively decrease product manufacturing times to increase profits. One of the best ways to reduce manufacturing times is to lean a process by eliminating unnecessary steps and improving the efficiency of required steps.

NC machines are increasingly used to manufacture complex part geometries because they efficiently and accurately machine parts. High-curvature sections exist in B-spline tool paths that cause the NC to decrease the tool feedrate to maneuver these high-curvature sections. High-curvature sections can be created by the designer or result from anomalies in the computer aided design (CAD) model. Slow feedrates increase machining times, which decrease profits.

During part design 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 the high-curvature ripples resulting in an overall reduced-curvature tool path. The new path allows higher feedrates resulting in reduced machining times and improved profits. A tool path that has been smoothed within a tolerance band is shown in Figure 1-1.

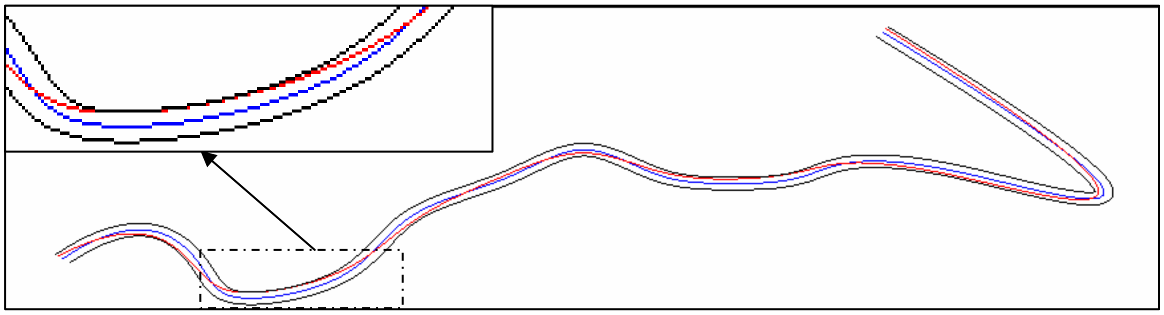

Figure 1-1 Tool path before (blue) and after smoothing (red) that is limited to path tolerance (black)

The sum of the path tolerance and the machine tool's accuracy must be less than the part's surface tolerance to ensure that the part will remain within tolerance. The tool path is allowed to touch the edge of the tool path tolerance band. Any inaccuracies in the machine tool will stack on top of the path tolerance. The path tolerance equation is shown in Equation (1-1).

$$Tolerance_{path} \leq Tolerance_{surface} - Accuracy_{machine} \tag{1-1}$$

1.1 Objectives

The objectives of this thesis are:

- Developed an algorithm that reduces the curvature in CAM-produced cubic B-spline tool paths to reduce machining time. The algorithm modifies the splines within a path tolerance. The algorithm quickly and efficiently performs all calculations.
- Wrote a C++ program that parses CAM-produced spline data, modifies the data according to the above algorithm, and outputs the modified spline data.
- Developed test data and criteria to ensure that the algorithm improved the tool path.
    - Compared the curvature values of the tool paths from before and after the application of the smoothing algorithm to ensure that the curvature decreased.
    - Used a trajectory generator that incorporates S-curve velocity profiles to test the tool paths before and after the application of the algorithm to measure the reduction in machining time.

1.2 Scope

This research 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. It does not produce commercial software to smooth tool paths nor does it upgrade a specific CAM or NC software. The research does not consider calculations before the creation of the initial spline and all of the required data for the initial spline's position is generated using current CAM software. Only cubic B-splines with Bezier end conditions are used to represent the tool paths. The methods discussed are extendable to any degree B-spline. Only paths for 3-axis end mills will are considered, meaning that only the position and not the orientation of the tool is considered. Only 2D planar paths are created, smoothed, and tested. Extending the methods in this thesis to 3D tool paths is discussed in theory only and was not tested. The tool paths are open-ended and not closed. Closed tool paths are discussed theoretically only. Tool paths are tested for a single pass without repeating similar tool paths for multiple passes. Nothing beyond the creation of the reduced curvature B-spline is included. Hence, reparameterizing the curve and calculating the inverse kinematics is not be done.

2 Background

After a part is fully designed and drawings are created using a Computer-Aided Drafting (CAD) system, a process plan is created using a Computer-Aided Manufacturing (CAM) program. The CAM program determines the tool paths that the machine needs to follow in order to machine the finished part. It creates mathematical representations of tool paths and sends them to a numerical control (NC) machine that commands the machine tool to follow the paths.

2.1 Tessellated Tool Paths

The simplest way to represent a complex path is to tessellate the path into small line chordal segments. Figure 2-1 shows a simple tool path tessellated with line chordal segments. By using a large number of line segments, the CAM system is able to represent the tool path with reasonable accuracy. However, a major problem with these linear paths is that they contain tangency and curvature discontinuities, which cause the motion planning algorithms of the machine's controller to slow or even stop the tool feedrate. Tangency (C^1) and curvature (C^2) continuity are reviewed in Section 2.3.5. The machine tool must slow down, as it cannot instantaneously change direction at each line intersection.

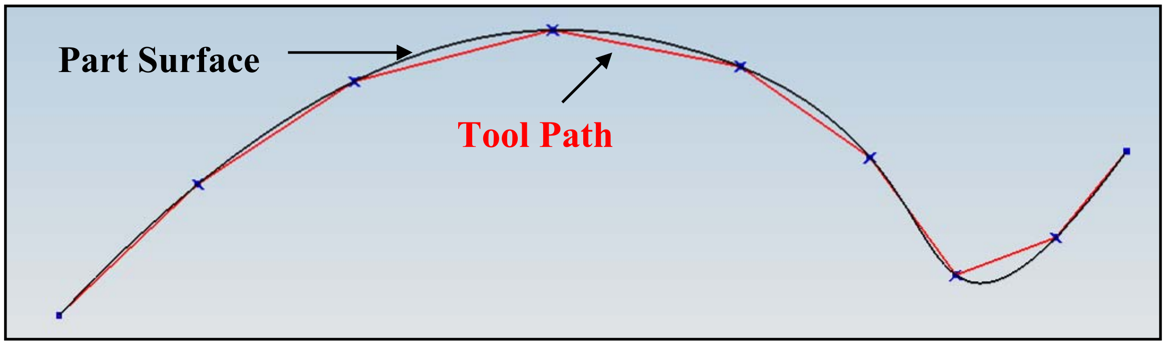

Figure 2-1 Tool path (black) that is represented by line chordal segments (red). To increase accuracy, spline can be tessellated into more line segments.

Jacobs, Sellen, and Wilfong (1989; 1995; 1989) use blending arcs to eliminate the tangency discontinuities and smooth the motion in a tessellated tool path. Figure 2-2 illustrates two line segments blended together with a circular arc. Even though these algorithms are successful in creating short, piecewise, C^1 continuous paths, they do not satisfy the requirements of this thesis. They still contain C^2 discontinuities that cause unnecessary decelerations of the tool. These decelerations are unacceptable because they unnecessarily increase the part's machining time.

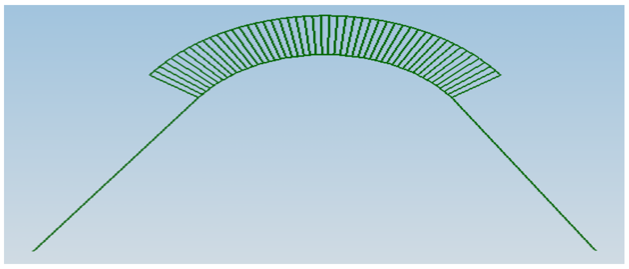

Figure 2-2 Intersection of two line chordal segments blended with arc. Removes tangency discontinuities.

By using the higher order functions cubic spirals and clothoids instead of circles to connect the lines, Kanayama and Scheuer (1997; 1997) eliminate the C^2 discontinuities in the tool paths. See Figure 2-3. The machine tool is able to remain at higher speeds, due to the C^2 continuity in the path.

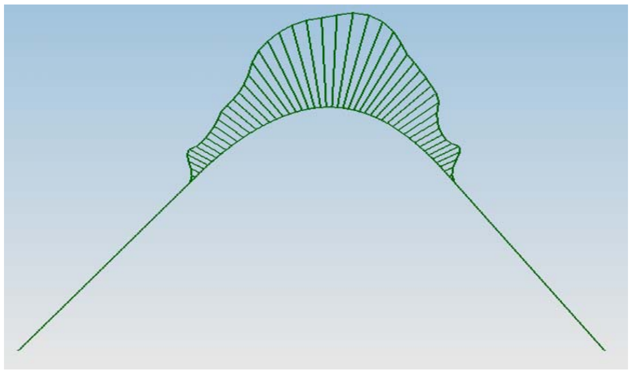

Figure 2-3 Intersection of two line chordal segments blended together with clothoid. Removes curvature discontinuities

The amount of data needed for these piecewise functions increases computational machining time. To produce the smoothest paths using the minimum data, this thesis will use cubic B-splines, which require relatively few control points. Figure 2-4 shows a simple tool path that is represented using a cubic B-spline. The spline and the tool path are practically identical.

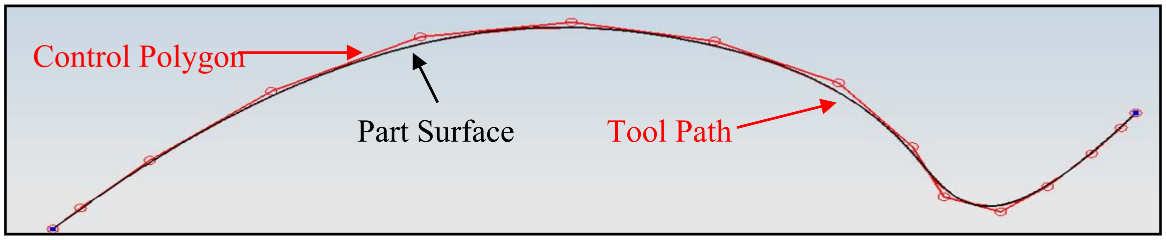

Figure 2-4 Ideal tool path (black) that is represented using cubic B-spline (red) and its control polygon (also red). Control polygon for Spline is all that can be seen. B-spline is almost perfectly lined up with ideal tool path and is difficult to see.

2.2 Spline Tool Paths

Researchers are exploring spline use for tool paths to reduce the amount of data necessary and guarantee C^{n-1} continuity, where n is the degree of the spline. Several researchers (Berglund, 2003; Eilers, 1996; Erkorkmaz, 2001; Fleisig, 2001; Geraerts, 2007; Jung, 2005; Langeron, 2004; Wang, 1993) have proposed methods of interpolating data using various types of splines that are all at least C^2 continuous and that allow the tool to move smoothly along the path. However, they all possess a critical flaw, which this thesis intends to minimize. Undesirable high-curvature oscillations (ripples) 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 these 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 (2003) 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 (2002) says that tangency discontinuities in CAD models cause these high-curvature sections. Figure 2-5 shows a tool path with high-curvature ripples caused by a severe tangency discontinuity.

Even when a B-spline is interpolated to fit a curvature-continuous surface, small ripples can still form because interpolation points are non-continuous values along a continuous surface. The position, tangent line, and curvature of the surface change discretely between interpolation points creating the same affect as a surface with minute discontinuities. These minute discontinuities create small ripples in the tool path. The amplitude of the ripples correlates directly to the severity of the discontinuity.

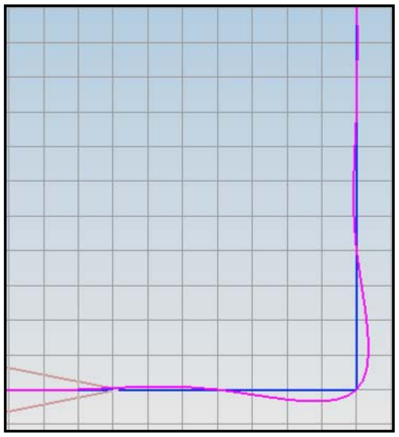

Figure 2-5 High-curvature ripples caused by a tangency discontinuity. Blue line represents surface and purple line is tool path

Not all high-curvature sections are created by anomalies. Most are created by the designer to fulfill a design requirement. Complex parts may contain unnecessarily small radii with high-curvatures. The designer may not realize the affect 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 shown in Figure 2-6.

Many researchers have tried to decrease the curvature of splines. Bohez, Lee, and Tang (2002; 1993; 1999) effectively filter noisy models, but none minimizes curvature and limits the tool path to within the path tolerance band. Eilers (1996) minimizes curvature by putting a penalty on sections with high curvature. He smoothes out splines that represent statistical data, but he does not consider NC tool paths constrained to a path tolerance.

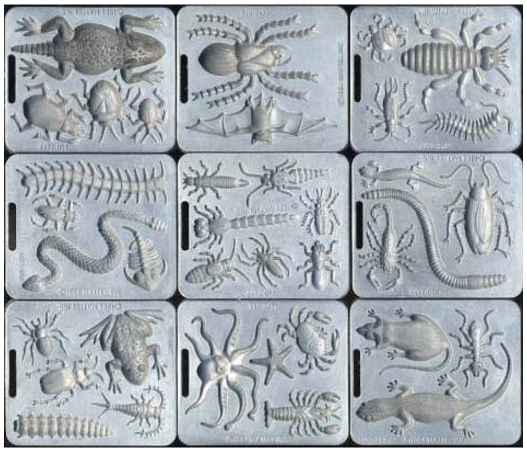

Figure 2-6 Molds for children's toys that contain very high-curvature. Curvature of molds could be slightly reduced without affecting their quality.

Langeron (2004) limits his tool paths to within a tolerance tube. He interpolates random points along the desired part surface without regard to curvature. As long as the spline remains within the tolerance band, he uses the spline parameters. However, Langeron's splines can produce even more high-curvature sections as the path wiggles back and forth within the tolerance band. Figure 2-7 shows an ideal tool path (turquoise), a path tolerance around that path (blue), and a splined tool path (purple) that interpolates random points.

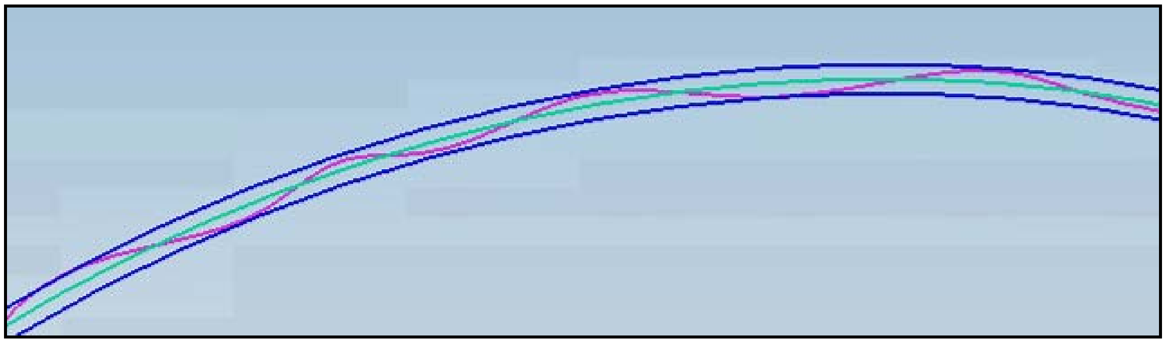

Figure 2-7 Ideal tool path (turquoise), path tolerance around that surface (blue), and splined tool path (purple) that interpolates random points. Splined tool path has excessive curvature.

The techniques used to smooth the paths along high-curvature sections in racecar tracks can be applied to high-curvature radii in tool paths. Velenis (2005) calculates the optimal path around a single corner given the physical limits of a racecar. The work of Beckman, Blinkhorn, Line (racing), Racing Line, the Racing Line, and Velenis (2007; 2006; 2007; 2007; 2007; 2005) can be used to better understand the optimal path around a curve.

In summary, although much work has been done to generate optimal tool paths, to date none creates smooth paths without unnecessary high-curvature sections nor does any deviate the tool path within a path tolerance band.

2.3 B-splines Review

This section is a review of B-spline concepts. Most of this section is based on work available from Sederberg (2007). B-splines are created by connecting Bezier curves end to end with C^{n-1} continuity where n is the degree of the B-spline.

2.3.1 Bezier Curves

Bezier curves were created by Dr. Pierre Bezier in the early 1960's as a tool for designers and artists who wanted to intuitively create splines. Bezier curves are created using a control polygon that is made up of control points as can be seen in Figure 2-8.

Dr. Pierre Bezier designed the Bezier curve so that it would mimic the shape of its control polygon, go through the first and last point in its control polygon, and be 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. The center of mass is calculated using Equation (2-1).

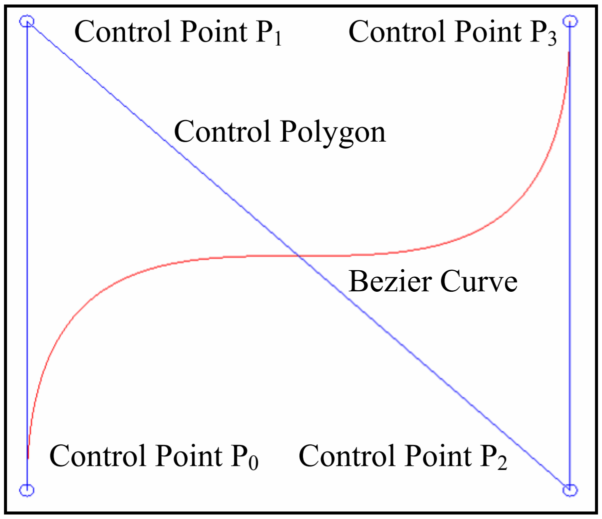

Figure 2-8 Bezier curve (red) and its control polygon and points (blue)

$$\overline{\mathbf{P}} = \sum_{i=0}^{n} \frac{m_i \mathbf{P}_i}{m_i} \tag{2-1}$$

If there are four equal masses distributed as shown in Figure 2-9 then their center of mass is

$$\overline{\mathbf{P}} = \frac{m_0 \mathbf{P}_0 + m_1 \mathbf{P}_1 + m_2 \mathbf{P}_2 + m_3 \mathbf{P}_3}{m_0 + m_1 + m_2 + m_3} \tag{2-2}$$

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 in Figure 2-8 become

$$m_0(t) = (1-t)^3 \tag{2-3}$$

$$m_1(t) = 3t(1-t)^2 \tag{2-4}$$

$$m_2(t) = 3t^2(1-t) \tag{2-5}$$

$$m_3(t) = t^3 \tag{2-6}$$

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. The curve formed in Figure 2-8 is a cubic Bezier curve (cubic is degree three). Note in Equation (2-7) that the mass equations sum to one and the equation for the Bezier curve can be written as Equation (2-8).

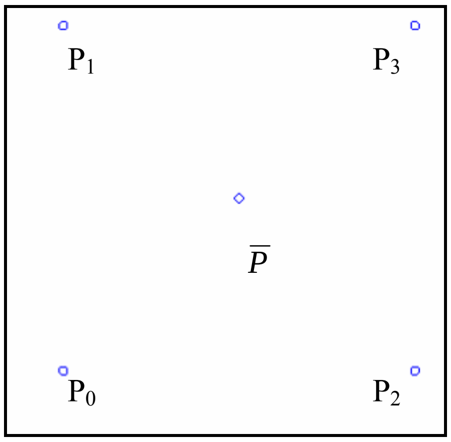

Figure 2-9 Center of mass of four points

$$(1-t)^3 + 3t(1-t)^2 + 3t^2(1-t) + t^3 = [(1-t)+t]^3 \equiv 1 \tag{2-7}$$

$$\mathbf{P}(t) = m_0(t)\mathbf{P}_0 + m_1(t)\mathbf{P}_1 + m_2(t)\mathbf{P}_2 + m_3(t)\mathbf{P}_3 \tag{2-8}$$

Figure 2-10 is the graph of the mass equations. Note that when $t = 0$, the curve passes through P_0 because $m_0 = 1$ and $m_1 = m_2 = m_3 = 0$. Also, when $t = 1$, the curve passes through P_3 because $m_3 = 1$ and $m_0 = m_1 = m_2 = 0$.

The locations, P_i, are the control points of the control polygon while the variable masses, $m_i(t)$, are normally called blending functions. In the case of Bezier curves, they are also called Bernstein polynomials. Bezier curves come in any degree and an *n* degree Bezier curve has *n+1* control points. The Bezier blending functions are defined in Equation (2-9).

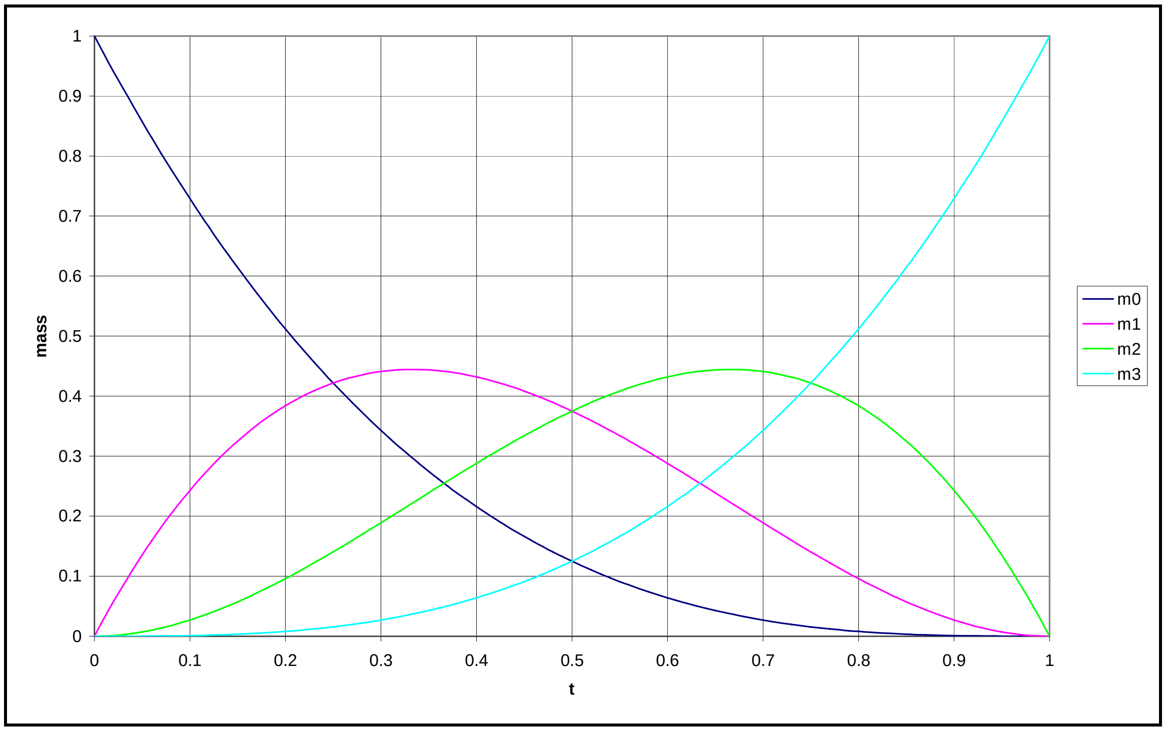

Figure 2-10 Mass functions of a Bezier curve. They are also known as blending functions. Bezier curve is weighted sum of these functions.

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i \tag{2-9}$$

where:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

For a cubic Bezier curve, $n = 3$ and $B_i^n(t)$ in Equations (2-10), (2-11), (2-12), and (2-13) represent the Bernstein polynomials.

$$B_0^3(t) = (1-t)^3 \tag{2-10}$$

$$B_1^3(t) = 3t(1-t)^2 \tag{2-11}$$

$$B_2^3(t) = 3t^2(1-t) \tag{2-12}$$

$$B_3^3(t) = t^3 \tag{2-13}$$

Equation (2-14) is the general equation for a Bezier curve. See Dr. Sederberg's text for more on Bezier curves including how to calculate position or curvature at any parameter value.

$$\mathbf{P}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{P}_i = \sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^i\mathbf{P}_i \tag{2-14}$$

2.3.2 B-splines

To create long complex curves in a simple yet robust way, 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. Not only does the B-spline guarantee C^{n-1} continuity, it also uses less control points than would be used by representing the same spline with multiple separate Bezier curves. An open string of *m* Bezier curves will contain *nm+1* control points while the same spline can be represented by a B-spline with only $m+n$ control points.

B-splines are very similar to Bezier curves in their form and function. If a B-spline consists of Bezier curves of degree n, then the B-spline is degree *n*. 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. For example, if the knot vector of a cubic B-spline is $\begin{bmatrix}0 & 0 & 0 & 2 & 3 & 5 & 5 & 5\end{bmatrix}$ then there are three Bezier curves in the spline. The three Bezier curves would go from $t = 0$ to $t = 2$, $t = 2$ to $t = 3$, and $t = 3$ to $t = 5$ respectively. Note that the knot vector contains n-1 extra knots at each end. These extra knots are used to determine the end conditions of the spline. When there are n-fold knots at both ends of the B-spline, it is said to have Bezier end conditions. The knot vector above has Bezier end conditions because the first three knots are 0 and the last three knots are 5. A B-spline with Bezier end conditions goes through its endpoints and is tangent to the first and last legs of its control polygon, just like a Bezier curve. This paper only considers cubic B-splines with Bezier end conditions, so no further end conditions will be discussed. See Dr. Sederberg's text for more on end conditions.

The B-spline equation is very similar to the Bezier curve equation. The main differences in the two equations are that the B-spline equation is written in polar form and that it uses different blending functions. See Equation (2-15).

$$\mathbf{P}(t) = \sum_{j=i}^{n+i} B_{j+i-1}^{n} \mathbf{P}\left(t_{j+1-n}, \ldots, t_{j}\right) \tag{2-15}$$

It is not necessary to understand all of the properties of B-splines to understand this thesis, so only the key concepts are covered. If the reader wishes to learn more about polar form and B-spline blending functions, they are discussed in detail in chapter 6 of Dr. Sederberg's text.

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. Figure 2-11 shows cubic B-splines with these shapes.

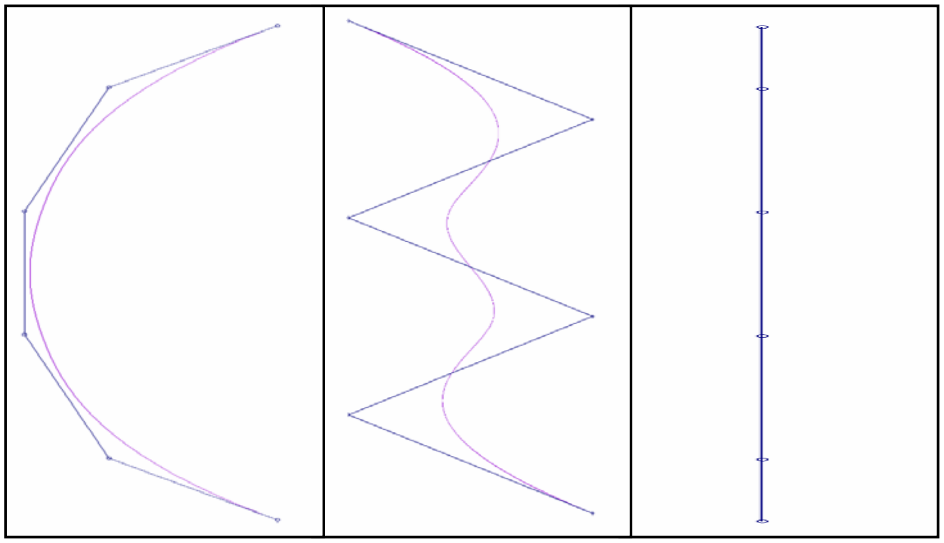

Figure 2-11 B-splines (magenta) with control polygons (blue) shaped like a semicircle, a saw-tooth, and a flat straight line, respectively. Note that B-spline takes general shape of its control polygon.

2.3.3 Convex Hull Property

Both Bezier curves and B-splines always remain within the convex hull of their control points. The convex hull can be imagined by placing a peg at each control point and then wrapping a string between the pegs. See Figure 2-12. 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.

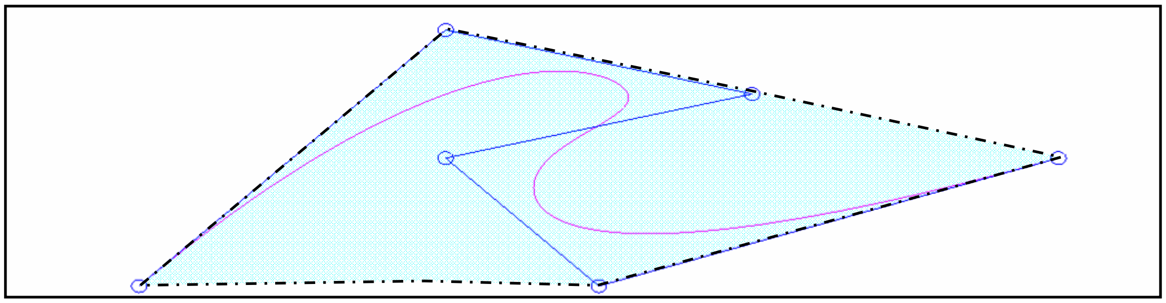

Figure 2-12 Convex hull (shaded light blue) of a B-spline (magenta)

2.3.4 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 (2-16) is the spline that represents the distance between the two splines. Note that this equation will work for both Bezier curves and B-splines because both have equations with the same basic form. Figure 2-13 shows a B-spline that is the difference of two other B-splines.

$$\mathbf{D}(t) = \mathbf{P}(t) - \mathbf{Q}(t) = \sum_{i=0}^{n} (\mathbf{P}_i - \mathbf{Q}_i) B_i^n(t) = \sum_{i=0}^{n} \mathbf{D}_i B_i^n(t) \qquad (2\text{-}16)$$

where:

$$\mathbf{P}(t) = \sum_{i=0}^{n} \mathbf{P}_i B_i^n(t)$$

$$\mathbf{Q}(t) = \sum_{i=0}^{n} \mathbf{Q}_i B_i^n(t)$$

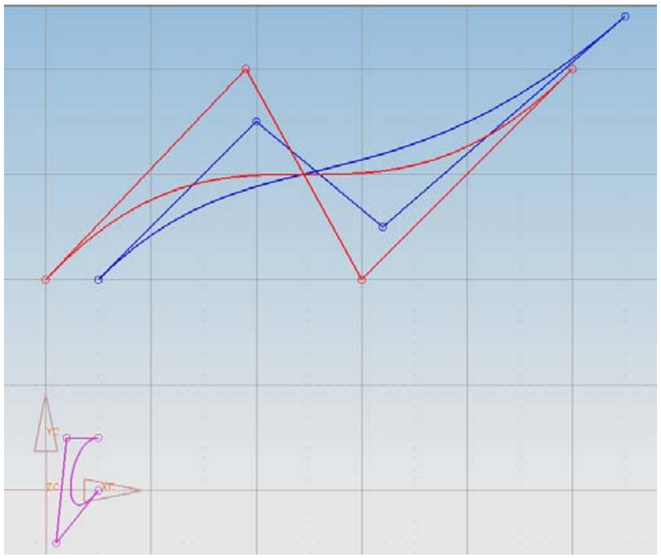

Figure 2-13 Difference B-spline (purple) which represents difference between two other B-splines (blue and red).

The convex hull property guarantees that the distance between the two curves is bounded by the largest distance from the origin to any of the control points, $\mathbf{D}_i$, 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. If a Bezier curve, **P**(t), is moved to become **Q**(t) and the control point, $\mathbf{D}_3$, is moved the furthest then the spline will not move farther than $|\mathbf{D}_3|$.

2.3.5 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

$$\mathbf{P}(t_1) = \mathbf{Q}(t_1), \mathbf{P}'(t_1) = \mathbf{Q}'(t_1), \ldots, \mathbf{P}^{(k)}(t_1) = \mathbf{Q}^{(k)}(t_1) \tag{2-17}$$

C^0 means the two curves share their endpoints' position and parameter value. C^1 means the two curves share the same end point and tangent vector (including magnitude). C^2 has the same continuity as C^1, but additionally has second-order parametric derivative continuity, and similarly for up to C^k continuity.

Parametric continuity C^k depends on the parameterization of the B-spline, 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. First order geometric continuity (G^1) means that the control polygons are colinear where they connect. This is also called tangency continuity. Curvature continuity or G^2 continuity is when the curvature is equal for both curves where they meet.G^2 continuity is sufficiently smooth for most tool paths.

2.3.6 Path Tolerance

The path tolerance is formed by offsetting the given tool path in both directions by the path tolerance radius ρ. The offset is formed by the set of all points that lay a perpendicular distance ρ from the given curve. See Equation (2-18). Using a -ρ will offset the curve in the opposite direction. Figure 2-14 shows a B-spline tool path with a calculated path tolerance.

$$\Omega(\rho, \mathbf{P}(t)) = \mathbf{P}(t) + \rho \frac{(y'(t), -x'(t))}{\sqrt{x'^2(t) + y'^2(t)}} \tag{2-18}$$

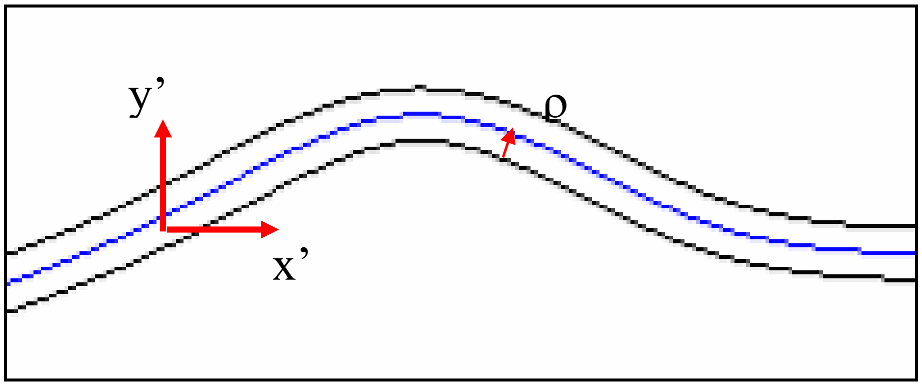

Figure 2-14 Path tolerance (offset) of B-spline tool path

If the path radius is less than the minimum radius of curvature of the spline then cusps will be formed in the offset as shown in Figure 2-15. If there are cusps in the path offset, the designer should choose a smaller path tolerance radius. It is important to note that the algorithm will still work if cusps are formed but the curvature around the cusp may not be reduced efficiently.

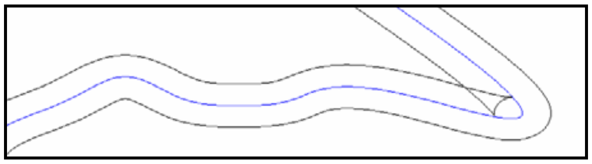

Figure 2-15 Cusps formed in tool path offset. Cusps are formed when path tolerance radius is smaller than minimum radius of curvature of tool path.

3 Method - Smoothing Algorithm

3.1 Problems with High Curvature

In High-curvature sections, a NC machine reduces the feedrate of a machine tool to remain within acceleration and jerk limits. Total acceleration must remain below the machine tool's acceleration limit and is the vector sum of both tangential and centripetal acceleration as seen in Equation (3-1) and Figure 3-1. High curvature increases centripetal acceleration as shown in Equation (3-2). As centripetal acceleration increases with high curvature, tangential acceleration decreases as shown in Equation (3-1). The obtainable feedrates decrease as tangential acceleration decreases.

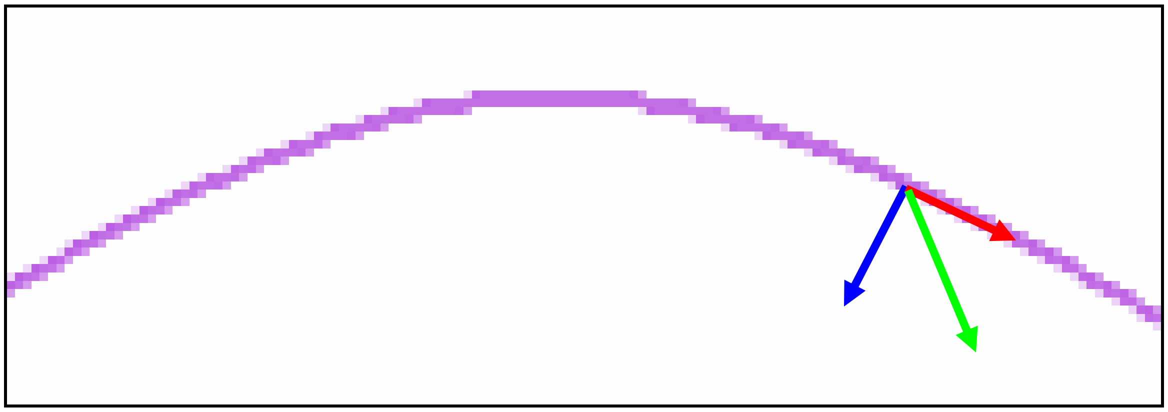

Figure 3-1 Acceleration along tool path. Tangential acceleration (red), centripetal acceleration (blue), and total acceleration (green)

$$\mathbf{A}_{\text{total}} = \mathbf{A}_{\text{centripital}} + \mathbf{A}_{\text{tangential}} \tag{3-1}$$

$$A_{\text{centripital}} = k_{\text{curvature}} V_{\text{feedrate}}^{2} \tag{3-2}$$

At the apex of a high-curvature ripple, the feedrate is a local minimum and the tangential acceleration is zero because the curvature is a local maximum. See Figure 3-2. The feedrate at an apex is calculated using Equation (3-3) that is derived from Equation (3-1) and Equation (3-2). In Equation (3-1), A_{total} is A_{max} and $A_{\text{tangential}}$ is zero. Equation (3-1) is substituted into Equation (3-2) and Equation (3-2) is solved for V_{feedrate} resulting in Equation (3-3). The feedrate is limited to these minimum feedrates at the apex of the high-curvature ripples. It is also limited by the acceleration between the apexes.

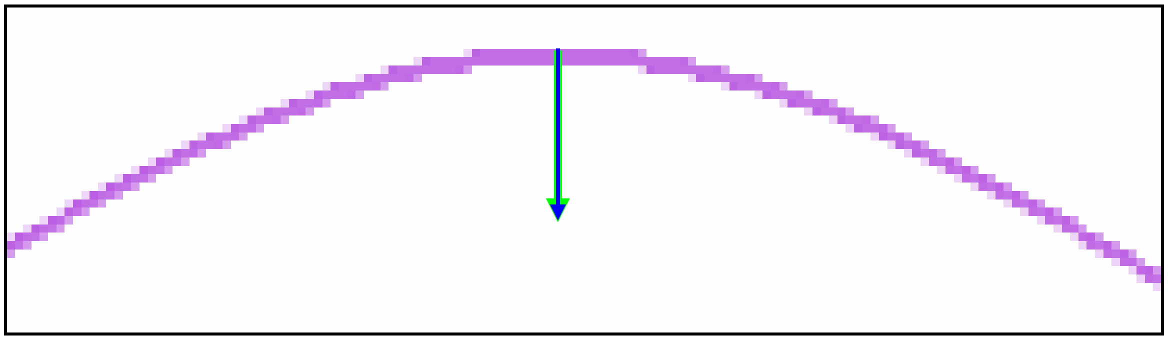

Figure 3-2 Centripal acceleration at apex of high-curvature ripple.

If the minimum velocity calculated in Equation (3-3) is greater than the maximum velocity limit then the curvature in the ripple is not severe enough to reduce the feedrate.

The maximum allowable curvature that will not reduce the feedrate is calculated by using Equation (3-4) that is derived from Equation (3-3).

$$V_{\text{feedrate}} = \sqrt{\frac{A_{\max}}{k_{\text{curvature}}}} \tag{3-3}$$

$$k_{max} = \frac{A_{\max}}{V_{\max}^{2}} \tag{3-4}$$

For example, if the $A_{\max}$ is 4000 mm/s^2 and the $V_{\max}$ is 400 mm/s then the maximum curvature is 0.025 1/mm. If a ripple has a curvature less than 0.025 1/mm then it will not affect the feedrate.

$$k_{max} = 0.025\frac{1}{mm} = \frac{4000\frac{mm}{s^2}}{\left(400\frac{mm}{s}\right)^2} \tag{3-5}$$

The smoothing algorithm in this thesis reduces the curvature of all ripples even if the curvature is below the value calculated in Equation (3-4) because it takes longer to calculate the curvature to distinguish between negligible and significant curvature values than it does to smooth everything. The algorithm efficiently reduces the curvature in tool paths without calculating the curvature. The algorithm smoothes a tool path by using general concepts of B-splines.

3.2 Algorithm Overview

A program was written that inputs spline data from a data file, modifies the spline to reduce the curvature, and prints out the new spline data. The algorithm in the program utilizes a very simple concept to smooth the curves. When all of the control points of a B-spline form a single straight line, the spline becomes a straight line with zero curvature. This formation also forms the shortest possible path between the two endpoints of the B-spline. The algorithm attempts to move all of the spline's control points to form a straight line. The control points will rarely form a straight line because the spline is constrained to the tolerance tube.

A direction vector is calculated 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. The fact that curvature is reduced by flattening a control polygon into a straight line is proven by Equation (3-6), the equation for the curvature of a B-spline. To use Equation (3-6), 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's control polygon are shown in Figure 3-3. $\mathbf{P_0}$ is the point where the curvature is being calculated and the first point in the Bezier's control polygon. The distance between $\mathbf{P_0}$ and $\mathbf{P_1}$ is the value a. The value h is the distance from $\mathbf{P_2}$ to the line formed by $\mathbf{P_0}$ and $\mathbf{P_1}$. If a line is flattened, h is decreased and the curvature, k, is decreased.

$$k = \frac{n-1}{n}\frac{h}{a^2} \tag{3-6}$$

where:

n = degree

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

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

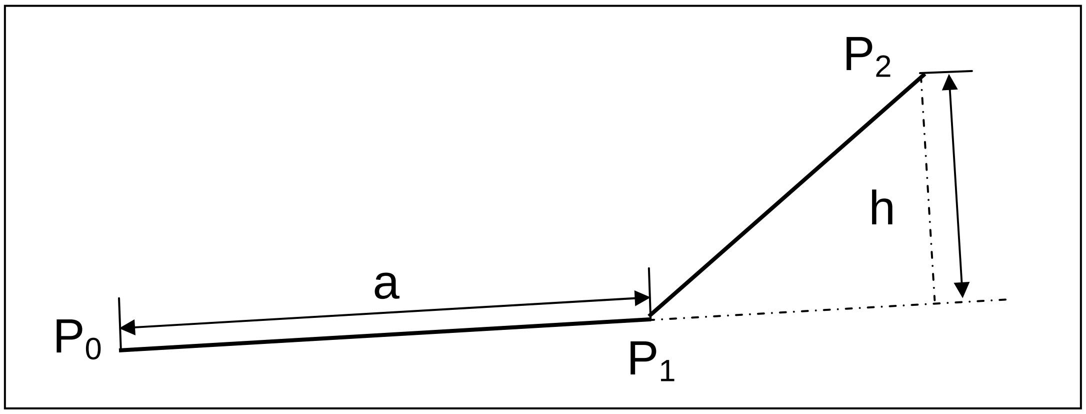

Figure 3-3 Parameters to calculate curvature of B-spline.

The control points are placed in one of three categories depending on how they need to move to form a straight line. The first category contains the endpoints of the control polygon. Each endpoint is smoothed by moving it towards the line formed by the next two control points in the control polygon. The direction vectors for endpoints are perpendicular to the initial control polygon and are discussed in detail in Section 3.6.1. In Figure 3-4, point 0 moves perpendicular to the control polygon and towards the line formed by points 1 and 2.

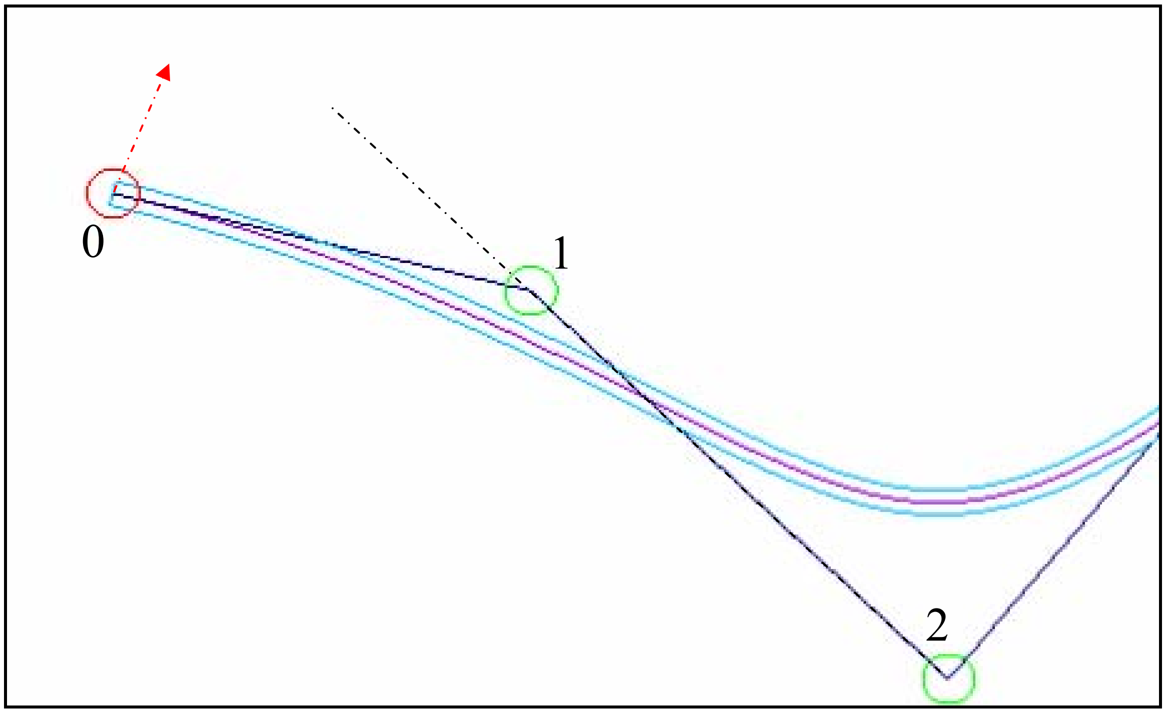

Figure 3-4 Direction vector (red) for endpoint

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. 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. The equation for the normalized bisector of a triangle is discussed in Section 3.6.2. As the ripple point moves along the bisector, the triangle flattens and forms a straight line. This will smooth and shorten the path. Figure 3-5 shows a few ripple points circled in green. Point 2 is moved along the bisector towards point 2' that forms a straight line with points 1 and 3. Note that the ripple points form high-curvature sections in the B-spline due to the saw-tooth pattern. Control points on a straight line are also considered ripple points with a zero magnitude direction vector.

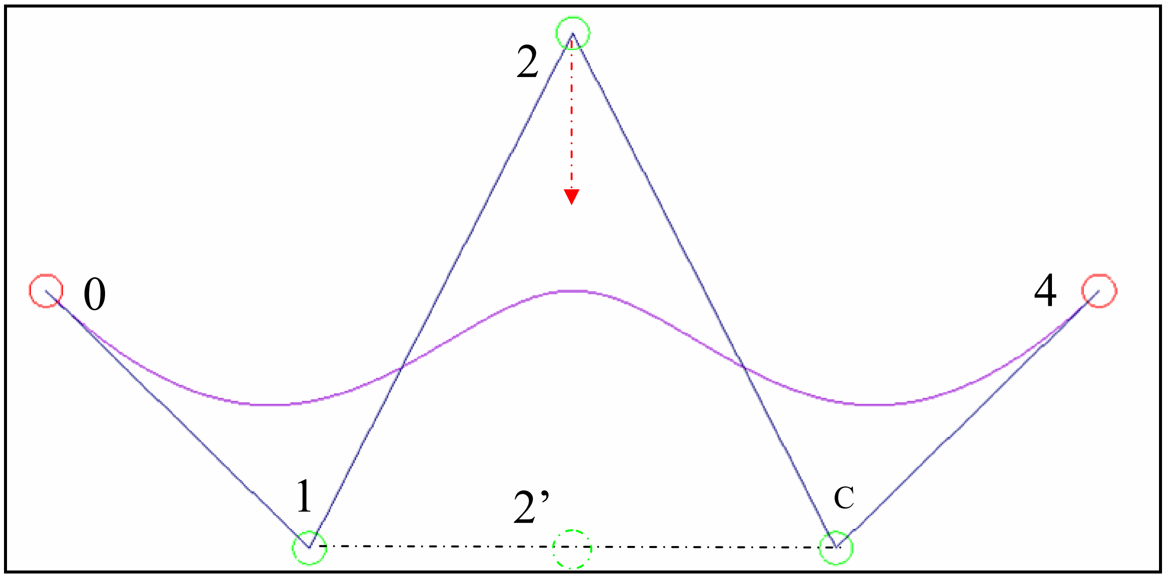

Figure 3-5 B-spline with ripple points (green). Point 2 is moved towards point 2' along direction vector

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. When a point is recognized as a smooth point, both the points before and after it are also categorized as smooth points. A group of smooth points consists of all adjacent smooth points. Each smooth control point receives an additional label according to its location in the group.

In each group there is a beginning point, middle points, one or two apex points, and an ending point. The beginning and ending points are the first and last points in each group. The apex point(s) are the median point(s) in each group. If there is 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. In Figure 3-6, points 4 and 9 are the beginning and ending points, respectively. Points 5 and 8 are middle points and points 6 and 7 are the apex points.

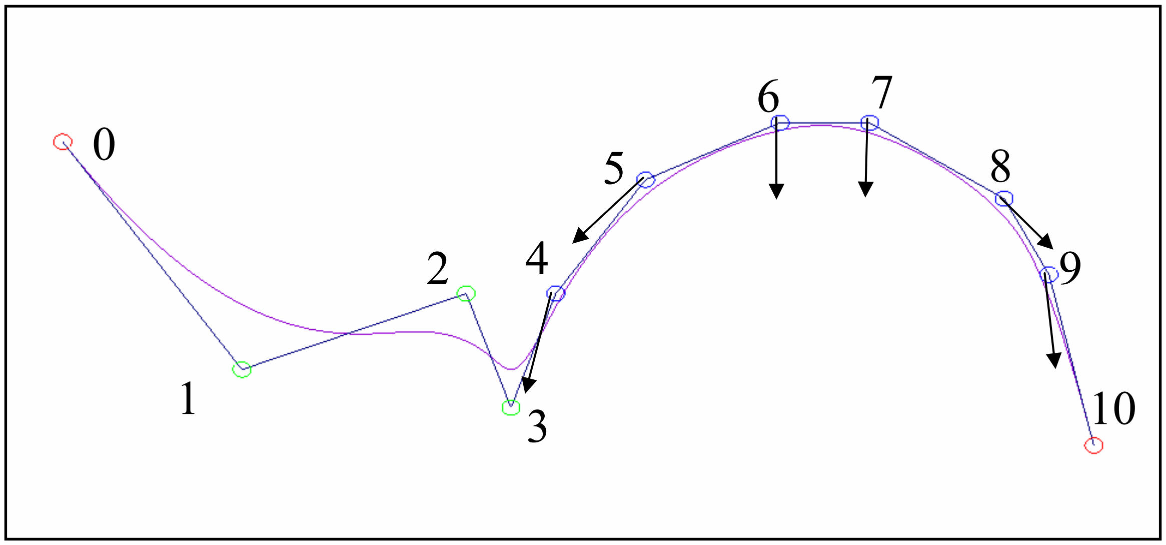

Figure 3-6 B-spline with smooth points (blue).

The direction vector for each control point moves the smooth group towards two objectives. The end and middle control points are moved outward from the center of the arc while the apex points are moved towards the center of the arc effectively flattening the arc. In addition, all of the points in a smooth group are moved towards the center of the group. This moves the smooth section so that it will touch at the inside corner of the tolerance tube in the smooth section. This not only smoothes the spline; it also shortens the spline. Figure 3-7 shows how the blue spline is spread out and moved down by the algorithm to form the red spline. Notice how the red spline is spread out until it touches the black tolerance tube on the outside at the end points and shifted down until it touches the black tolerance tube on the inside of the center of the arc. Also note how this appears to be a path that a racecar driver would take around a corner.

After all of the points are categorized and their direction vectors are calculated, the distance that each control point moves along its direction vector is calculated. To guarantee that the spline remains within the path tolerance, the control points are moved at most the minimum distance between the path and the path tolerance tube. The convex hull principle for B-splines guarantees that a spline will not move more than the control point that is moved the most in the control polygon. The spline will remain within the tolerance because each control point is moved no more than the minimum distance between the spline and the path tolerance.

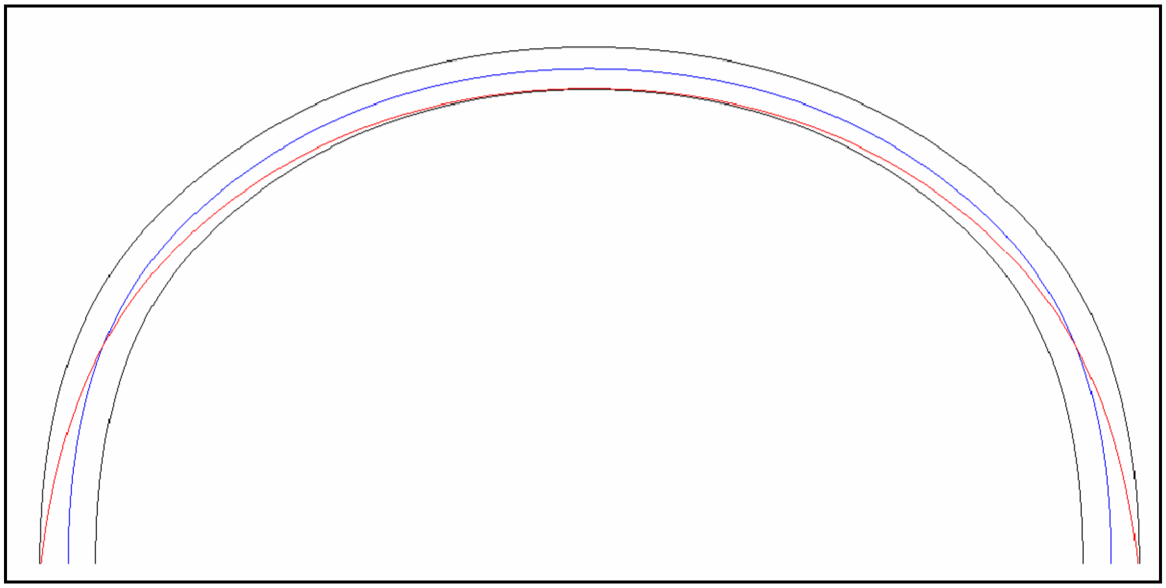

Figure 3-7 Spline that has been flattened. Spline is spread out and moved down by algorithm to form red spline. Notice how red spline is spread out until it touches black tolerance tube on outside and shifted down until it touches black tolerance tube on the inside.

Based on the control polygon's geometry, the spline responds differently to the movements of different control points. The spline will move further with the movement of some control points than others. If all of the control points are moved a distance equal to the radius of the tolerance, the endpoints of the spline move completely to the edge of the tolerance tube while points on the spline corresponding to ripple points hardly move at all. To ensure that the whole spline is moved and smoothed evenly, the distance that each control point moves is scaled using sensitivity values as explained in Section 3.7. The spline will not smooth completely in one step so the process is iterated until the spline touches the path tolerance. 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. When the process is complete, the spline will be smooth and almost touch the tolerance tube at the inside of every ripple. See Figure 3-8.

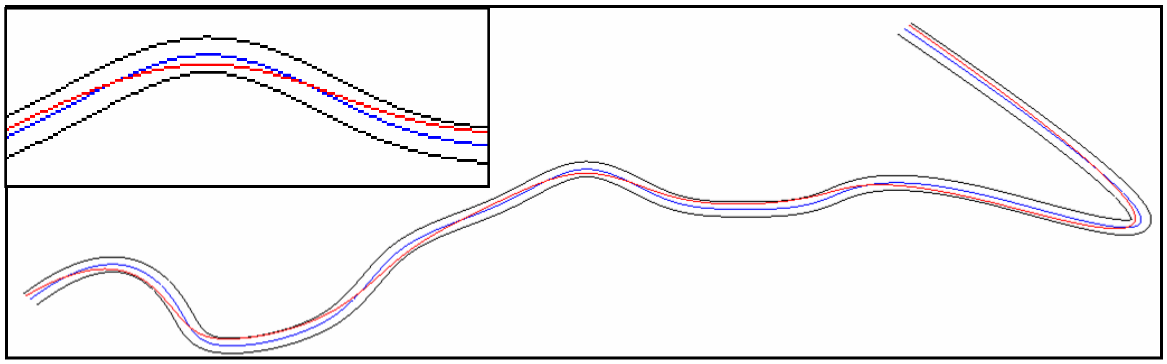

Figure 3-8 Smoothed tool path

3.3 Implementation of Algorithm

The steps of the algorithm are described in detail in Sections 3.4 through 3.11. The algorithm reads in the initial spline data, calculates the initial parameters for the iterations, and then iterates until the spline converges to a smooth spline bounded by the tolerance tube.

3.4 Parsing

The algorithm parses the control polygon, knot vector, path tolerance, precision, and accuracy from the data file input. The control polygon consists of the control points for the CAM-generated spline. The knot vector consists of the knots for the control polygon. There are always two less knots than control points in an open cubic B-spline. The path tolerance is the radius of the path tolerance around the CAM-generated spline. The precision is the number of points checked on each knot interval of the B-spline per iteration. The accuracy is the minimum percentage of the tolerance path radius allowed. The spline is considered to be touching the tolerance tube when it comes to within this accuracy.

3.5 Categorization of Each Point

The control points are categorized into end, ripple, and smooth categories during each iteration. Points can change categories as the control polygon moves and changes shape. Figure 3-9 and Figure 3-10 show an example of a smooth point becoming part of a straight line, which causes it to be reclassified as a ripple point. Points 1, 2, 3 and 4 make up one smooth section while points 5, 6, 7, and 8 make up another. Note that there are no ripples between the two smooth sections in Figure 3-9. But after the smoothing algorithm is applied, there is a ripple point. In Figure 3-10, points 4, 5 and 6 form a straight line and point 5 has become a ripple point because control points in the middle of a straight line are considered ripple points.

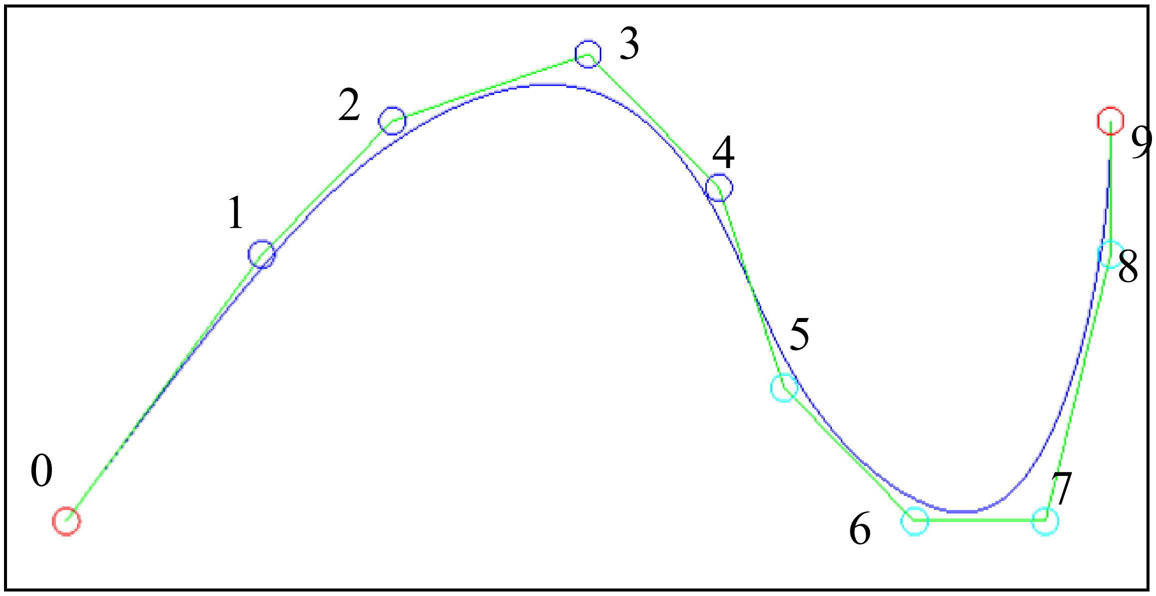

Figure 3-9 B-spline with no ripple points

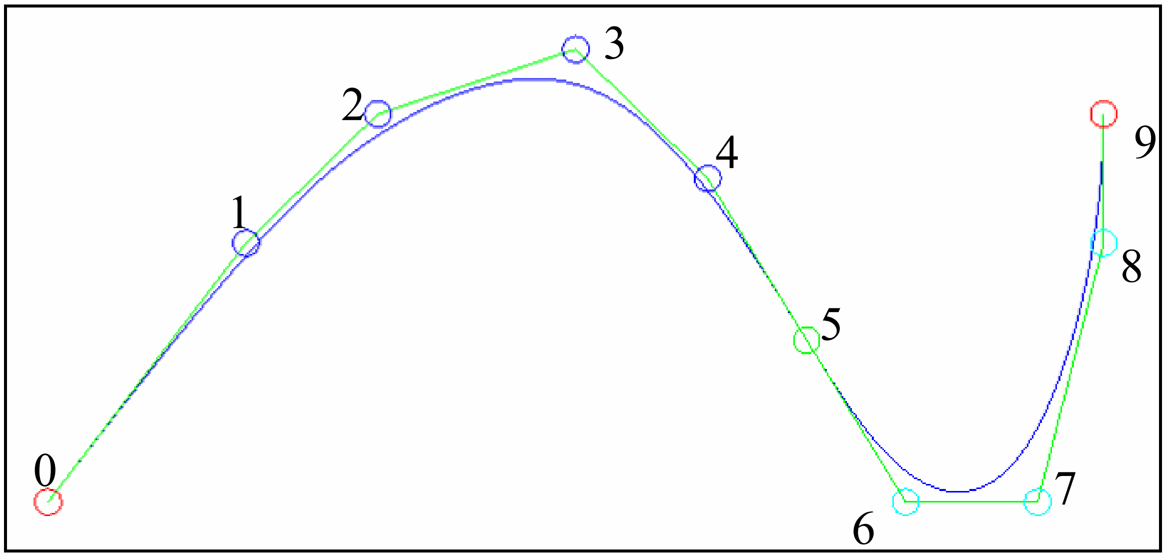

Figure 3-10 B-spline that has been smoothed to create a straight line between points 4, 5, and 6. By definition, point 5 is now categorized as a ripple point.

The algorithm categorizes the points using the shape of the control polygon. The first and last points are always categorized as endpoints. If there are only four control points in the control polygon (a single Bezier), the two middle points are categorized as ripples. With more points, ripple points and smooth points are distinguished using lines and the control polygon's relationship to these lines. Two examples follow that illustrate how the algorithm determines if a point (point 2) is categorized as a ripple or smooth point. Figure 3-11 and Figure 3-12 illustrate how point 2 is categorized as a ripple point and Figure 3-13 and Figure 3-14 illustrate how it is categorized as a smooth point.

To distinguish between ripple and smooth points, the algorithm first calculates the equation of a line that goes through the point being considered (point 2) and the point just before it (point 1). The implicit equation of this line can be seen in Equation (3-7).

$$aX + bY + c = 0 \qquad \textbf{(3-7)}$$

where:

$a = Y_{i-1} - Y_i$

$b = X_i - X_{i-1}$

$c = X_{i-1}Y_i - X_iY_{i-1}$

Figure 3-11 and Figure 3-13 show how this line splits the 2D plane into two parts: positive and negative. A point on the positive side of the line will have a positive distance value while a point on the negative side will have a negative distance value.

Next, the points immediately before and after this line (points 0 and 3) are checked to determine on which side of the line they fall. This is accomplished by finding the perpendicular distance to the line using the distance formula for an implicit line. The perpendicular distance to the line is D in Equation (3-8), where a, b, and c are calculated using Equation (3-7). Equation (3-9) is for the point before the line (point 0) and Equation (3-10) is for the point after the line (point 3).

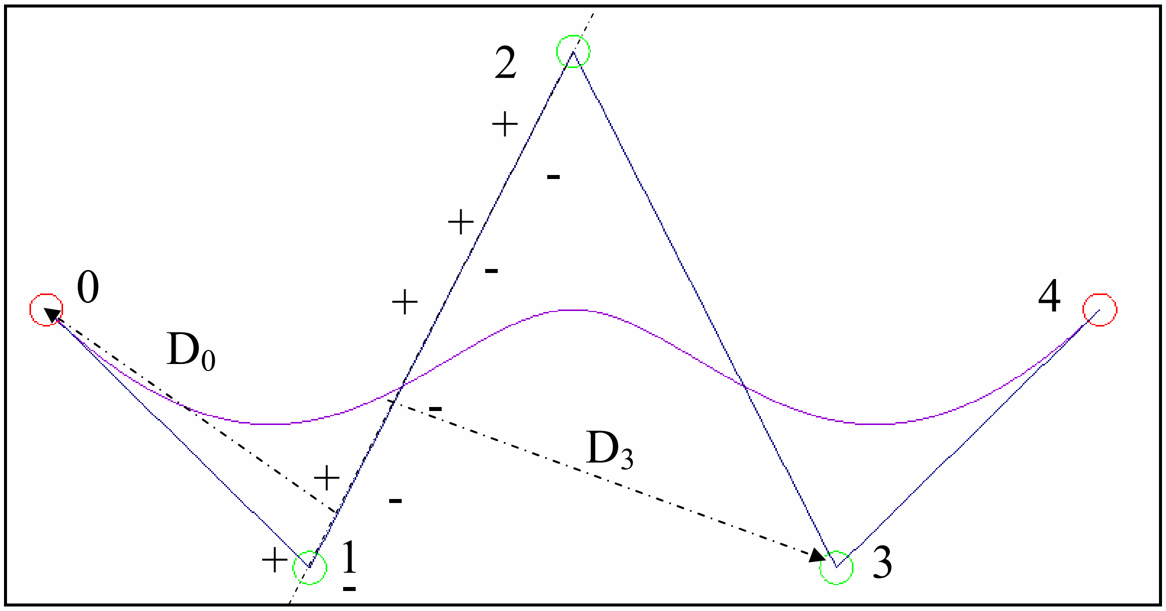

Figure 3-11 A line going through points 1 and 2 and the positive and negative sides of this line

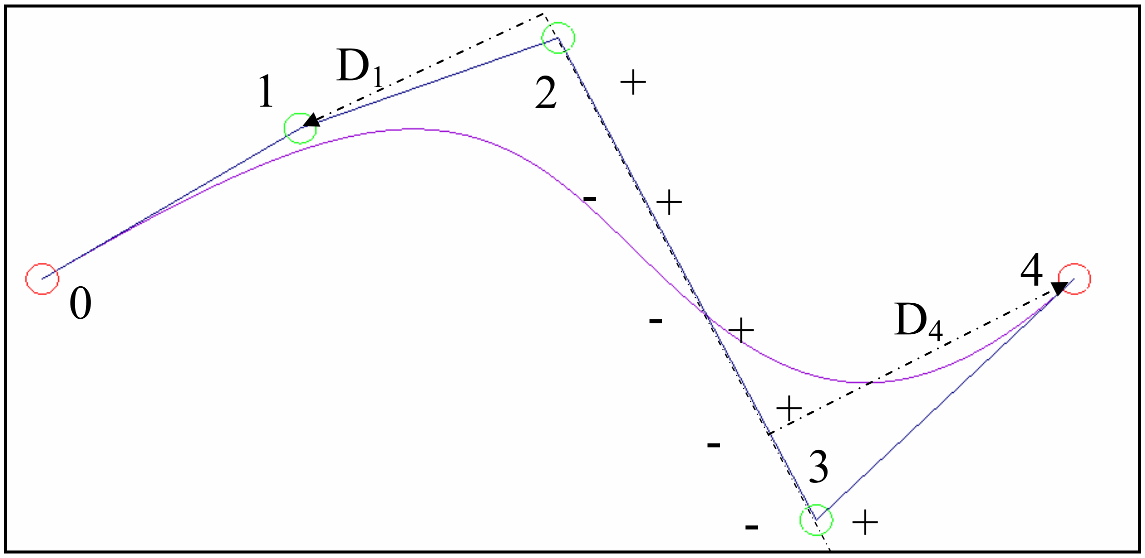

Figure 3-12 A line going through points 2 and 3 and the positive and negative sides of this line

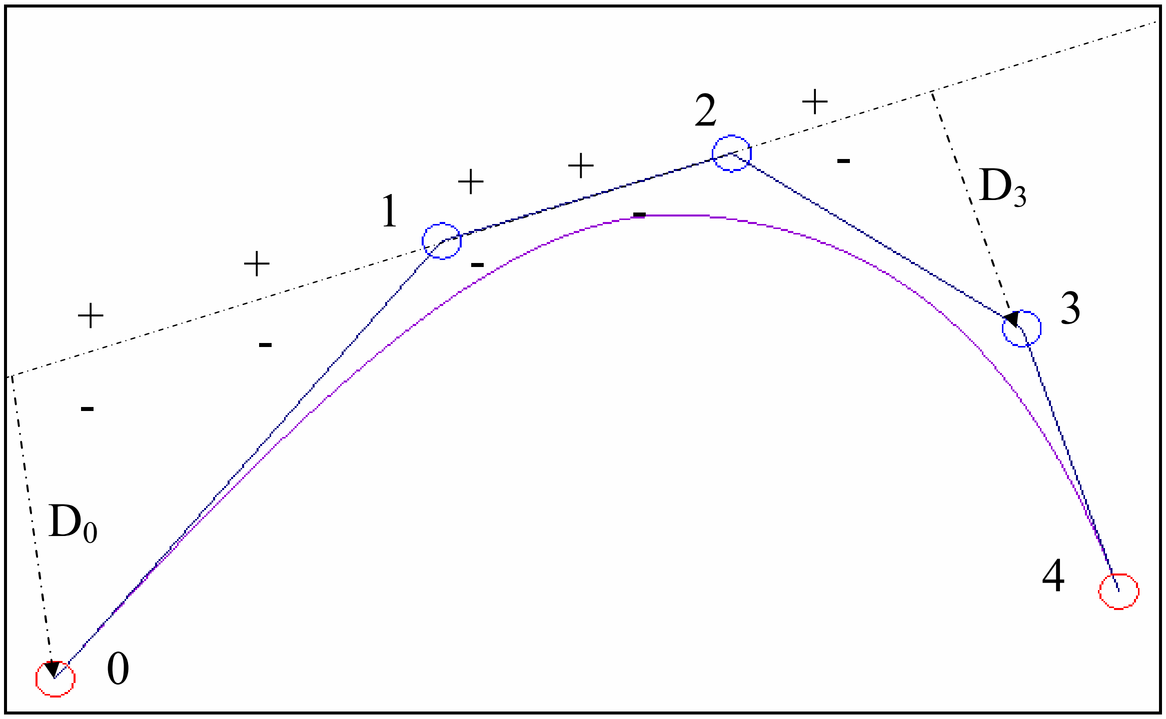

Figure 3-13 A line going through points 1 and 2 and the positive and negative sides of this line

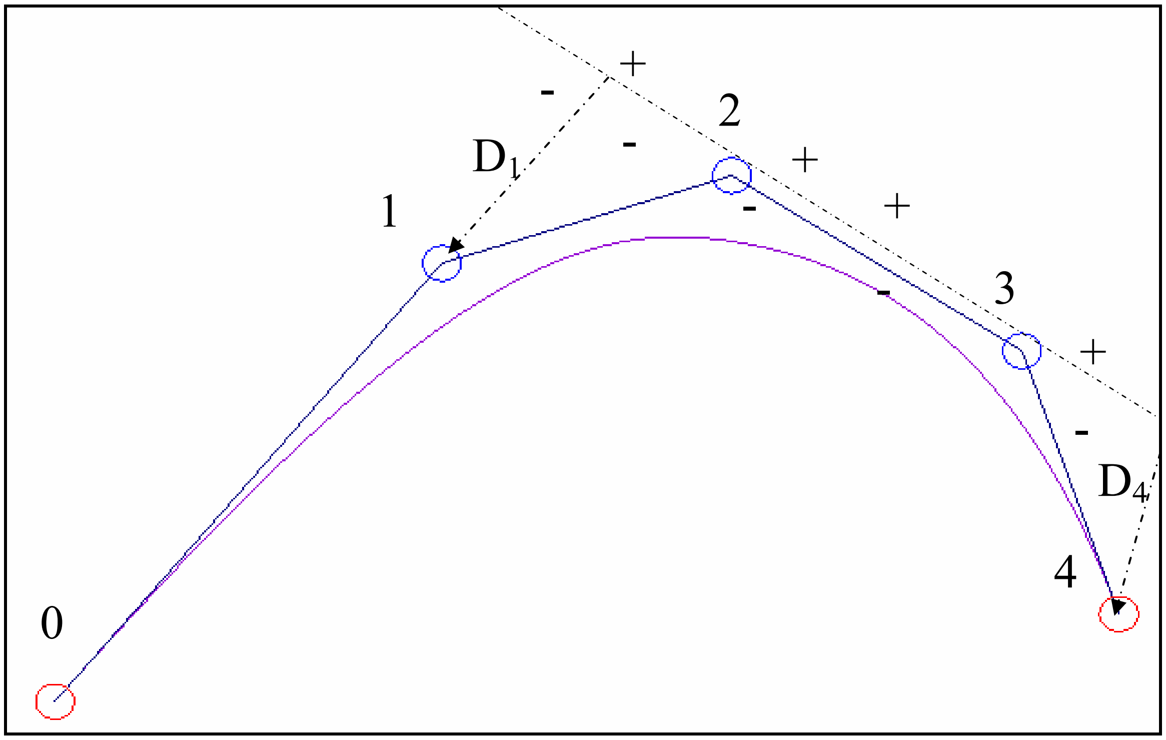

Figure 3-14 A line going through points 2 and 3 and the positive and negative sides of this line

$$D = \frac{a}{\sqrt{a^2+b^2}} X + \frac{b}{\sqrt{a^2+b^2}} Y + \frac{c}{\sqrt{a^2+b^2}} \tag{3-8}$$

$$D_{i-2} = \frac{a}{\sqrt{a^2+b^2}} X_{i-2} + \frac{b}{\sqrt{a^2+b^2}} Y_{i-2} + \frac{c}{\sqrt{a^2+b^2}} \tag{3-9}$$

$$D_{i+1} = \frac{a}{\sqrt{a^2+b^2}} X_{i+1} + \frac{b}{\sqrt{a^2+b^2}} Y_{i+1} + \frac{c}{\sqrt{a^2+b^2}} \tag{3-10}$$

The sign of the distance to the line determines which side of the line the points lay. Positive distance values are on one side of the line and negative values are on the other. The distance values for both points (0 and 3) are compared to see if the points lie on the same side of the line (same sign) or not (opposite sign). If the points are on opposite sides then the point is categorized as a ripple point. This is illustrated in Figure 3-11 where points 0 and 3 are on opposite sides of the line.

If the points lay on the same side of the line then the process is repeated using a line going from the point being considered (point 2) to the point after it (point 3). The implicit equation of this line can be seen in Equation (3-11).

$$aX + bY + c = 0 \tag{3-11}$$

where:

$a = Y_i - Y_{i+1}$

$b = X_{i+1} - X_i$

$c = X_i Y_{i+1} - X_{i+1} Y_i$

Figure 3-12 shows this new line splitting the 2D plane into positive and negative sides. Again, the points immediately before and after this line (points 1 and 4) are checked to determine on which side of the line they fall. Equation (3-8) is used with the new points to calculate the distances using Equations (3-12) and (3-13). Equation (3-12) is for the point before the line (point 1) and Equation (3-13) is for the point after the line (point 4).

$$D_{i-1} = \frac{a}{\sqrt{a^2 + b^2}} X_{i-1} + \frac{b}{\sqrt{a^2 + b^2}} Y_{i-1} + \frac{c}{\sqrt{a^2 + b^2}} \tag{3-12}$$

$$D_{i+2} = \frac{a}{\sqrt{a^2 + b^2}} X_{i+2} + \frac{b}{\sqrt{a^2 + b^2}} Y_{i+2} + \frac{c}{\sqrt{a^2 + b^2}} \tag{3-13}$$

Positive distance values are on one side of the line and negative values are on the other. The distance values for both points are compared to see if the points lay on the same side of the line (same sign) or not (opposite sign). If the points are on opposite sides then the point is categorized as a ripple point. This is illustrated in Figure 3-12 where points 1 and 4 are on opposite sides of the line.

If both points fall on the same side of the line in both of the previous tests, the point (point 2) is categorized as a primary smooth point. When a point is categorized as a primary smooth point, the two points adjacent to it in the control polygon are categorized as secondary smooth points.

Only primary smooth points, not secondary smooth points, will cause their adjacent points to become smooth. Note that the categorization of secondary smooth points will cause these points to be categorized twice. The categorization as a smooth point overrides the categorization as a ripple point. In Figure 3-13, both points 1 and 3 are considered secondary smooth points because they are adjacent to the primary smooth point 2.

A smooth group is formed by all contiguous smooth points. A smooth group consists of a beginning point, middle points, apex point(s), and an ending point.

The re-categorization of points helps determine whether a point in a smooth group is a beginning or ending point. If a point is initially categorized as a ripple and then as a secondary smooth, it is the beginning point of a smooth group. If a point is initially categorized as a secondary smooth and then would qualify as a ripple, it is the ending point of a smooth group. In Figure 3-13, points 1 and 3 are the beginning and ending points, respectively.

The second and second-to-last control points in a control polygon do not have enough points surrounding them to categorize them because the algorithm needs two points before and two points after each point to determine the category. To resolve this, the second and second-to-last points are always set as ripples unless they are adjacent to a primary smooth point, which would cause them to become secondary smooth points.

The next step is to determine the apex point(s) in a smooth group. The apex point is the median point(s) in a group of smooth points. If there is an odd number of points in a smooth group then there is only one apex point. If there is an even number of points then there are two apex points. Equation (3-14) calculates the index of a single apex point and Equation (3-15) and Equation (3-16) calculate the indices of each point of a double apex.

For an odd number of points:

$$i_{apex} = i_{beginning} + (n_s - 1)/2 \qquad (3\text{-}14)$$

For an even number of points:

$$i^1_{apex} = i_{beginning} + \frac{n_s}{2} - 1 \qquad (3\text{-}15)$$

$$i^2_{apex} = i_{beginning} + \frac{n_s}{2} \qquad (3\text{-}16)$$

In Figure 3-15, points 4 and 9 are the beginning and ending points, respectively, of a smooth group. Points 5 and 8 are middle points and points 6 and 7 are the apex points. When the categorization of all points is complete, direction vectors can be calculated for each control point.

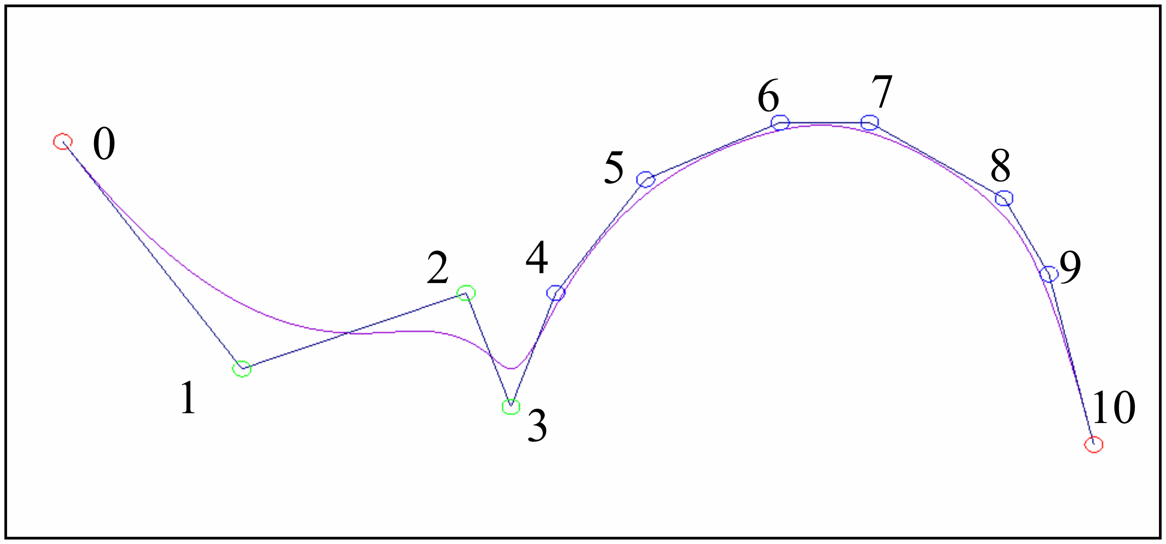

Figure 3-15 B-spline with smooth points (blue). Points 4 and 9 are beginning and ending points, respectively, of smooth group. Points 5 and 8 are middle points and points 6 and 7 are apex points.

3.6 Direction Vectors

Each control point is assigned a normalized direction vector that tells the control point which direction to move to smooth the spline. This direction vector is iteratively updated as the control polygon's shape changes.

3.6.1 Endpoints

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. As the control polygon moves during each iteration, it will no longer be parallel to the tolerance tube at the ends. However, the endpoint direction vector must remain perpendicular to the tolerance tube so that the endpoint remains at the end of the tolerance tube to prevent a gap in the tool path.

The objective is to position the endpoint to form a line with the next two (or previous two) control points in the control polygon. In each iteration, the direction vector incrementally moves the endpoint towards that line. It is possible to overshoot the target line in a given iteration. This creates the necessity of reversing the direction vector to again point towards the line. In Figure 3-16, if point 0 moves to 0' then its direction vector is reversed to again point at the target line formed by points 1 and 2. The initial direction vector is calculated using Equation (3-17) and can be seen in Figure 3-17.

At each iteration, it is determined whether the endpoint direction vector must be reversed. In addition to overshooting the target line, Equation (3-17) has the limitation that it does not guarantee that the direction vector points in the right direction. To determine if the direction vector points in the wrong direction, the minimum distance from the first point to a line going through the second and third points is calculated using Equation (3-18). Equation (3-17) assumes that the first point is on the positive side of the line. If the distance to the line is negative, the direction vector is reversed using Equation (3-19).

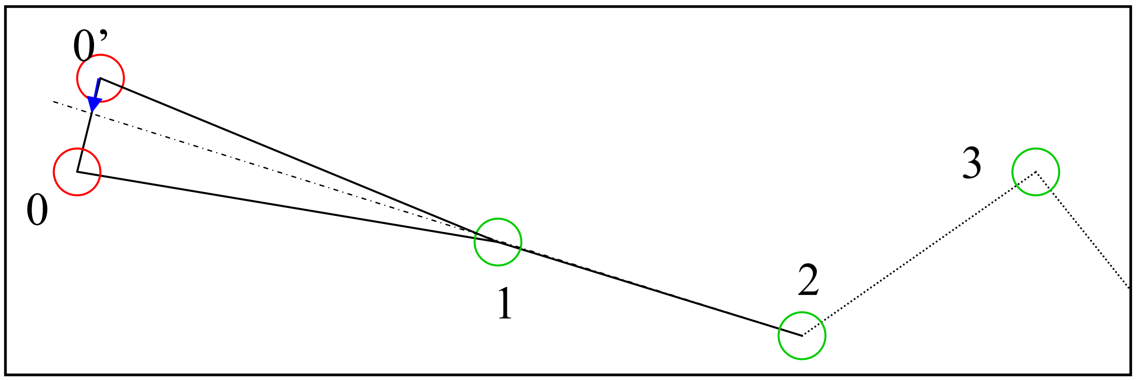

Figure 3-16 Endpoint 0 has moved to 0’, which is past target line formed by points 1 and 2, requiring that direction vector (blue) be reversed.

$$\mathbf{DV'}_0 = \begin{bmatrix} X \\ Y \end{bmatrix}_0 = \begin{bmatrix} \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{-\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \qquad \textbf{(3-17)}$$

where:

$\Delta X = X_1 - X_0$

$\Delta Y = Y_1 - Y_0$

A special case occurs if the distance to the line is zero. The point is at the target line so there is no need to move the point. The direction vector is set to null.

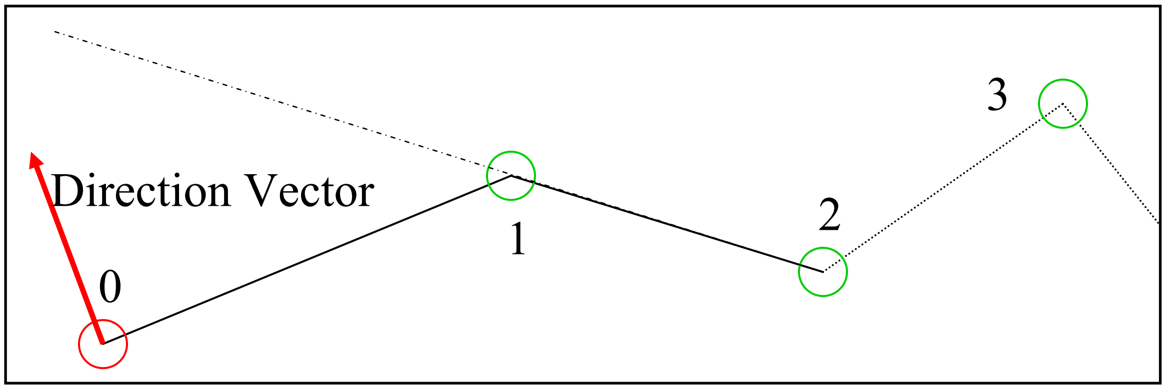

Figure 3-17 Direction vector for endpoint. It is perpendicular to original control polygon

The direction vector for the last control point is calculated in a similar manner to the first point. Equations (3-20), (3-21), and (3-22) are the modified forms of Equations (3-17), (3-18), and (3-19) for the last point. Figure 3-18 illustrates the direction vector for the last point.

$$D_0 = \frac{a}{\sqrt{a^2+b^2}}X_0 + \frac{b}{\sqrt{a^2+b^2}}Y_0 + \frac{c}{\sqrt{a^2+b^2}} \tag{3-18}$$

where:

$a = Y_1 - Y_2$

$b = X_2 - X_1$

$c = X_1Y_2 - X_2Y_1$

$$\mathbf{DV}_0 = \left(\frac{D_0}{|D_0|}\right)\mathbf{DV'}_0 \tag{3-19}$$

$$\mathbf{DV'}_{m-1} = \begin{bmatrix} X \\ Y \end{bmatrix}_{m-1} = \begin{bmatrix} \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \\ \dfrac{-\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \tag{3-20}$$

where:

$\Delta X = X_{m-2} - X_{m-1}$

$\Delta Y = Y_{m-2} - Y_{m-1}$

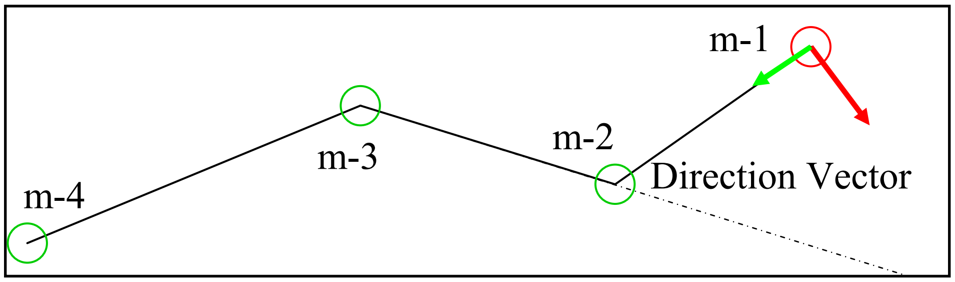

Figure 3-18 Direction vector for endpoint. It is perpendicular to original control polygon

$$D_{m-1} = \frac{a}{\sqrt{a^2 + b^2}} X_{m-1} + \frac{b}{\sqrt{a^2 + b^2}} Y_{m-1} + \frac{c}{\sqrt{a^2 + b^2}} \tag{3-21}$$

where:

$a = Y_{m-2} - Y_{m-3}$

$b = X_{m-3} - X_{m-2}$

$c = X_{m-2} Y_{m-3} - X_{m-3} Y_{m-2}$

$$\mathbf{DV}_{m-1} = \left(\frac{D_{m-1}}{|D_{m-1}|} \right) \mathbf{DV'}_{m-1} \tag{3-22}$$

3.6.2 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 direction vector (shown in red) for ripple control points is illustrated in Figure 3-19.

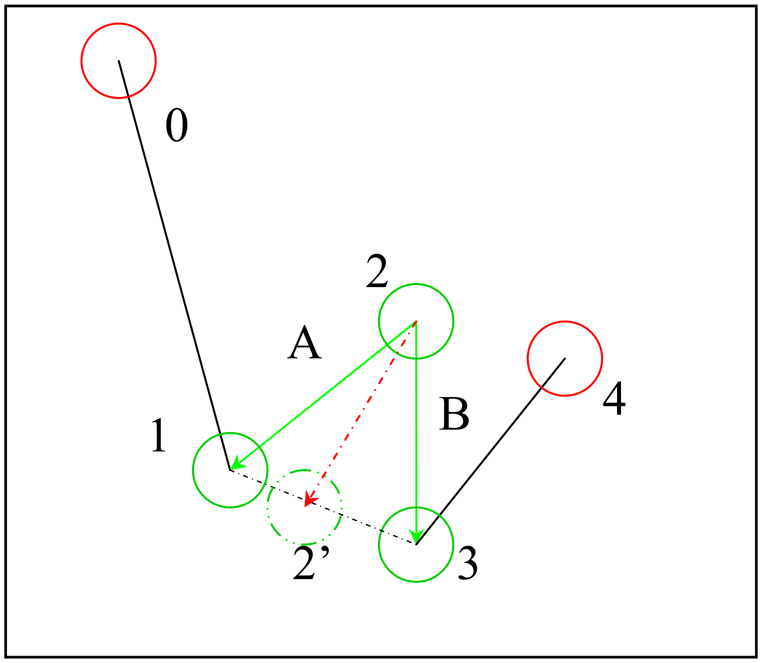

Figure 3-19 Direction vector (red) for ripple point defined as normalized sum of vectors vectors A and B

First, the normalized vectors **A** and **B** (green) are created using Equation (3-23) and Equation (3-24). Vector **A** goes from the point being considered (point 2) to the point before it (point 1) and vector **B** goes from the point being considered to the point after it (point 3). The normalized sum of vectors **A** and **B** calculated in Equation (3-25) is the direction vector and bisects the angle between **A** and **B**. If the points (1, 2, and 3) form a straight line, vectors **A** and **B** will point in opposite directions and Equation (3-25) will be $\begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$.

$$\mathbf{A}_i = \begin{bmatrix} X \\ Y \end{bmatrix}_i = \begin{bmatrix} \dfrac{\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \qquad (3\text{-}23)$$

where:

$\Delta X = X_{i-1} - X_i$

$\Delta Y = Y_{i-1} - Y_i$

$$\mathbf{B}_i = \begin{bmatrix} X \\ Y \end{bmatrix}_i = \begin{bmatrix} \dfrac{\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \qquad (3\text{-}24)$$

where:

$\Delta X = X_{i+1} - X_i$

$\Delta Y = Y_{i+1} - Y_i$

$$\mathbf{DV}_i = \left| \mathbf{A}_i + \mathbf{B}_i \right| \qquad (3\text{-}25)$$

3.6.3 Smooth Points

Direction vectors for smooth points flatten the curve and reduce the curvature as shown in Figure 3-20, resulting in a shorter curve.

The procedure to calculate the direction vectors for smooth sections combines multiple vectors and can be seen in the smooth group (points 4-9) in Figure 3-21. The apex points' direction vectors (the **A** vectors on points 6 and 7) are calculated first. Then preliminary direction vectors (the **B** vectors, blue, on points 4, 5, 8, and 9) are calculated for the middle and ending points. Each preliminary vector is then combined with a scaled version of the apexes' direction vectors (**C** vectors, black, on points 4, 5, 8, and 9) to create the non-apex final direction vector (**A** vectors, red, on points 4, 5, 8, and 9).

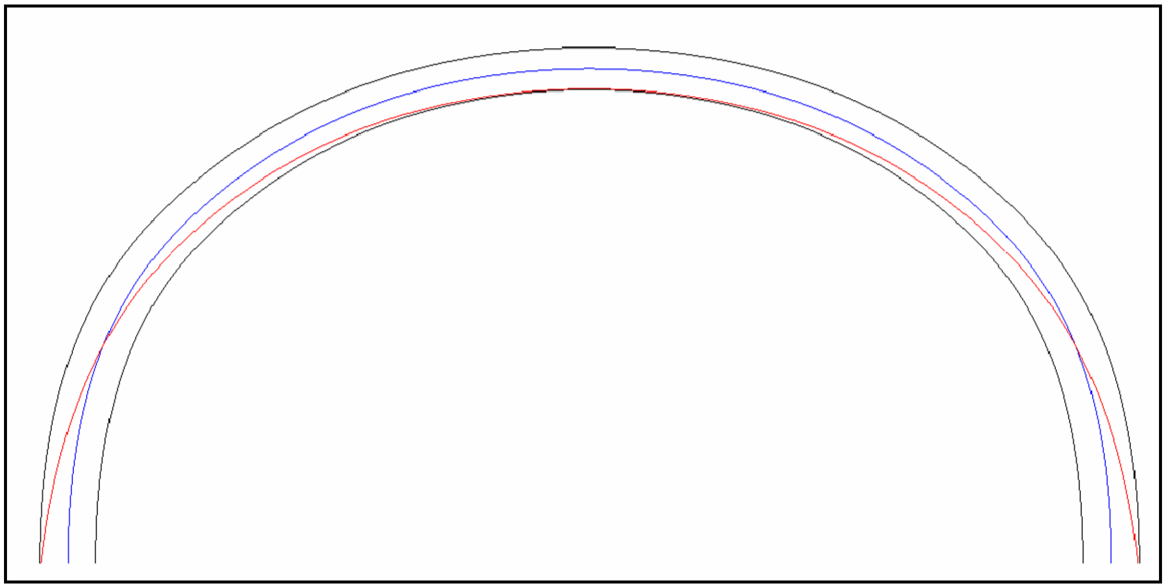

Figure 3-20 Spline that has been flattened. Spline is spread out and moved down by algorithm to form red spline

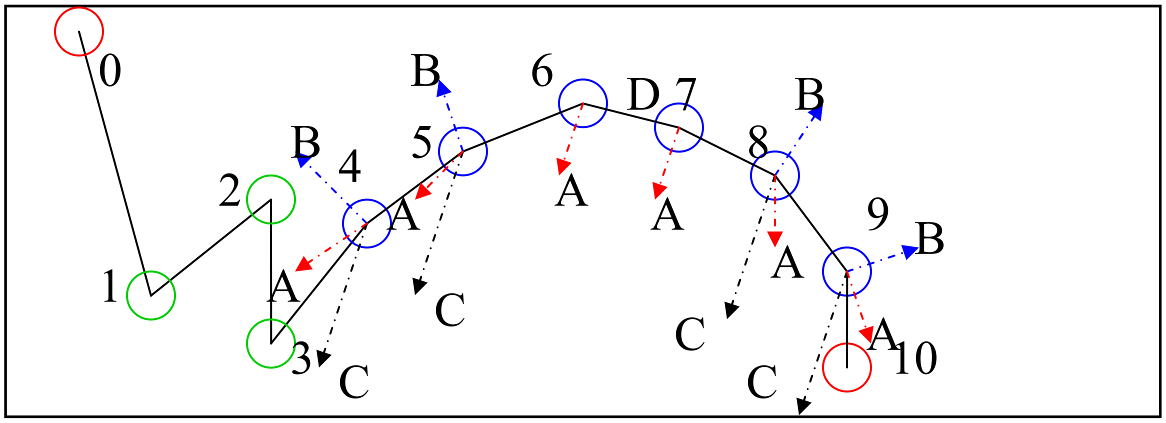

Figure 3-21 A (red) direction vectors for smooth point group (blue points). Preliminary vectors are B (blue) and C (black) vectors.

The apexes' direction vectors are calculated in one of two ways. If the apex is a single point then the direction vector is calculated using the same method as is used for a ripple point's direction vector using Equations (3-23), (3-24), and (3-25).

If it is a double apex then each of the points is assigned the same direction vector. A preliminary direction vector is calculated using Equation (3-26) that is perpendicular to the line that goes through both apex points (points 6 and 7) as illustrated by line D in Figure 3-21.

$$\mathbf{DV'}_i = \begin{bmatrix} X \\ Y \end{bmatrix}_{m-1} = \begin{bmatrix} \dfrac{-\Delta Y}{\sqrt{\Delta X^2 + \Delta Y^2}} \\ \dfrac{\Delta X}{\sqrt{\Delta X^2 + \Delta Y^2}} \end{bmatrix} \tag{3-26}$$

where:

$\Delta X = X_{i+1} - X_i$

$\Delta Y = Y_{i+1} - Y_i$

Equation (3-26) does not guarantee that the direction of the apex vectors is correct. The correct direction is shown in Figure 3-21. 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 using Equation (3-27). Equation (3-26) assumes that this point is on the positive side of the line. If the distance to the line is negative, the preliminary direction vector is reversed using Equation (3-28).

$$D_i = \frac{a}{\sqrt{a^2+b^2}}X_i + \frac{b}{\sqrt{a^2+b^2}}Y_i + \frac{c}{\sqrt{a^2+b^2}} \qquad \textbf{(3-27)}$$

where:

$a = Y_i - Y_{i+1}$

$b = X_{i+1} - X_i$

$c = X_i Y_{i+1} - X_{i+1} Y_i$

Preliminary direction vectors (**B** vectors, blue, in Figure 3-21) are calculated for all non-apex points using Equation (3-29). This method is the same as for direction vectors for the ripple case, except that the direction vectors are reversed. Equation (3-29) uses the values from Equation (3-23) and Equation (3-24). The preliminary direction vector moves the control points out, widens the tool path. This effectively lowers the curvature.

$$\mathbf{DV}_i = \mathbf{DV}_{i+1} = \left(D_i / \left|D_i\right|\right)\mathbf{DV'}_i \tag{3-28}$$

$$\mathbf{DV}_i^1 = -\left|\mathbf{A}_i + \mathbf{B}_i\right| \tag{3-29}$$

A scaled vector is created at each non-apex point by scaling the apex direction vector by 9.0 using Equation (3-30). If the preliminary vector is too large then it can cause high-curvature sections where the smooth group endpoints connect with adjacent control points. Therefore, the scaled vector is scaled to outweigh the preliminary vector when they are summed. The value of 9.0 was chosen through trial and error. It produces the best overall results. The scaled vector aligns the movement of the non-apex points with the apex points, so that the curve stays intact as it flattens. The normalized sum of the preliminary vector and scaled vector in Equation (3-31) is the direction vector for each of the smooth points.

$$\mathbf{DV}_i^2 = 9.0\mathbf{DV}_i^{apex} \tag{3-30}$$

$$\mathbf{DV}_i = \left|\mathbf{DV}_i^1 + \mathbf{DV}_i^2\right| \tag{3-31}$$

By summing the two vectors together and normalizing the result, the final direction vector obtains attributes from both of the vectors. The curve is flattened and lengthened by the preliminary vector but shortened by the scaled vector. 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 Figure 3-20.

The null vector case mentioned in the end and ripple point sections does not apply to smooth points. If any of the control points in a smooth group becomes part of a straight line then those control points are re-categorized as ripple points in the next iteration.

3.7 Sensitivity Values

Several iterations of control point movement are required to smooth the curve without crossing the tolerance limits. The algorithm uses two key variables to calculate the move distance for each control point in each iteration: the sensitivity value and the maximum allowable move distance. 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.

Sensitivity values are initialized before starting the first iteration by performing a mock iteration using the original control polygon and the radius of the tolerance tube as the maximum allowable distance to calculate the starting sensitivity values. After the sensitivity values are calculated, the control polygon is returned to its original state and the first iteration is executed.

To determine how far the spline deviates as a result of moving the control points during a smoothing iteration, a difference spline is created by subtracting the previous iteration's control points from the current iteration's control points. See Equation (3-32).

The difference spline is a cubic B-spline that has the same knot vector and number of control points as the original spline.

$$\mathbf{D}_i = \mathbf{P}_i^{smoothed} - \mathbf{P}_i^{original} \tag{3-32}$$

A parameter value is assigned to each control point in the spline. The first control point is assigned the parameter value equal to the first knot value. In other words, the first control point is assigned the beginning of the spline. The second point is assigned the parameter value halfway between the first and second knot values. If the first knot is 1 and the second knot is 2, then the second control point's parameter value is 1.5. The third control point is assigned the second knot value. The subsequent control points are assigned the same way, except the last and second-to-last control points. The second-to-last control point is assigned the parameter value halfway between the last and the second-to-last knot values. The last control point is assigned the last knot value. Equations (3-33) through (3-37) calculate the parameter value for each control point. The parameter value is t. The knot value is k. The number of control points is m and the number of knots is n.

$$t_0 = k_0 \tag{3-33}$$

$$t_1 = \frac{k_0 + k_1}{2} \tag{3-34}$$

$$t_i = k_{i-1} \tag{3-35}$$

$$t_{m-2} = \frac{k_{n-2} + k_{n-1}}{2} \tag{3-36}$$

$$t_{m-1} = k_{n-1} \tag{3-37}$$

The sensitivity value is calculated in Equation (3-38) by dividing the difference spline's value at each control point's corresponding parameter value by the control point's move distance.

$$S_i = D_i / MD_i \tag{3-38}$$

3.8 Maximum Distance

The maximum allowable distance in each iteration is the minimum distance between the spline and the path tolerance. For the first iteration, the maximum allowable distance is initialized to be the radius of the path tolerance tube.

The maximum allowable distance is calculated as the difference between the radius of the tolerance tube and the maximum distance between the original spline and the current iteration spline. See Figure 3-22. To calculate the maximum distance, a difference spline that is similar to the sensitivity difference spline is created by subtracting the original polygon from the current iteration polygon. This difference spline represents the distance that the spline has moved from the original position. See

Equation (3-39). Figure 3-23 shows a spline with the maximum distance location indicated.

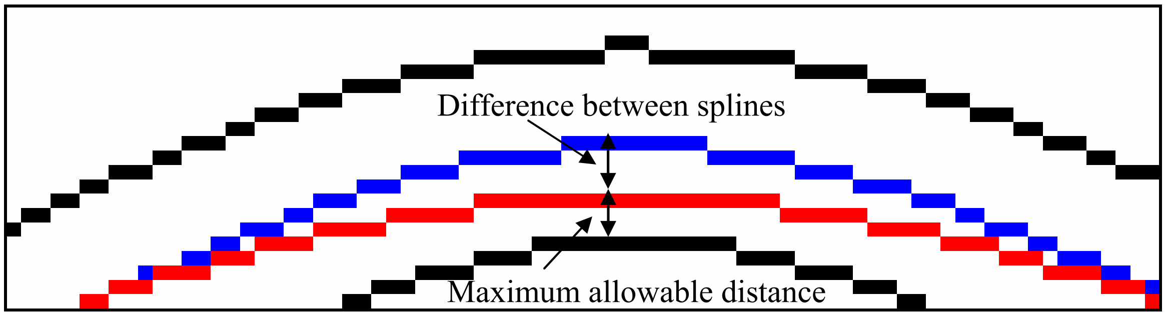

Figure 3-22 Calculating maximum allowable distance

$$D_i^{allowable} = \left(R - \left|\mathbf{P}_i^{original} - \mathbf{P}_i^{current}\right|\right)_{\min} \qquad \textbf{(3-39)}$$

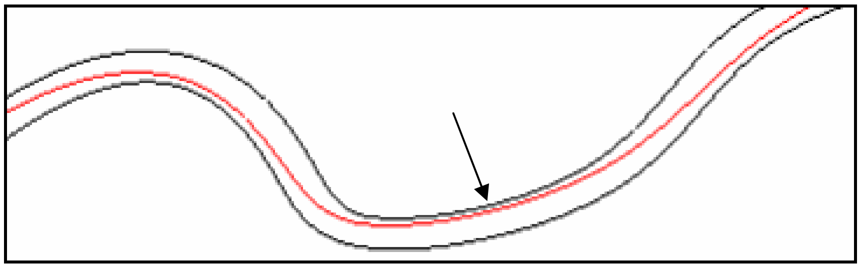

Figure 3-23 Maximum allowable distance on spline

Note that Equation (3-39) will error on the safe side. The radius of the tolerance tube is the minimum distance between the original spline and the tolerance tube. If a point on the spline moves in any direction other than perpendicular to the original spline then it will actually be further from the tolerance tube than Equation (3-39) calculates. Figure 3-24 shows a line at an angle that does not reach the tolerance tube. Lines **A** and **B** have the same length but **B** does not reach the tolerance tube.

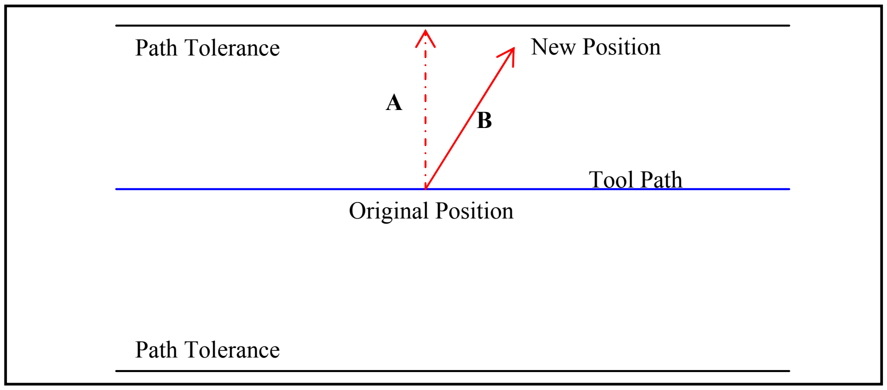

Figure 3-24 Point on spline that does not move perpendicular to original spline will not reach tolerance tube. Most points move very close to perpendicular, so this inaccuracy is minimal.

To find the minimum distance, each knot interval is split into a number of steps as determined by the precision and the distance is evaluated at each of these divisions. This is equivalent to splitting the B-spline into its Bezier curves and splitting up each of the Bezier curves.

3.9 Move Distance

A move distance is the distance that a control point moves during each iteration. To calculate the move distance, each sensitivity value is inverted and then scaled from zero to one by dividing the inverted sensitivity value by the maximum inverted sensitivity value. See Equation (3-40).

The smallest sensitivity value before inversion becomes a 1.0 through inversion and scaling. The largest sensitivity value before inversion becomes the smallest value but never becomes 0.0. Equation (3-40) ensures that each move distance is non-zero. If the largest sensitivity value were scaled to a 0.0 move distance, the control point with the largest sensitivity value would never move. The scaled inverted sensitivity value is multiplied by the maximum allowable distance to produce the move distance value for each control point. Equation (3-41) scales all of the move distances so that all of the control points will move the spline values the same distance.

$$S_i^{inv} = \frac{\left(1/S_i \right)}{\left(1/S_i \right)_{\max}} \quad \textbf{range } (0.0, 1.0] \tag{3-40}$$

$$MD_i = D_{\max} * S_i^{inv} \tag{3-41}$$

The example below uses a control polygon with five control points to demonstrate how move distances are calculated. The initial sensitivities were calculated in the previous iteration.

Initial Sensitivities:

$$\begin{bmatrix} \frac{1}{5} & \frac{2}{5} & \frac{1}{2} & \frac{4}{5} & 1 \end{bmatrix} = \begin{bmatrix} 20\% & 40\% & 50\% & 80\% & 100\% \end{bmatrix} \tag{3-42}$$

The sensitivities are inverted and the smallest sensitivity is the largest value.

Sensitivity Values:

$$\begin{bmatrix} \frac{1}{1/5} & \frac{1}{2/5} & \frac{1}{1/2} & \frac{1}{4/5} & \frac{1}{1} \end{bmatrix} = \begin{bmatrix} 5 & \frac{5}{2} & 2 & \frac{5}{4} & 1 \end{bmatrix} \tag{3-43}$$

The inverted sensitivities are scaled by dividing them by the maximum inverted sensitivity (5). The new scale is $(0.0, 1.0]$

Scaled Inverted Sensitivities:

$$\begin{bmatrix} 5 & \frac{5}{2} & 2 & \frac{5}{4} & 1 \end{bmatrix} \Big/ 5 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{2}{5} & \frac{1}{4} & \frac{1}{5} \end{bmatrix} \tag{3-44}$$

The maximum allowable distance was set to 10. The scaled values are multiplied by the maximum allowable distance to become the move distances.

Move Distances:

$$10 * \begin{bmatrix} 1 & \frac{1}{2} & \frac{2}{5} & \frac{1}{4} & \frac{1}{5} \end{bmatrix} = \begin{bmatrix} 10 & 5 & 4 & \frac{5}{2} & 2 \end{bmatrix} \tag{3-45}$$

The actual distances moved are simulated in this example by multiplying the move distances by the sensitivity values. This is close to what would happen during the smoothing section that is discussed in Section 3.10. The sensitivity values do not match the actual movements of the spline because the sensitivities are linear and the spline is cubic, but they are a close approximation. It is important to note that all of the points moved the same distance due to the scaling, which causes the spline to smooth evenly.

When control points form part of a straight line with two or more adjacent control points, they are initially assigned a 0.0 sensitivity value, because the spline is already in its ideal position. These 0.0 sensitivity values are special cases that are eliminated by setting the sensitivity value to 1.0 so that it does not cause a division error during scaling.

Actual Distances Moved:

$$\begin{bmatrix} 10*\frac{1}{5} & 5*\frac{2}{5} & 4*\frac{1}{2} & \frac{5}{2}*\frac{4}{5} & 2*1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \end{bmatrix} \tag{3-46}$$

3.10 Smooth Control Polygon

After all of the direction vectors and distances are determined, the polygon is smoothed. Smoothing is accomplished by moving each control point its move distance in the direction of its direction vector. This movement is calculated by multiplying the move distance by the direction vector and adding this result to the control point's current position as shown in Equation (3-47). The resulting control polygon is used to calculate the distances and sensitivities for the next iteration.

$$\mathbf{P}_i^{new} = \mathbf{P}_i^{old} + MD_i * \mathbf{DV}_i \tag{3-47}$$

3.11 Termination

The iterations terminate when the spline comes within the predefined precision of the tolerance tube. At this point, the spline is close enough to the path tolerance to consider it as touching the tolerance band. Upon termination, the previous iteration control polygon is used as the solution. The current iteration control polygon is not used because the spline has passed the tolerance band.

3.12 Example of Reduction in Curvature

To illustrate the effects of reducing curvature in high-curvature tool paths, eight similar tool paths were created that all have a single ripple. The peak curvature value for each tool path has a higher value than the previous one. The curvature values range from 1/15 mm^{-1} to 8/3 mm^{-1}. The tool paths were smoothed with the algorithm using multiple path tolerance radii. The original (blue) and smoothed (red) tool paths are shown inside of a path tolerance (black) in Figure 3-25.

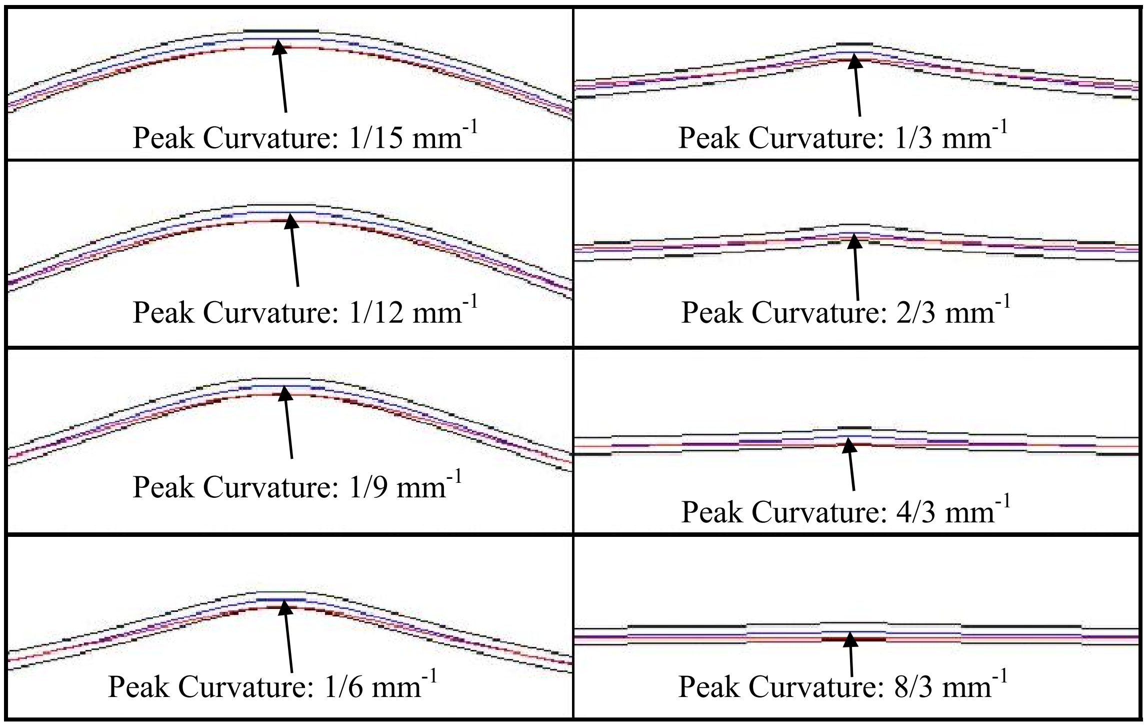
Peak Curvature: 1/15 mm^{-1}
Peak Curvature: 1/3 mm^{-1}
Peak Curvature: 1/12 mm^{-1}
Peak Curvature: 2/3 mm^{-1}
Peak Curvature: 1/9 mm^{-1}
Peak Curvature: 4/3 mm^{-1}
Peak Curvature: 1/6 mm^{-1}
Peak Curvature: 8/3 mm^{-1}

Figure 3-25 Eight tool paths with different severity of curvature

Figure 3-26 shows the reduction in curvature for each tool path at different path tolerance radii. At a given path tolerance, the percent reduction in curvature is greater for tool paths with more severe curvature. As can be seen, the effectiveness of the algorithm increases with the severity of the curvature. For example, with a path tolerance of 0.125 mm, the tool path with an initial curvature of 8/3 mm^{-1} shows a much higher percent improvement in curvature than the tool path with an initial curvature of 1/15 mm^{-1}.

When the curvature improvement reaches 100%, the spline becomes a straight line with zero curvature.

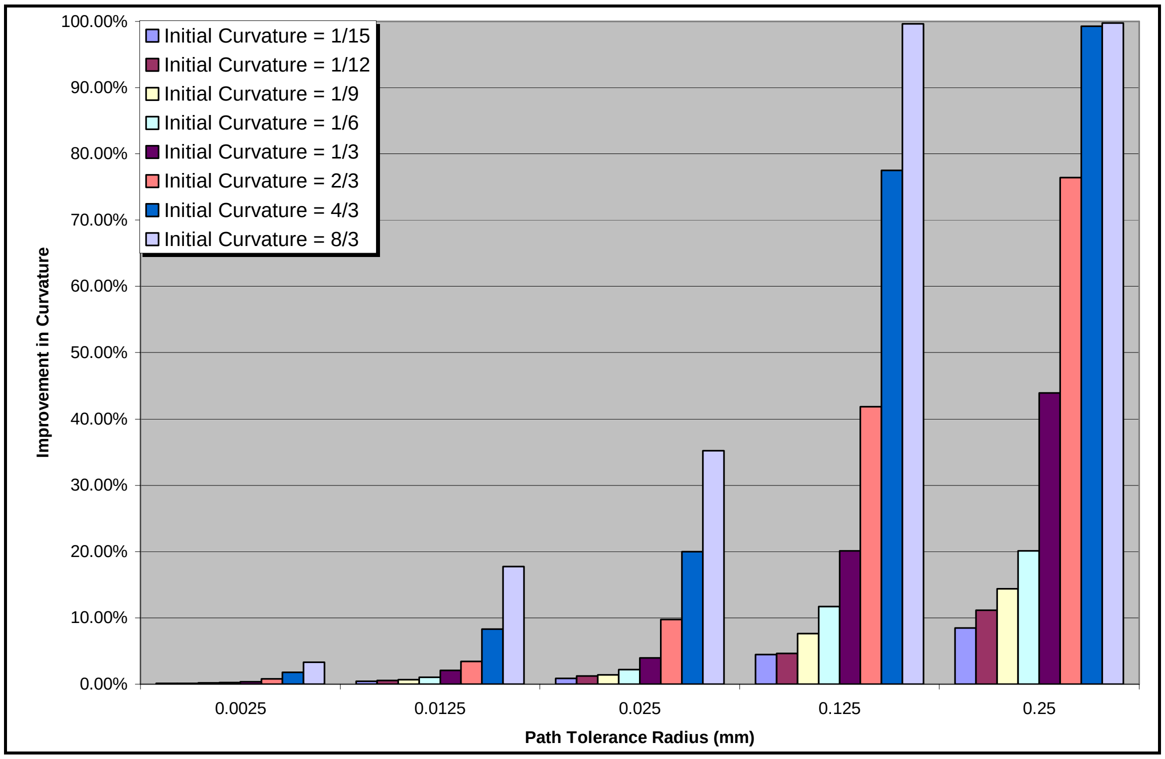

Figure 3-26 Percent reduction in curvature for tool paths with single high-curvature ripple.

Figure 3-27 shows the reduction in machining time for each tool path at different path tolerance radii. At a given path tolerance, the percent reduction in machining time is greater for tool paths with more severe curvature. As can be seen, the effectiveness of the algorithm increases with the severity of the curvature. For example, with a path tolerance of 0.125 mm, the tool path with an initial curvature of 8/3 mm^{-1} shows a much higher percent improvement in machining time than the tool path with an initial curvature of 1/15 mm^{-1}.

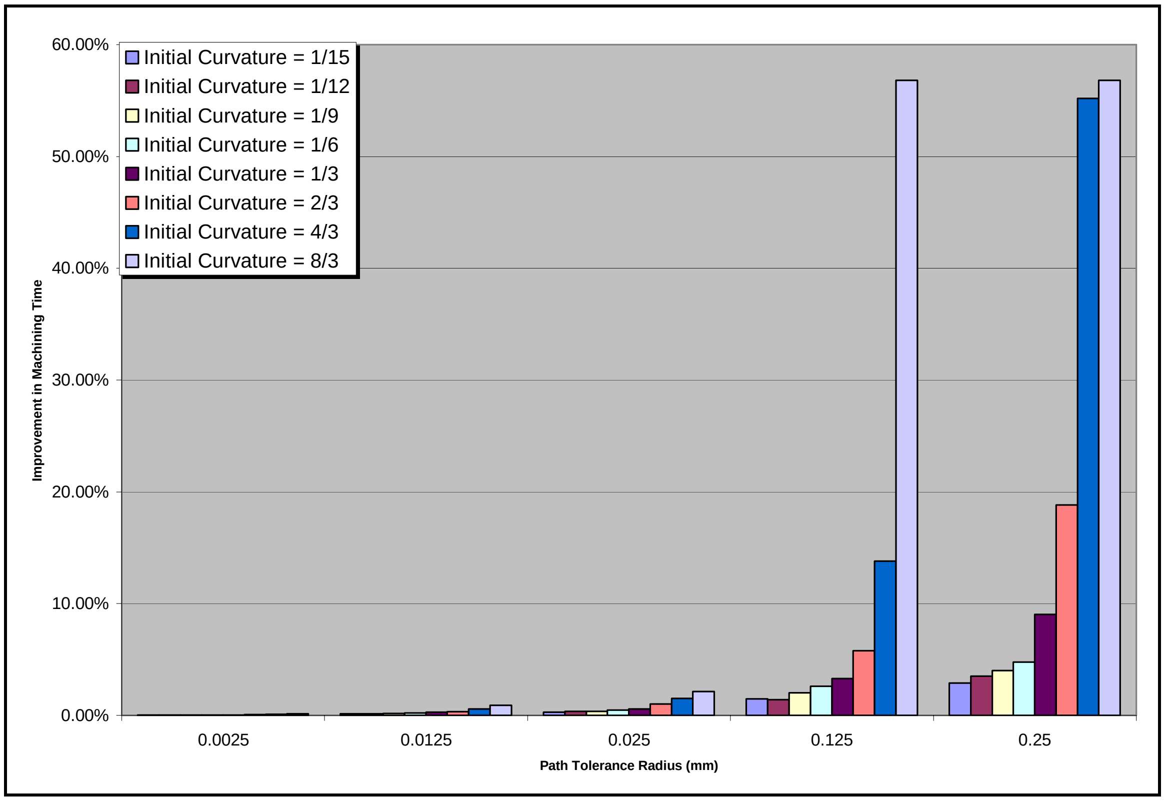

Figure 3-27 Percent reduction in machining time for tool paths with single high-curvature ripple.

4 Results

4.1 Sample Tool Paths

Six sample tool paths were created to test the smoothing algorithm. The tool paths represent a single pass across a surface because this will sufficiently test the algorithm and keep the amount of data to a minimum. It is not necessary to zigzag across the part multiple times because all of the passes will be similar and not produce additional unique test cases. Each tool path is composed of a single B-spline and contains a different degree of high-curvature ripples.

The first tool path is shown in Figure 4-1 and represents a tool path with low curvature radii. This tool path is simple and smooth. It measures 766 mm in length and the maximum curvature is 0.175 mm^{-1}. The tool feedrate should be close to the commanded value.

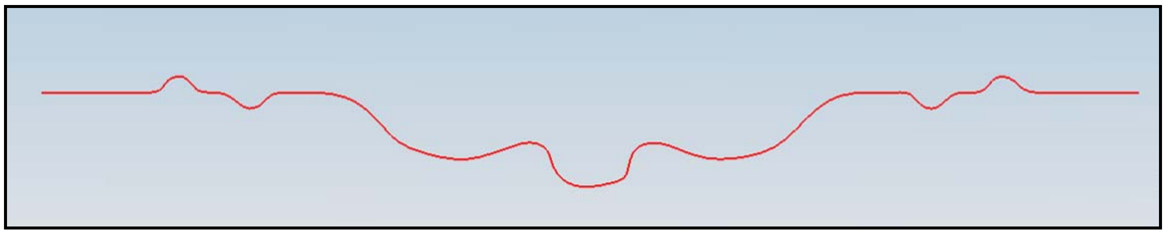

Figure 4-1 Test Case 1: Tool path with many low curvature radii. Tool path is fairly smooth and long. NC controller will decrease the feedrate through these low curvature sections, but not significantly.

The second tool path is shown in Figure 4-2 and mainly has low curvature radii with just two high-curvature radii. The part is 376 mm long and has a maximum curvature of 9.0 mm^{-1}. This tool path will slow down the feedrate significantly due to the two small radii.

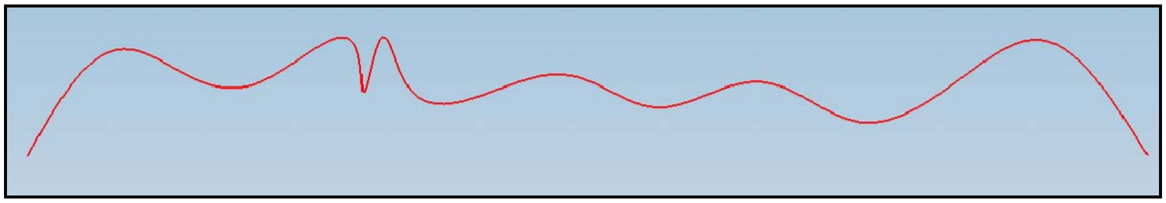

Figure 4-2 Test Case 2: Tool path with many low curvature radii and two high-curvature radii. Ttool path is fairly smooth and of medium length, except for two high-curvature radii. NC controller will decrease feedrate through these curvature sections, especially at high-curvature ripples.

The tool path for test case 3 is complex with multiple high-curvature ripples and is shown Figure 4-3. The path is 210 mm long and has a maximum curvature of 28 mm^{-1}. This tool path represents an extreme case in which the feedrate will be very slow.

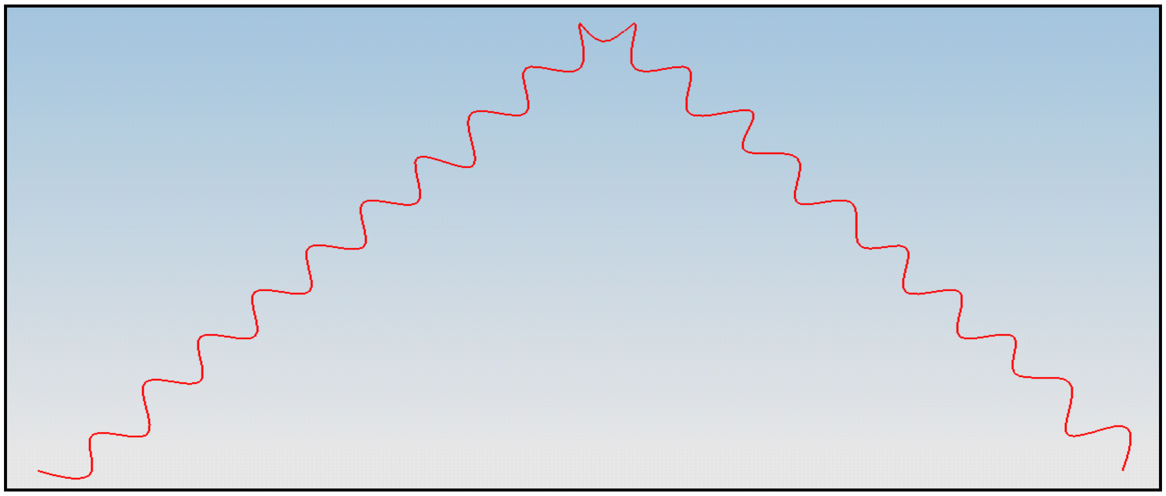

Figure 4-3 Test Case 3: Tool path with many high-curvature radii. Tool path is very complex and short. NC controller will decrease feedrate drastically.

The tool path shown in Figure 4-4 is generated for factory siping on a snow tire mold with complex geometry. Siping creates the little slits in the tread of the tire. A snow tire with factory siping can be seen in Figure 4-5.

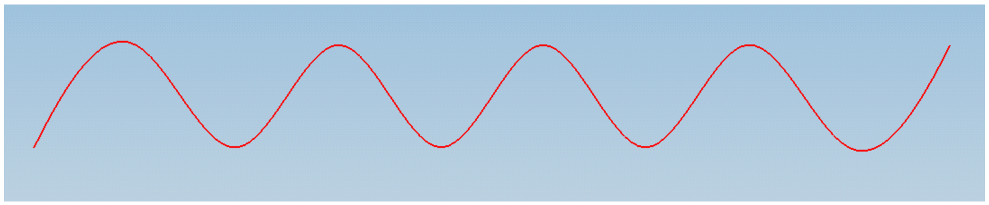

Figure 4-4 Test Case 4: Tool path generated along offset of the complex surface of a snow tire. It contains multiple sections of high-curvature which will drastically decrease feedrate.

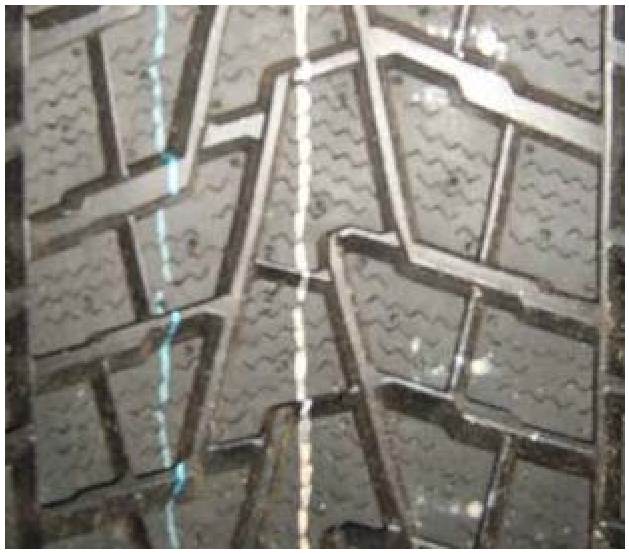

Figure 4-5 Snow tire with siping and very complex geometry

The next test case in Figure 4-6 was designed as a straight line with zero curvature but two of the data points used to interpolate the tool path were noisy. These position discontinuous 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. This increased curvature significantly lowers the feedrate.

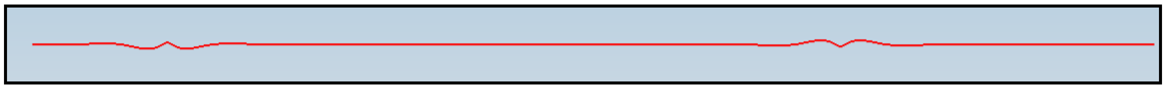

Figure 4-6 Test Case 5: Linear tool path of ideally zero curvature with two noisy points. Points are both 0.2 mm away from straight line and form ripples that almost reach 0.4 mm from straight line. These noisy points create sections of high-curvature ripples. The NC tool will decrease feedrate from ideal to maneuver high-curvature ripples.

The final test case tool path illustrates how tangency discontinuities form high-curvature ripples in the tool paths and can be seen in Figure 4-7. The tool path tries to follow a series of 90-degree corners, but is unable to do so without creating high-curvature ripples. The ripples form because a cubic B-spline is unable to represent a tangency discontinuity.

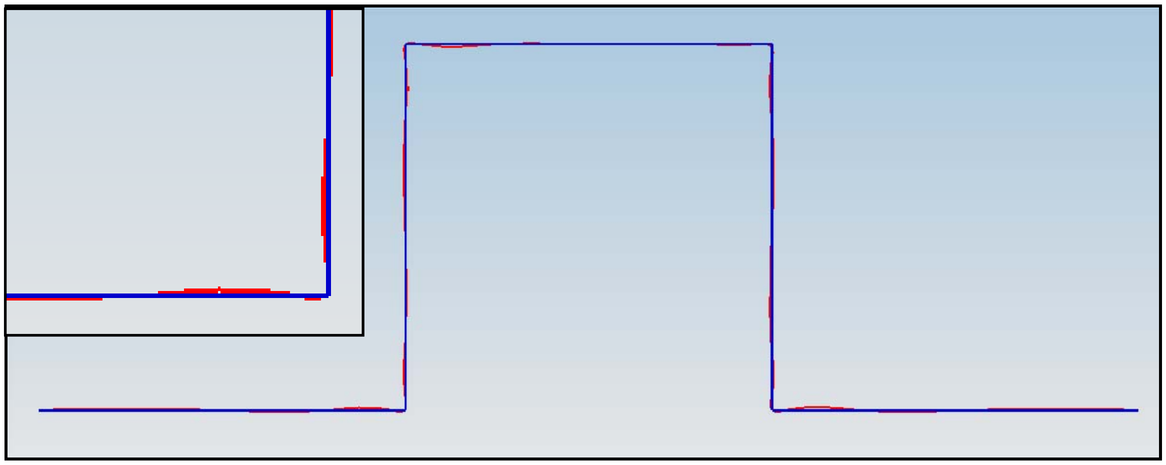

Figure 4-7 Test Case 6: Aool path with 90-degree corners. Corners are tangency discontinuous and they cause severe ripples in tool path. NC tool will decrease feedrate from ideal value to maneuver high-curvature ripples.

It is important to note that a CAM program may recognize intentional design discontinuities and split the tool path into multiple sections connected with tangency discontinuities. However, this test case represents tangency discontinuities that are caused by anomalies and not the designer. The CAM package will not recognize anomalies and will create ripples in the tool path instead of splitting it into multiple sections.

4.2 Smoothing Parameters

The smoothing algorithm required the following user-defined 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 was set to 5 causing each Bezier in the B-spline to be split into 5 sections in the maximum distance function. Each of the tool paths was smoothed three separate times using different path tolerances: 0.025 mm, 0.125 mm, and 0.25 mm.

4.3 Smoothed Tool Path Results

The smoothed tool paths and the tolerance band are shown for a path tolerance of 0.25 mm in Figure 4-8 through Figure 4-13. The figures are close-ups of sections with high curvature that show the tolerance band and the smooth path. The figures show that the smooth paths flatten around all of the corners. The tool paths are red and the path tolerance tube is black.

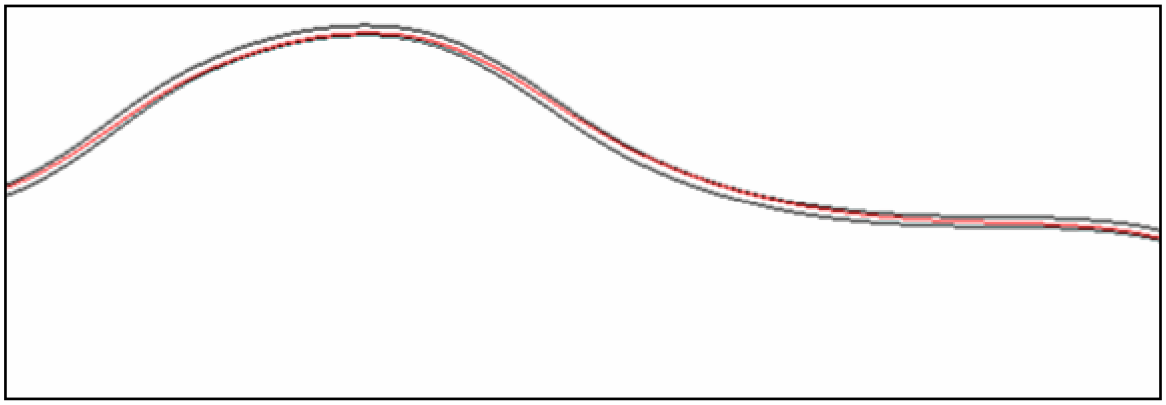

Figure 4-8 Test Case 1: Smoothed path for tool path in Figure 4-1

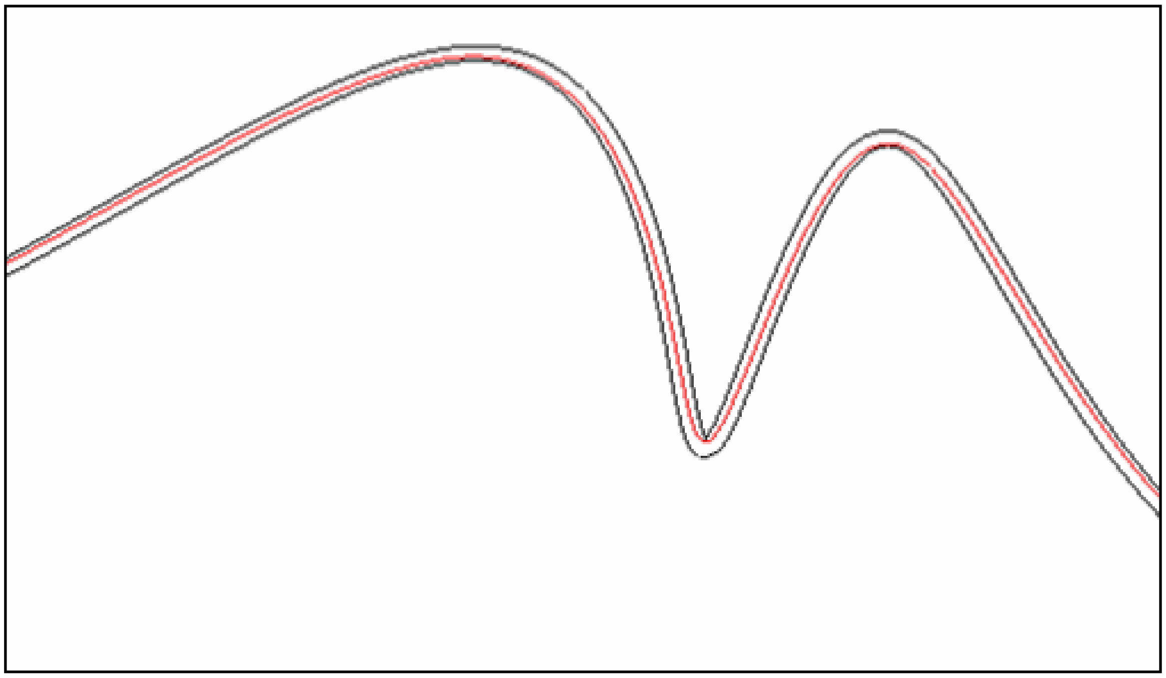

Figure 4-9 Test Case 2: Smoothed path for tool path in Figure 4-2

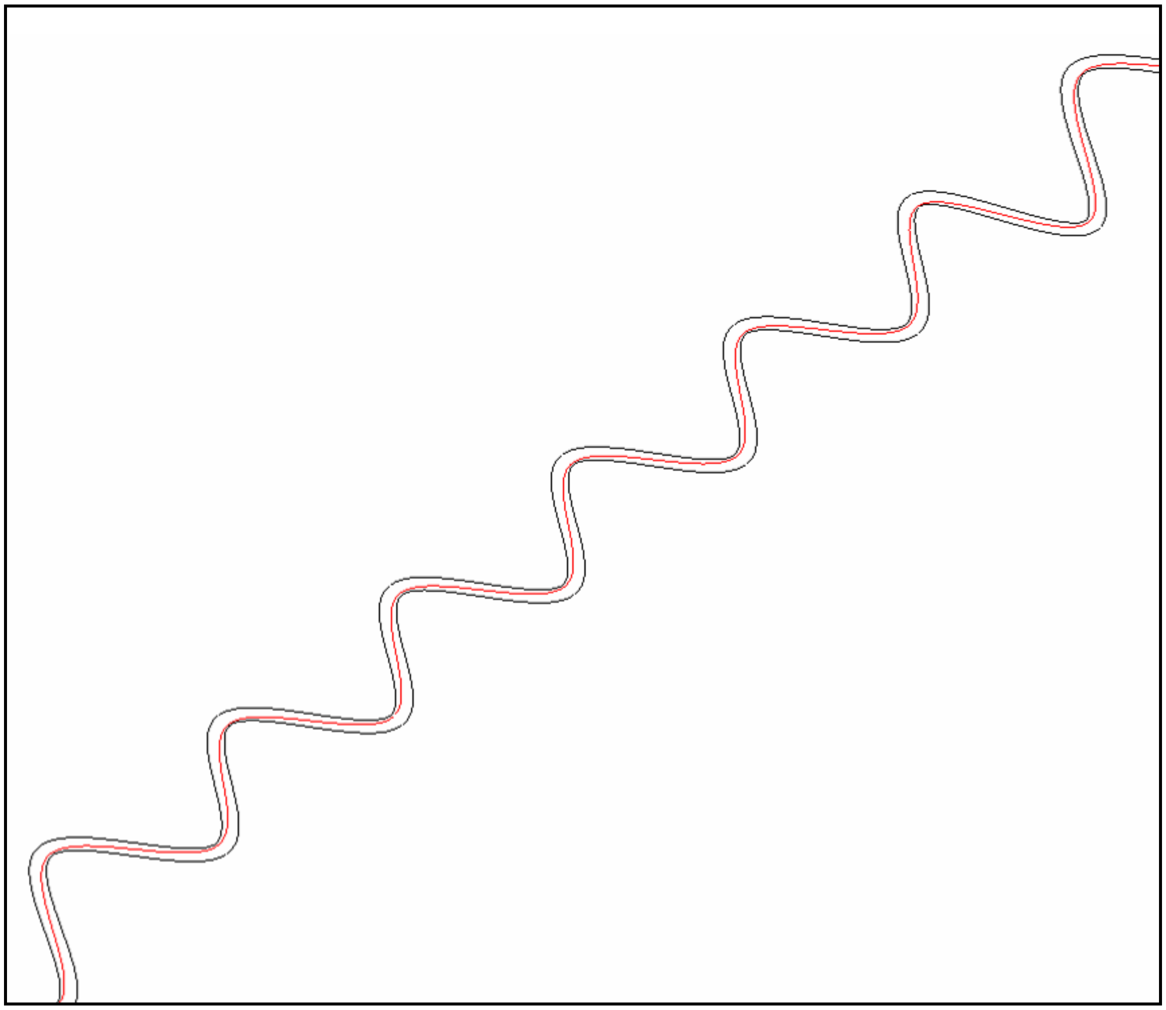

Figure 4-10 Test Case 3: Smoothed path for tool path in Figure 4-3

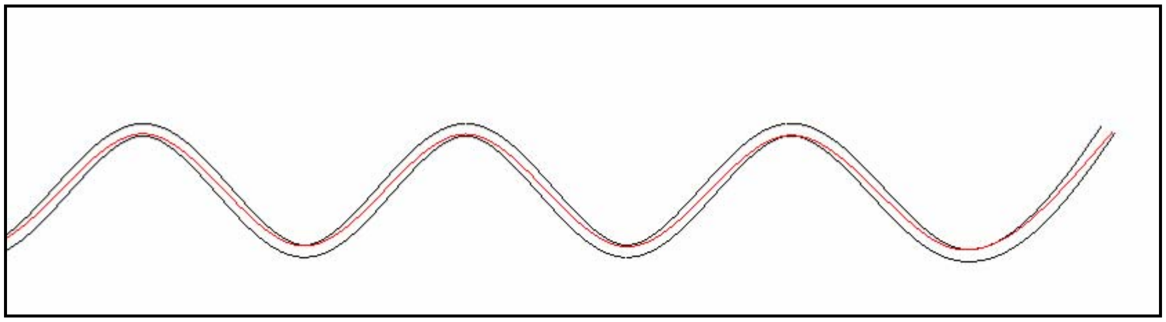

Figure 4-11 Test Case 4: Smoothed path for tool path in Figure 4-4

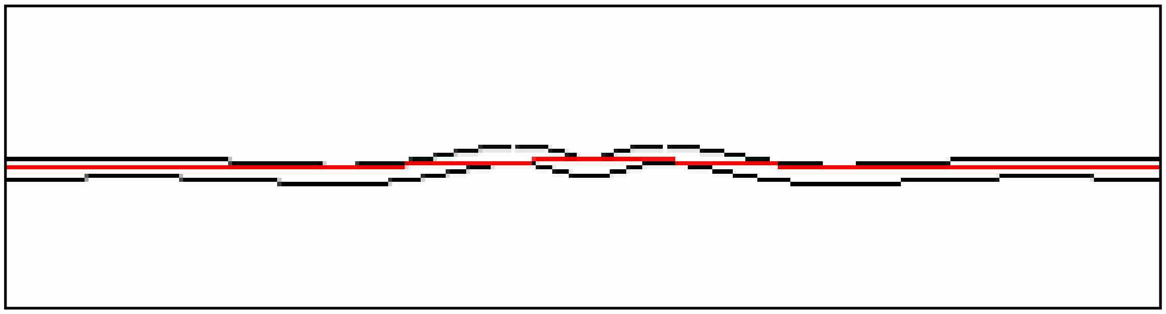

Figure 4-12 Test Case 5: Smoothed path for tool path in Figure 4-6

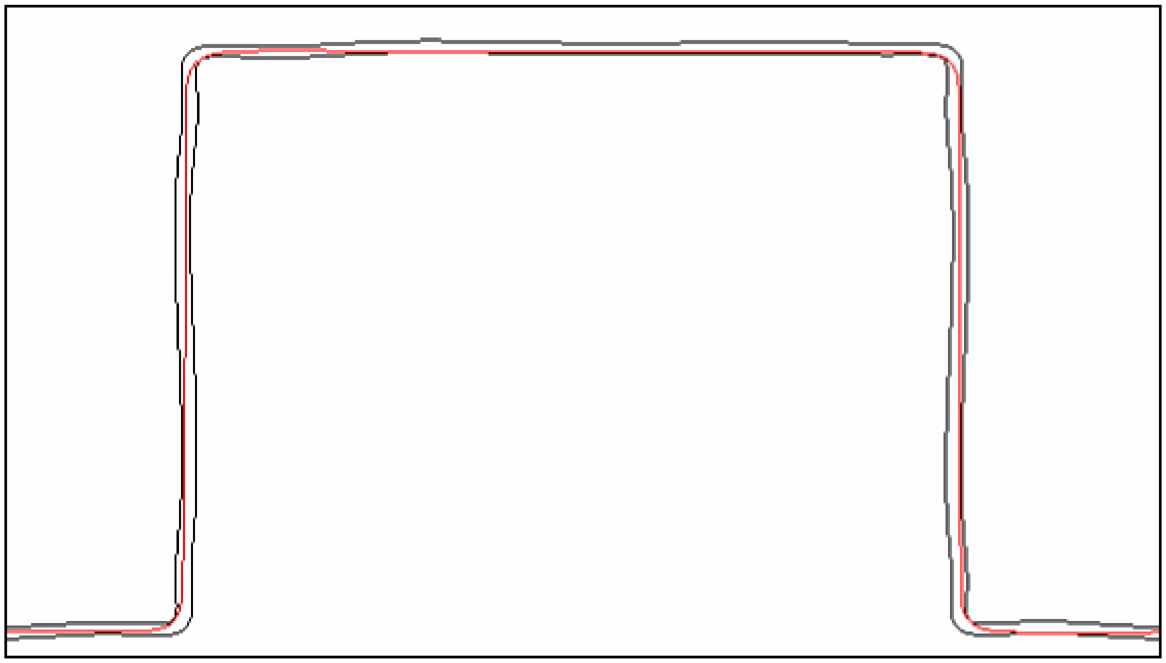

Figure 4-13 Test Case 6: Smoothed path for tool path in Figure 4-7

4.4 Curvature Results

The curvature along each of the tool paths is recorded in Figure 4-14 through Figure 4-19 and in Table 4-1. The curvature graphs and table show that the high-curvature ripples decrease with the smoothing algorithm and the tool path becomes increasingly smoother with a larger path tolerance radius.

Table 4-1 Relative curvature measurements from before and after smoothing algorithm

Part	Path Tolerance (mm)	Curvature Average (1/mm)	Curvature Difference (1/mm)	Curvature 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%
Test Case 6: Square	Original	0.175172	0	0.00%
	R = 0.025	0.168204	0.0069681	3.98%
	R = 0.125	0.143115	0.0320576	18.30%
	R = 0.250	0.131753	0.0434191	24.79%

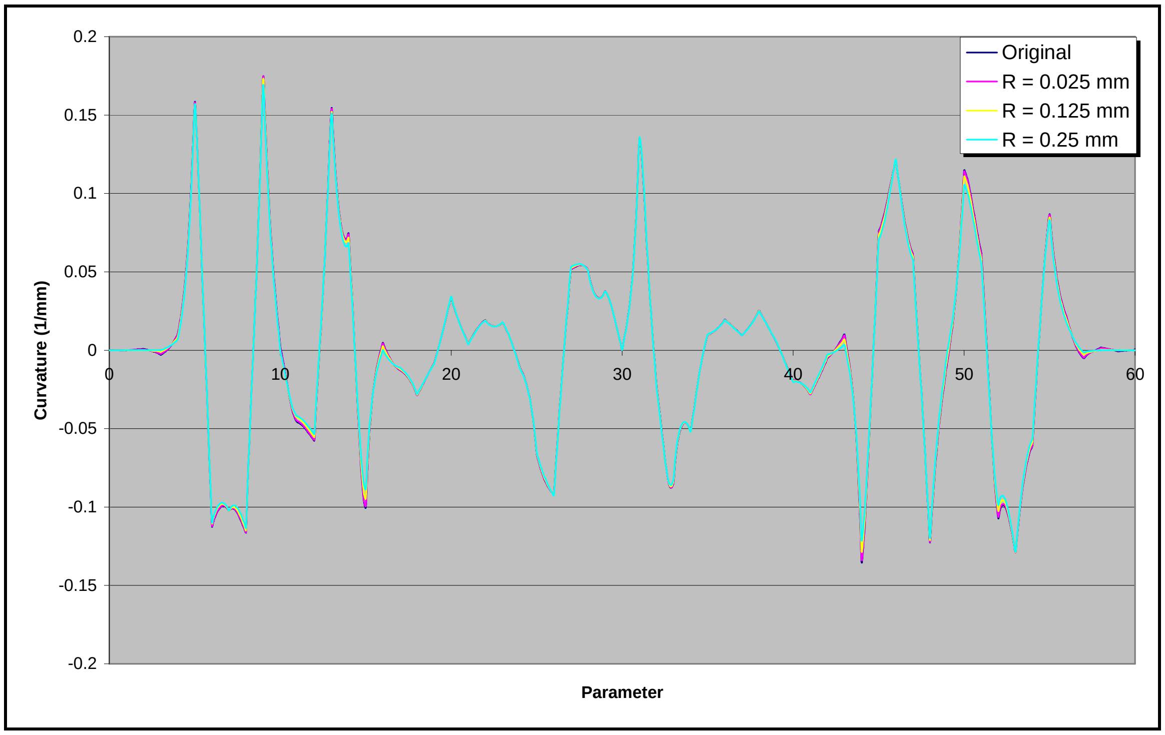

Figure 4-14 Test Case 1: Curvature for part in Figure 4-1

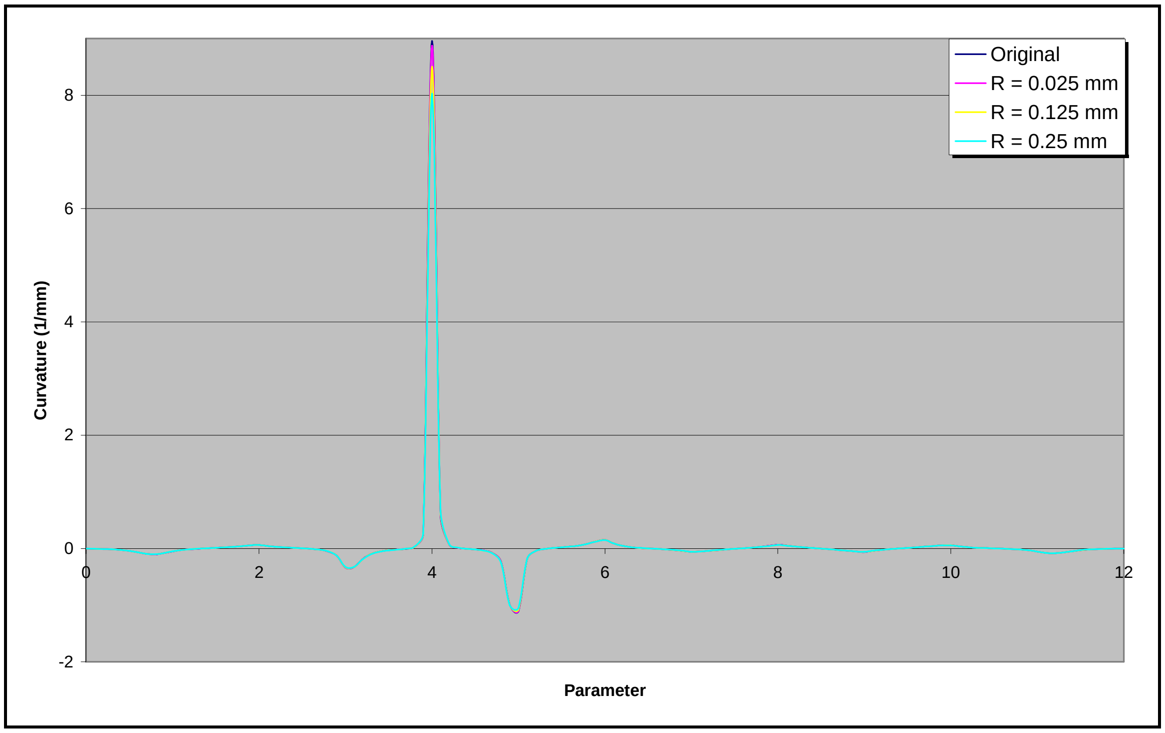

Figure 4-15 Test Case 2: Curvature for part in Figure 4-2

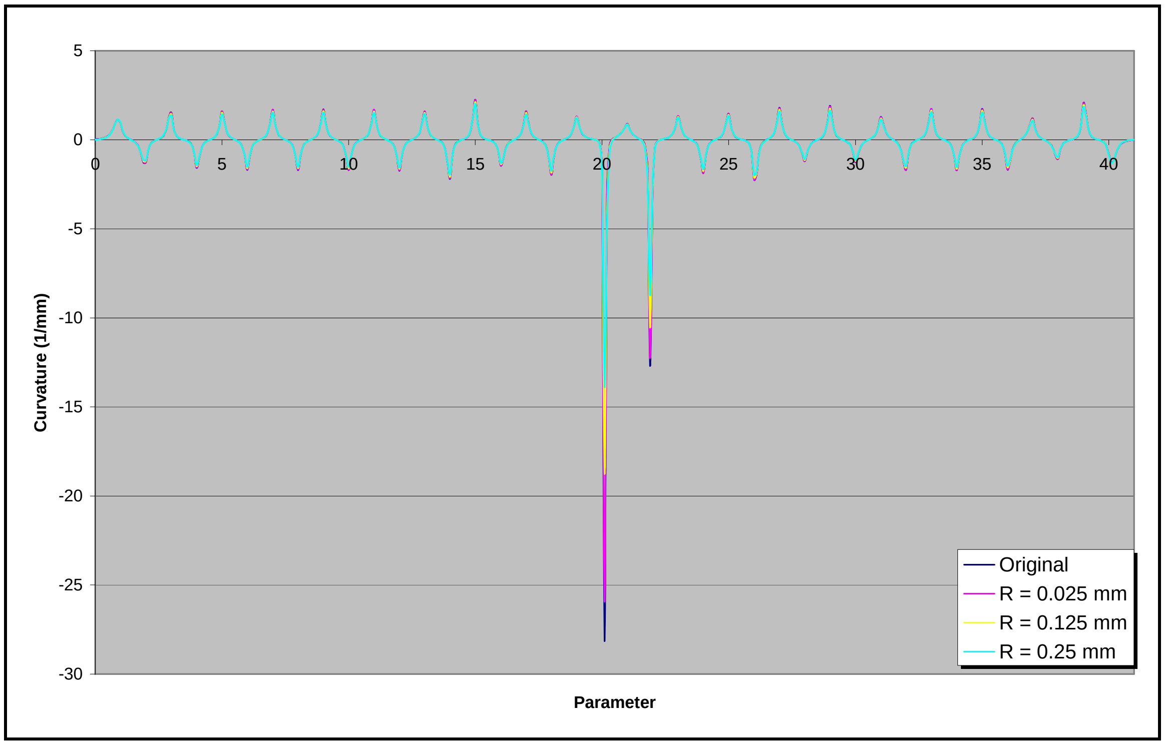

Figure 4-16 Test Case 3: Curvature for part in Figure 4-3

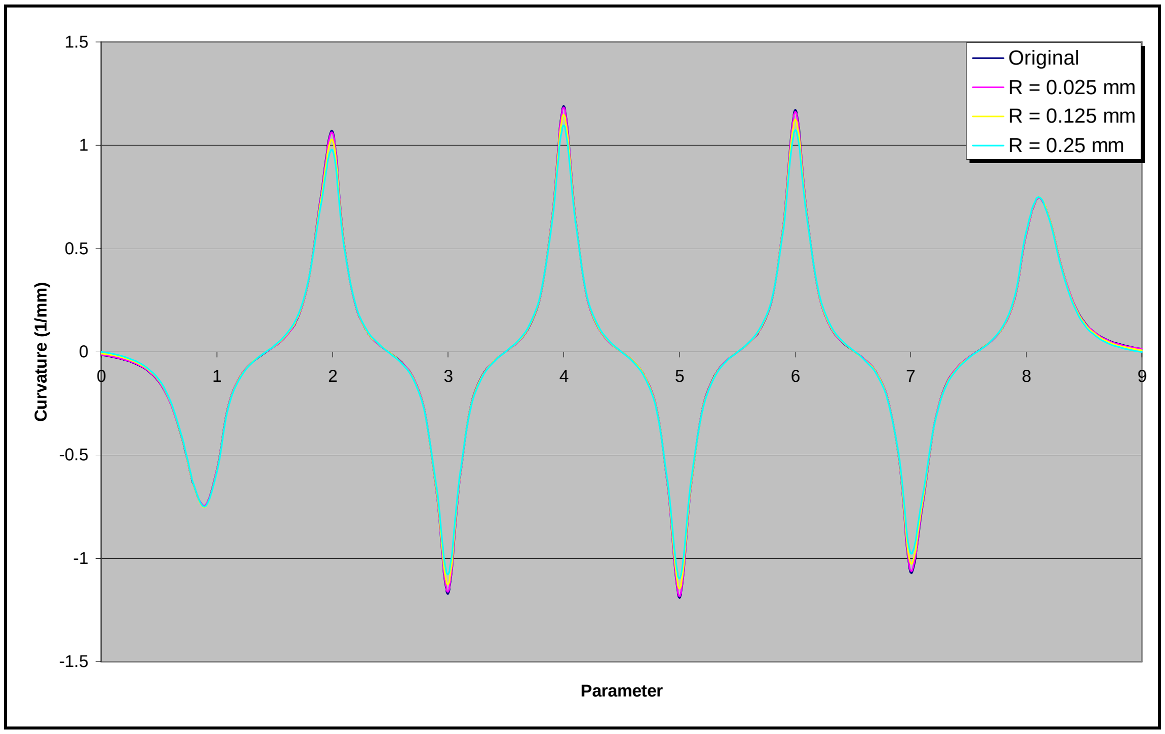

Figure 4-17 Test Case 4: Curvature for part in Figure 4-4

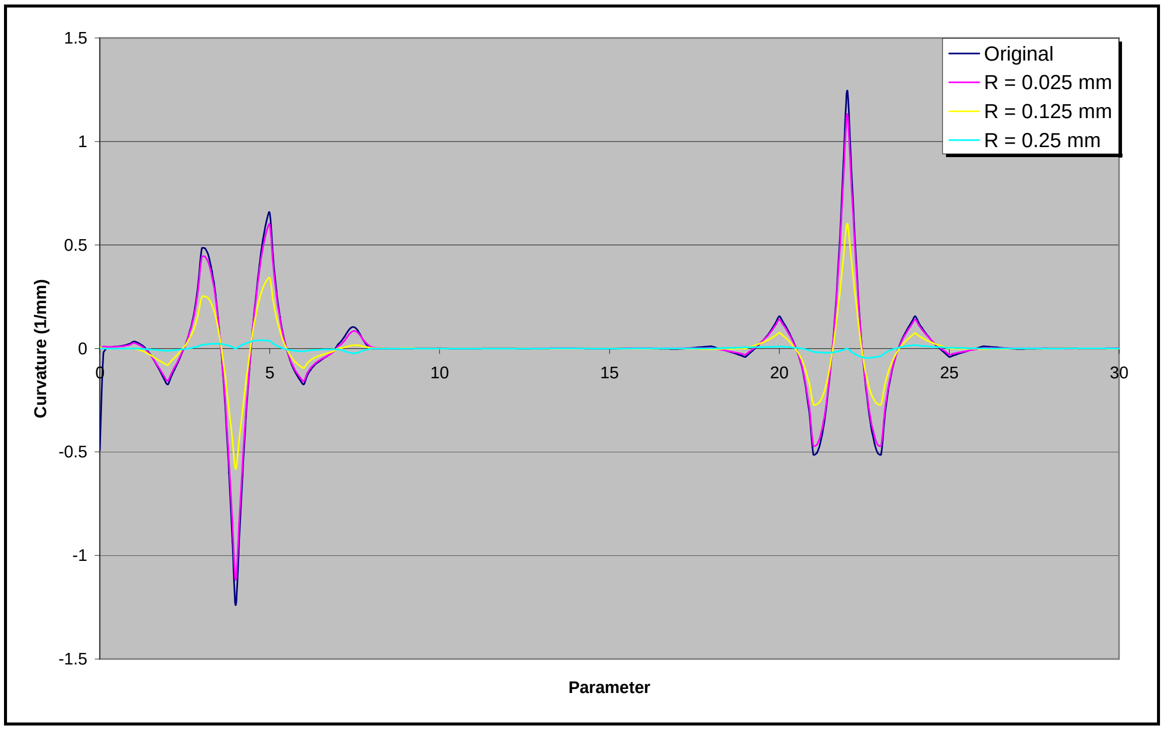

Figure 4-18 Test Case 5: Curvature for part in Figure 4-6

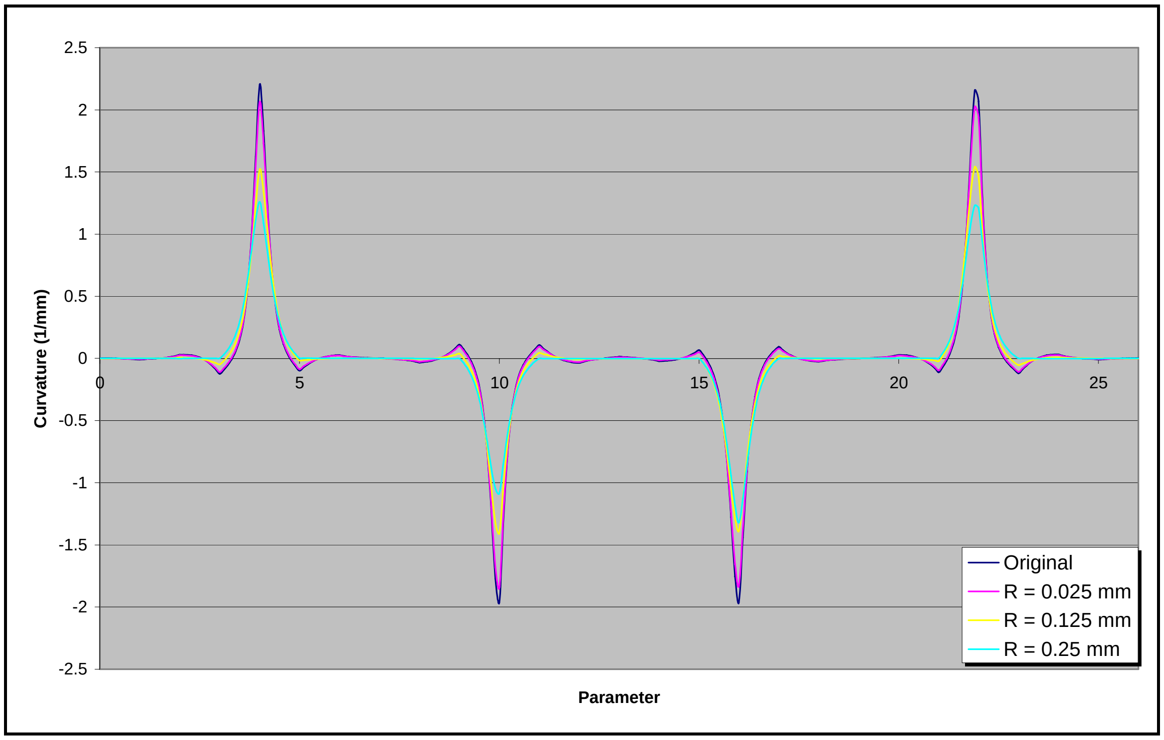

Figure 4-19 Test Case 6: Curvature for part in Figure 4-7

4.5 Machine Time Simulation

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 the values below for all tool paths.

Feedrate: $V_f = 400\, mm/s$

Acceleration: $A_m = 4000\, mm/s^2$

Jerk: $J_m = 20{,}000\, mm/s^3$

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 velocities at multiple distances along the tool path were calculated using Equation (4-1). The maximum and minimum velocities correspond to specific positions along the tool path.

$$V_{\text{feedrate}} = \sqrt{\frac{A_{\max}}{k_{\text{curvature}}}} \tag{4-1}$$

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. Below are the equations for the s-curve profiles.

The S-curve velocity profile uses constant jerk to derive equations of motion. Figure 4-20 shows an ideal s-curve velocity profile and the jerk profile that goes with it.

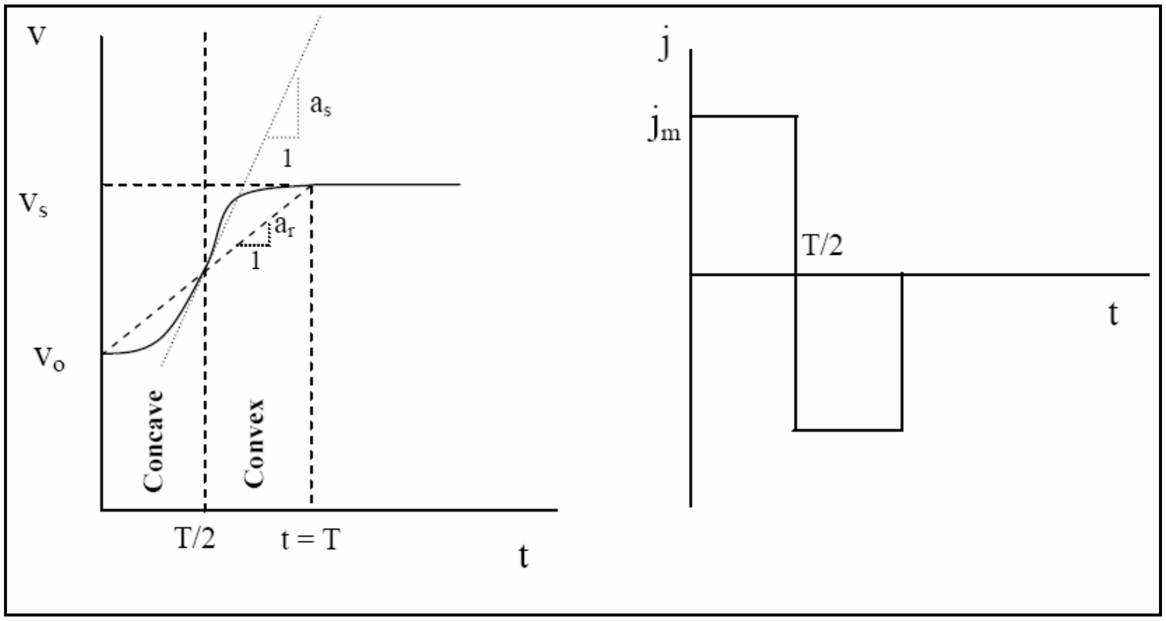

Figure 4-20 Constant jerk s-curve velocity

If the required change in velocity is small, the maximum acceleration will not be reached. and a_s is decreased. For more information on using S-curve velocity, see chapter 5 of Dr. Red's (2007) on-line course notes.

The equations of motion for the concave portion of the s-curve are:

$$s(t) = v_0 t + jt^3/6 \tag{4-2}$$

$$v(t) = v_0 + jt^2/2 \tag{4-3}$$

$$a(t) = jt \tag{4-4}$$

The equations of motion for the convex portion of the s-curve are:

$$s(t) = v_{1/2} t + a_s t^2/2 - jt^3/6 \tag{4-5}$$

$$v(t) = v_{1/2} + a_s t - jt^2/2 \tag{4-6}$$

$$a(t) = a_s - jt \tag{4-7}$$

$$v_{1/2} = \frac{v_0 + v_s}{2} \tag{4-8}$$

If the required change in velocity is large, the maximum acceleration will be reached and a section of constant acceleration is placed between the concave and convex portions of the profile as seen in Figure 4-21. The velocity continues to increase without going above the acceleration limit in this linear portion.

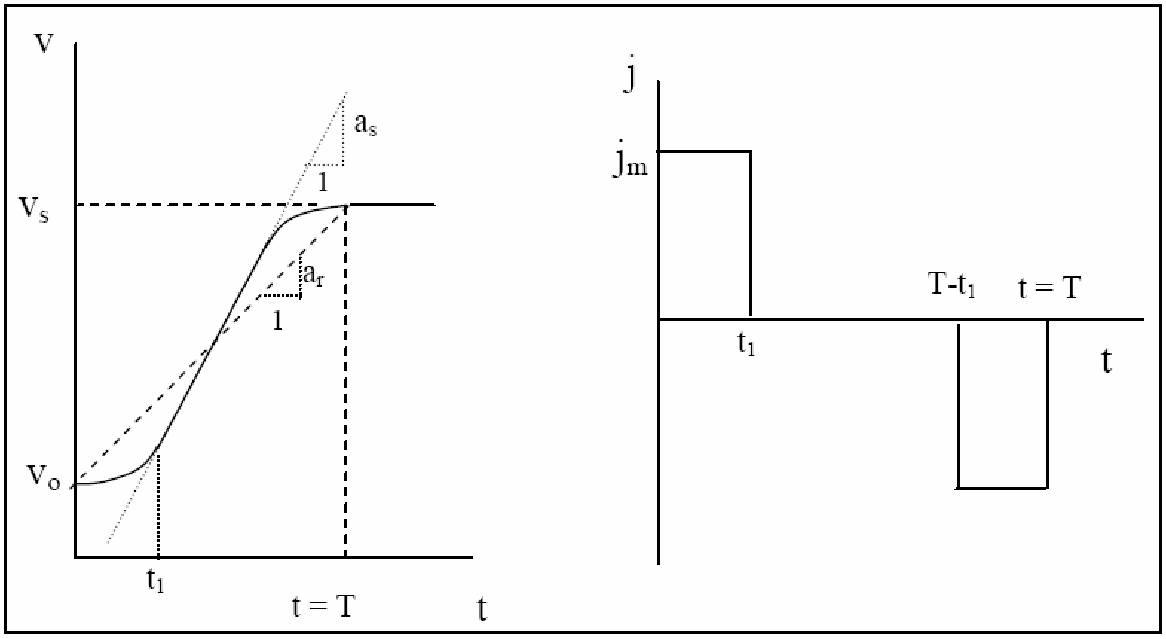

Figure 4-21 S-curve profile with linear period

The equations of motion for the linear portion of the s-curve are:

$$s(t) = v_1 t + a_s\, t^2/2 \tag{4-9}$$

$$v(t) = v_1 + a_s t \tag{4-10}$$

$$v_1 = v_0 + \frac{a_s^2}{2j} \tag{4-11}$$

The trajectory generator assumed that the feedrate was capable of maximum velocity at the beginning and end of the tool paths because the sample tool paths tested did not include non-cutting engagement moves.

Due to the high-curvature, the NC controller decreased the feedrate along all of the tool paths. Feedrates for each of the tool paths can be seen in Figure 4-22 through Figure 4-27 and Table 4-2. The graphs are all on a scale of 0 to 400 mm/s. Note that the graphs shift to the left as the path tolerance radii increase because the machine tool is traversing the tool path faster.

Table 4-2 Feedrate measurements from before and after smoothing algorithm

Part	Path Tolerance (mm)	Feedrate Average (mm/s)	Feedrate Difference (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%
Test Case 6: Square	Original	58.2	0.0	0.00%
	R = 0.025	59.5	1.3	2.29%
	R = 0.125	65.6	7.4	12.79%
	R = 0.250	71.6	13.4	23.01%

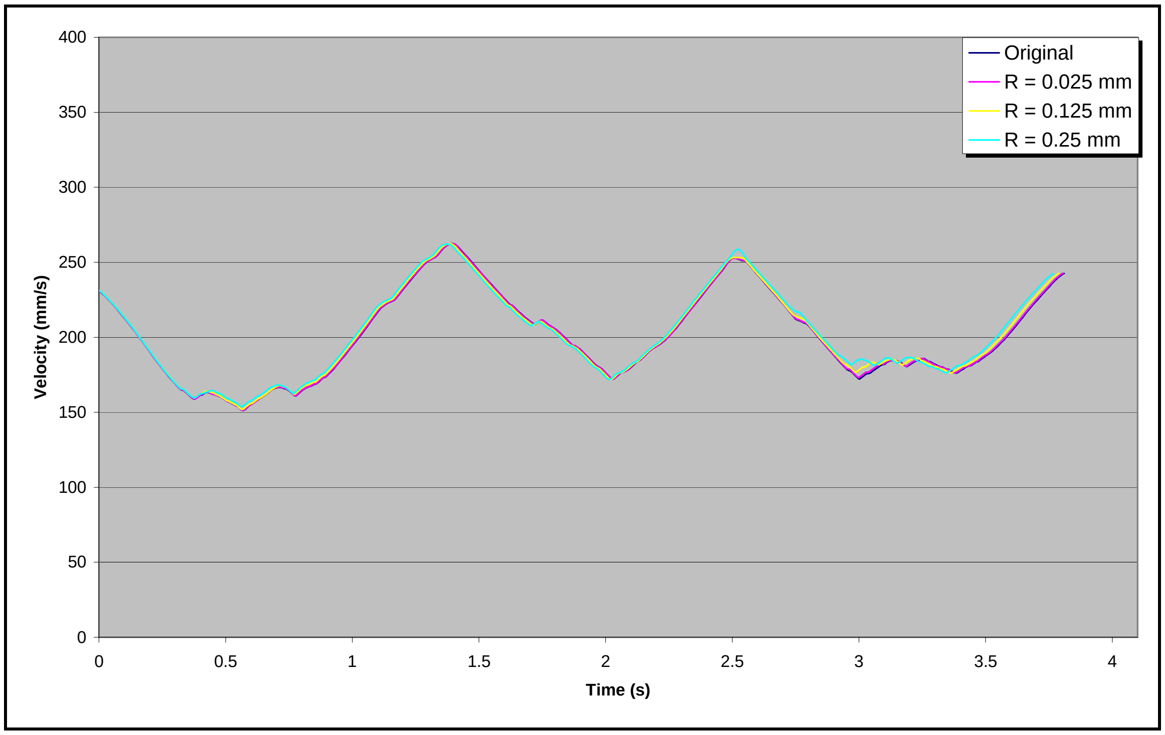

Figure 4-22 Test Case 1: Feedrate for tool path in Figure 4-1

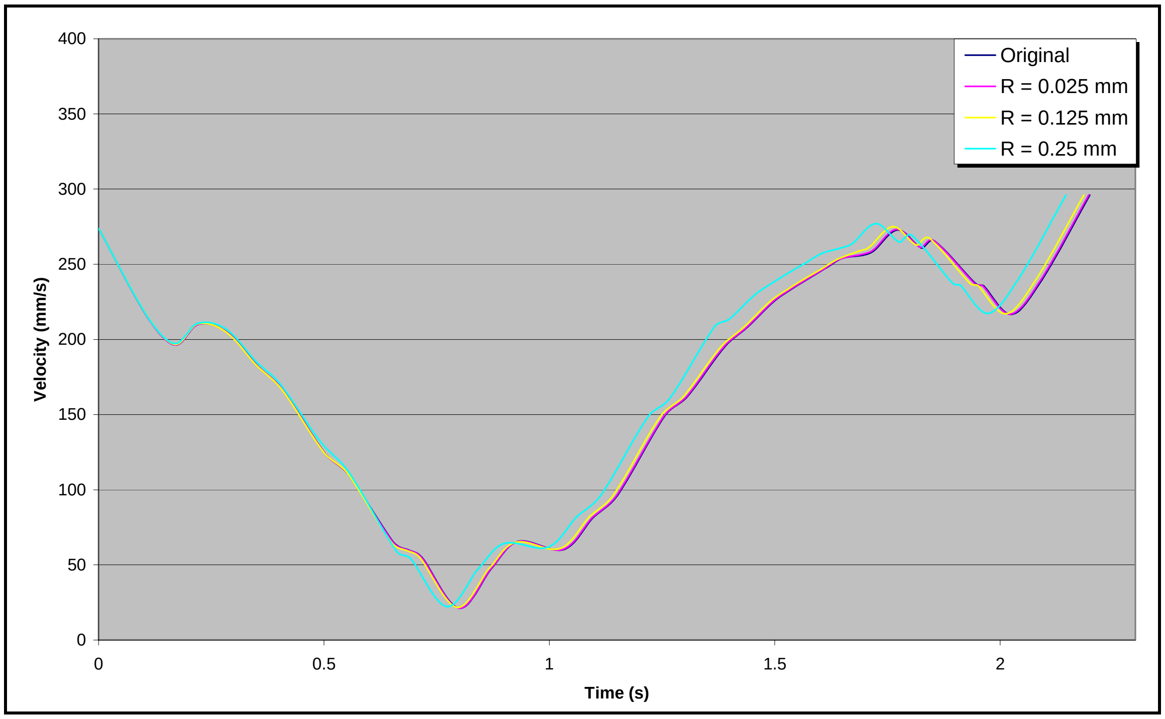

Figure 4-23 Test Case 2: Feedrate for tool path in Figure 4-2

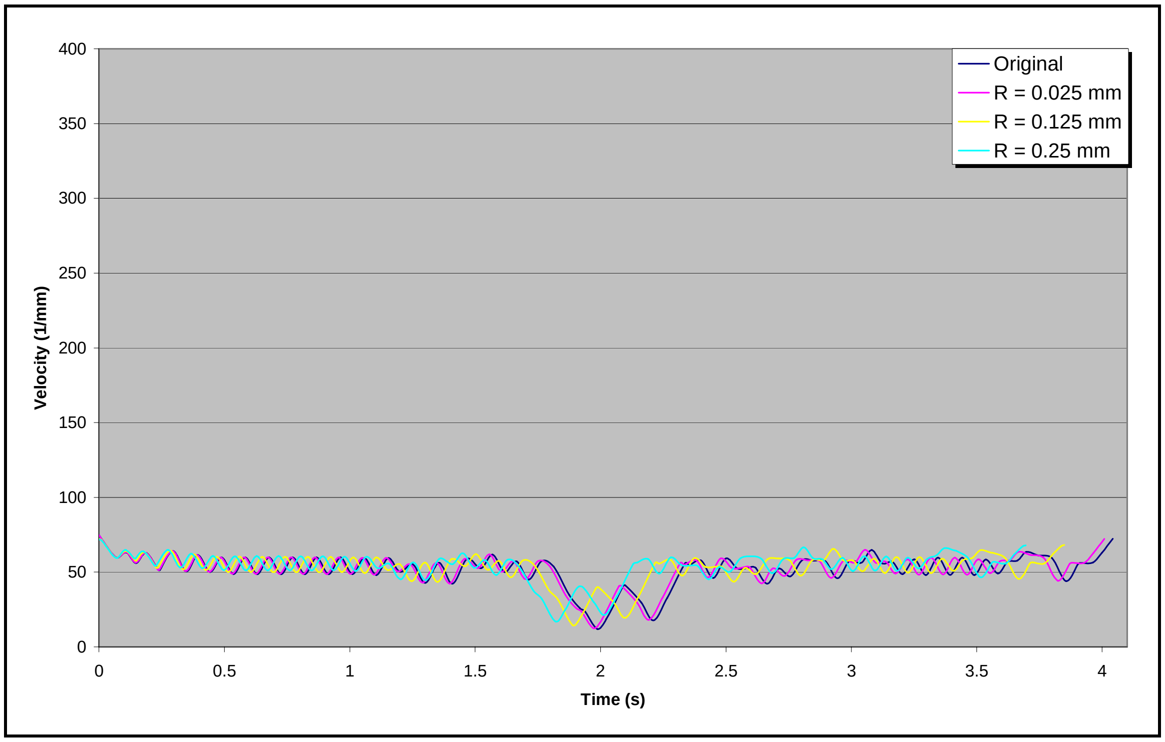

Figure 4-24 Test Case 3: Feedrate for tool path in Figure 4-3

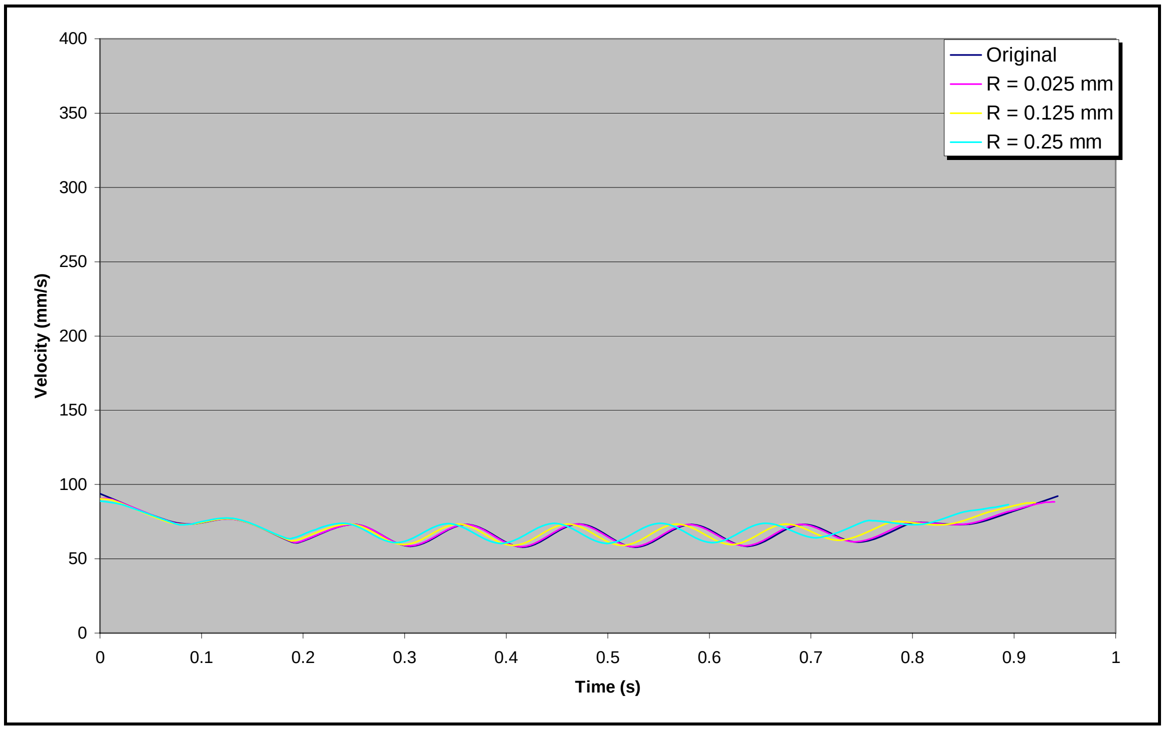

Figure 4-25 Test Case 4: Feedrate for tool path in Figure 4-4

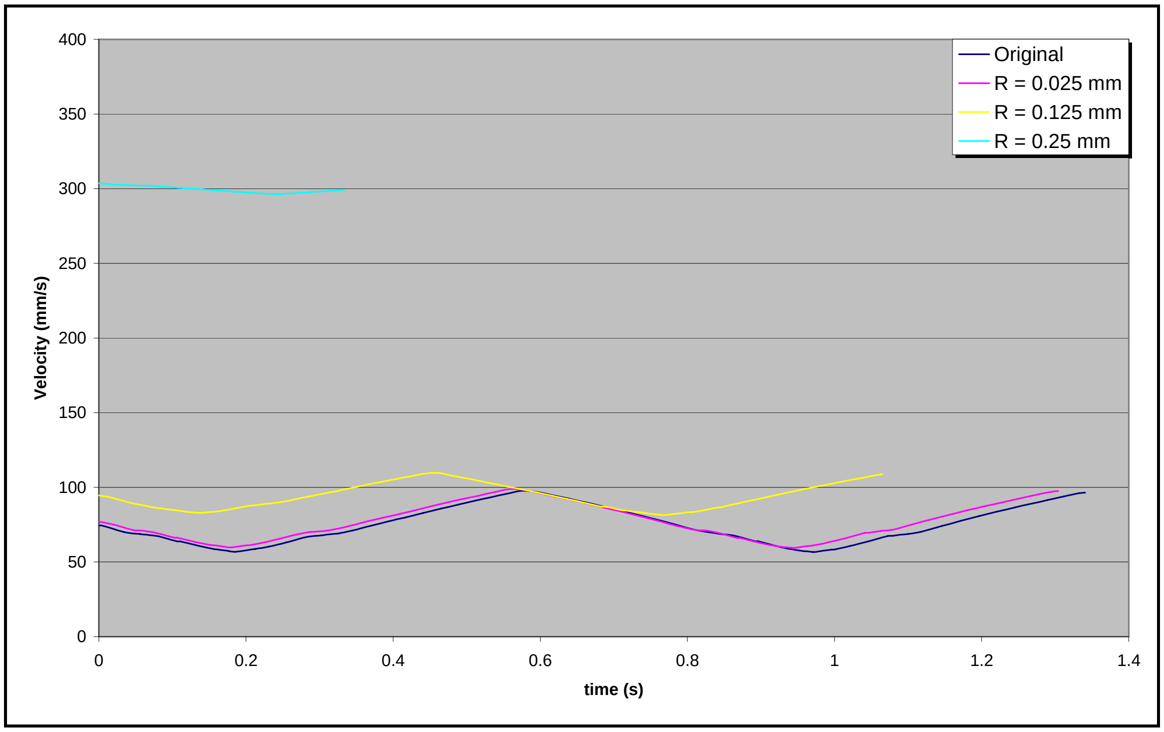

Figure 4-26 Test Case 5: Feedrate for tool path in Figure 4-6

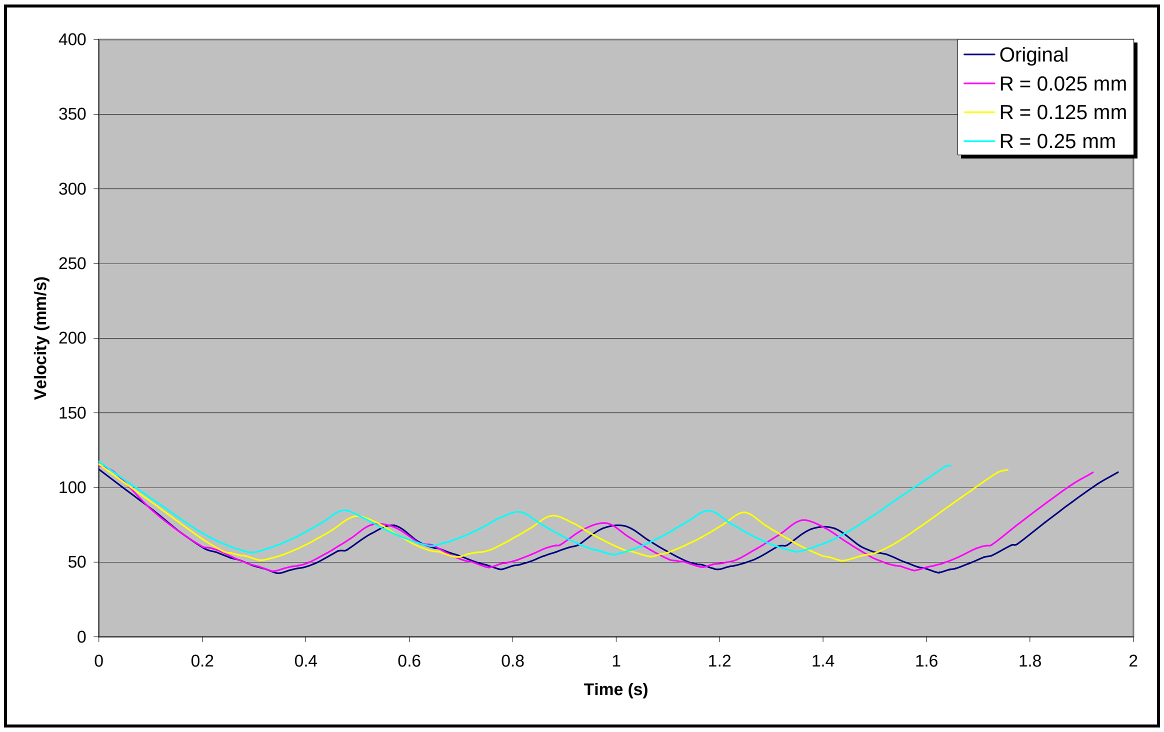

Figure 4-27 Test Case 6: Feedrate for tool path in Figure 4-7

4.6 Tool Path Length Results

The tool path lengths are recorded in Table 4-3. The tool paths decrease in length during the smoothing algorithm.

Table 4-3 Lengths of tool paths from before and after smoothing algorithm

Part	Path Tolerance (mm)	Length (mm)	Difference (mm)	Improvement %
Test Case 1: Simple	Original	766.089	0.000	0.00%
	R = 0.025	765.773	0.316	0.04%
	R = 0.125	764.601	1.488	0.19%
	R = 0.250	763.681	2.408	0.31%
Test Case 2: Medium	Original	376.394	0.000	0.00%
	R = 0.025	376.158	0.236	0.06%
	R = 0.125	375.179	1.215	0.32%
	R = 0.250	373.853	2.541	0.68%
Test Case 3: Complex	Original	209.649	0.000	0.00%
	R = 0.025	208.522	1.127	0.54%
	R = 0.125	203.649	6.000	2.86%
	R = 0.250	198.750	10.899	5.20%
Test Case 4: Snow Tire	Original	66.292	0.000	0.00%
	R = 0.025	66.051	0.242	0.36%
	R = 0.125	65.092	1.200	1.81%
	R = 0.250	63.817	2.475	3.73%
Test Case 5: Line	Original	100.444	0.000	0.00%
	R = 0.025	100.383	0.061	0.06%
	R = 0.125	100.157	0.287	0.29%
	R = 0.250	100.017	0.427	0.43%
Test Case 6: Square	Original	123.277	0.000	0.00%
	R = 0.025	123.200	0.077	0.06%
	R = 0.125	122.914	0.363	0.29%
	R = 0.250	122.263	1.014	0.82%

4.7 Simulated Machining Times Results

The machining times for each of the tool paths are displayed in Table 4-4. These demonstrate how much time was saved with the smoothing algorithm.

Table 4-4 Machining times for all of tool paths and percent of time saved

Part	Path Tolerance (mm)	Time (s)	Difference (mm)	Improvement (%)
Test Case 1: Simple	Original	3.810	0.000	0.00%
	R = 0.025	3.806	0.004	0.11%
	R = 0.125	3.790	0.020	0.54%
	R = 0.250	3.769	0.042	1.09%
Test Case 2: Medium	Original	2.199	0.000	0.00%
	R = 0.025	2.197	0.003	0.11%
	R = 0.125	2.186	0.013	0.59%
	R = 0.250	2.147	0.053	2.39%
Test Case 3: Complex	Original	4.042	0.000	0.00%
	R = 0.025	4.009	0.033	0.83%
	R = 0.125	3.849	0.193	4.78%
	R = 0.250	3.695	0.348	8.60%
Test Case 4: Snow Tire	Original	0.943	0.000	0.00%
	R = 0.025	0.940	0.003	0.36%
	R = 0.125	0.921	0.022	2.37%
	R = 0.250	0.895	0.048	5.14%
Test Case 5: Line	Original	1.341	0.000	0.00%
	R = 0.025	1.304	0.037	2.76%
	R = 0.125	1.065	0.276	20.58%
	R = 0.250	0.334	1.007	75.09%
Test Case 6: Square	Original	1.970	0.000	0.00%
	R = 0.025	1.922	0.048	2.45%
	R = 0.125	1.757	0.213	10.83%
	R = 0.250	1.647	0.324	16.43%

4.8 Calculation Times Results

The calculation times required by the algorithm to smooth the paths are shown in Table 4-5. 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 calculation times increase with the number of control points.

Table 4-5 Calculation times for all tool paths

Part	Number of Control Points	Tolerance Radius (mm)	Times (s)
Test Case 1: Simple	62	R = 0.025	1.636
		R = 0.125	1.661
		R = 0.250	1.514
Test Case 2: Medium	14	R = 0.025	0.076
		R = 0.125	0.085
		R = 0.250	0.090
Test Case 3: Complex	43	R = 0.025	0.800
		R = 0.125	0.969
		R = 0.250	0.779
Test Case 4: Snow Tire	11	R = 0.025	0.101
		R = 0.125	0.101
		R = 0.250	0.113
Test Case 5: Line	32	R = 0.025	0.231
		R = 0.125	0.339
		R = 0.250	0.851
Test Case 6: Square	28	R = 0.025	0.237
		R = 0.125	0.160
		R = 0.250	0.741

4.9 Discussion of Results

As the curvature in a tool path is reduced by the new algorithm, the machining time decreases as shown in Table 4-4. The algorithm decreased machining times by 1% to 9%

for design-induced sections of high curvature and by 16% to 75% for CAM-induced ripples using high path tolerances.

4.9.1 Tool Path Length

The algorithm decreased the tool path length for all of the tool paths. Table 4-3 shows the improvement in tool path length for all test cases and Table 4-6 shows the average decrease in tool path length for each tool path tolerance radius.

Table 4-6 Average percent changes in tool path length, curvature, feedrate, and machining time.

Tolerance Radius (mm)	Tool Path Length Average Percent Reduction	Curvature Average Percent Reduction	Feedrate Average Percent Increase	Machining Time Average Percent Reduction
Overall	1.05%	13.11%	21.85%	8.61%
0.025	0.20%	2.78%	1.12%	1.10%
0.125	1.02%	13.28%	7.31%	6.61%
0.25	1.93%	23.26%	57.13%	18.12%

4.9.2 Curvature

The algorithm decreased the curvature for all of the tool paths. The curvature graphs and Table 4-1 in Section 4.4 show the improvement in curvature for all test cases and Table 4-6 shows the average decrease in curvature for each tool path tolerance radius.

4.9.3 Feedrate

The algorithm increased the feedrate for all of the tool paths. The feedrate graphs and Table 4-2 in Section 4.5 show the improvement in curvature for all test cases and

Table 4-6 shows the average increase in feedrate. The feedrates were increased because the curvature was decreased. The curvature graphs in Section 4.4 and the corresponding feedrate graphs in Section 4.5 demonstrate an inverse relationship between the amount of curvature and the feedrate. Figure 4-28 shows that the percent decrease in curvature correlates to the percent increase in feedrate. The P-value for the Pearson correlation coefficient in Table 4-7 confirms the correlation. A P-value less than 0.05 corresponds to a significant correlation.

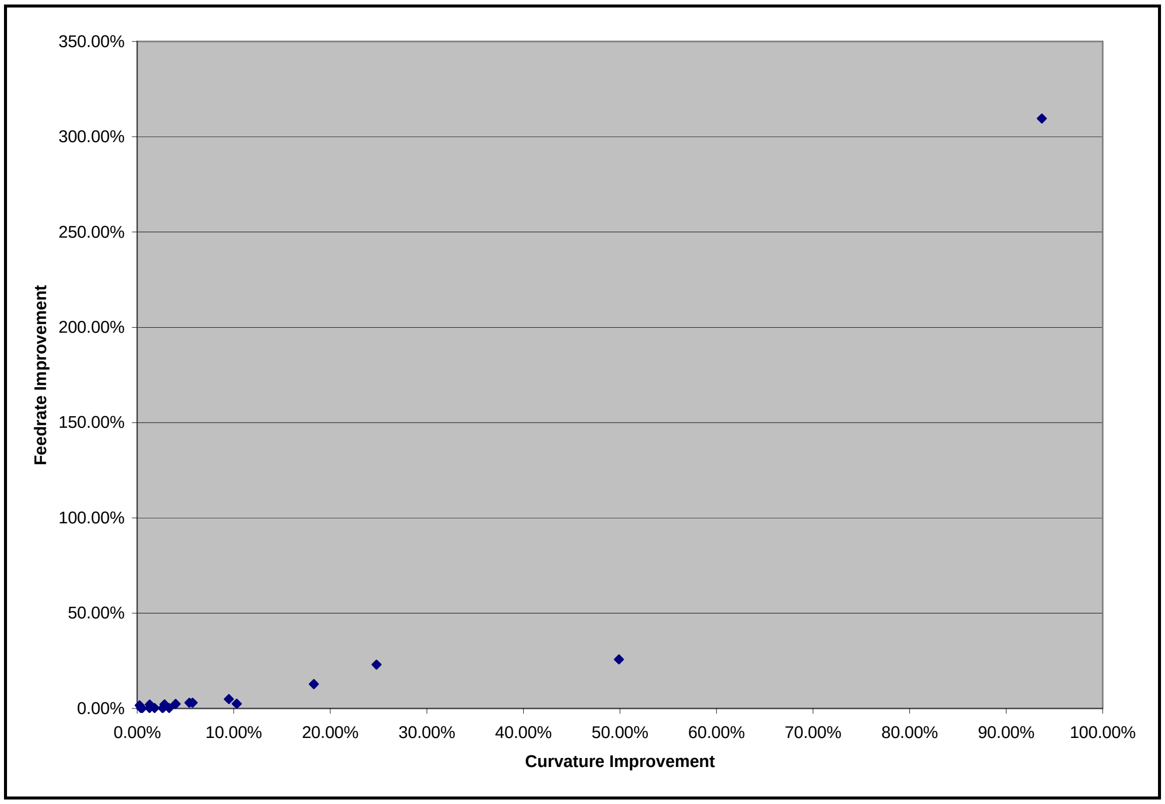

Figure 4-28 Percent reduction in curvature compared to percent increase in feedrate showing that decrease in curvature directly correlates to increase in feedrate.

4.9.4 Simulated Machining Times

The algorithm decreased the machining times for all of the tool paths. Table 4-3 shows the improvement in machining time for all test cases and Table 4-6 shows the average decrease in machining time compared to the percent improvement in curvature, tool path length, and feedrate. Machining times were decreased because the feedrates were increased, as machining times are inversely proportional to feedrates. Figure 4-29 shows that the percent increase in feedrate correlates to the percent decrease in maching time. The P-value for the Pearson correlation coefficient in Table 4-7 confirms the correlation. It can also be shown that as the curvature decreases the machining time decreases. This correlation is shown in Figure 4-30 and Table 4-7.

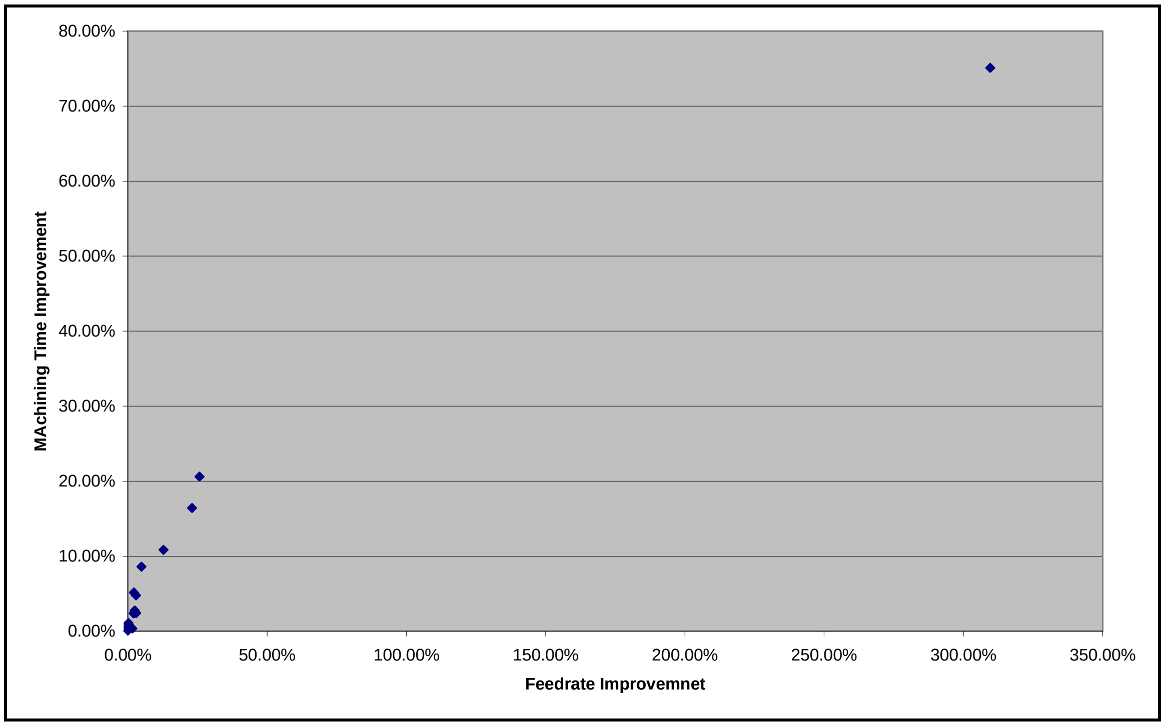

Figure 4-29 Percent increase in feedrate compared to percent decrease in machining time showing that increase in feedrate directly correlates to decrease in machining time.

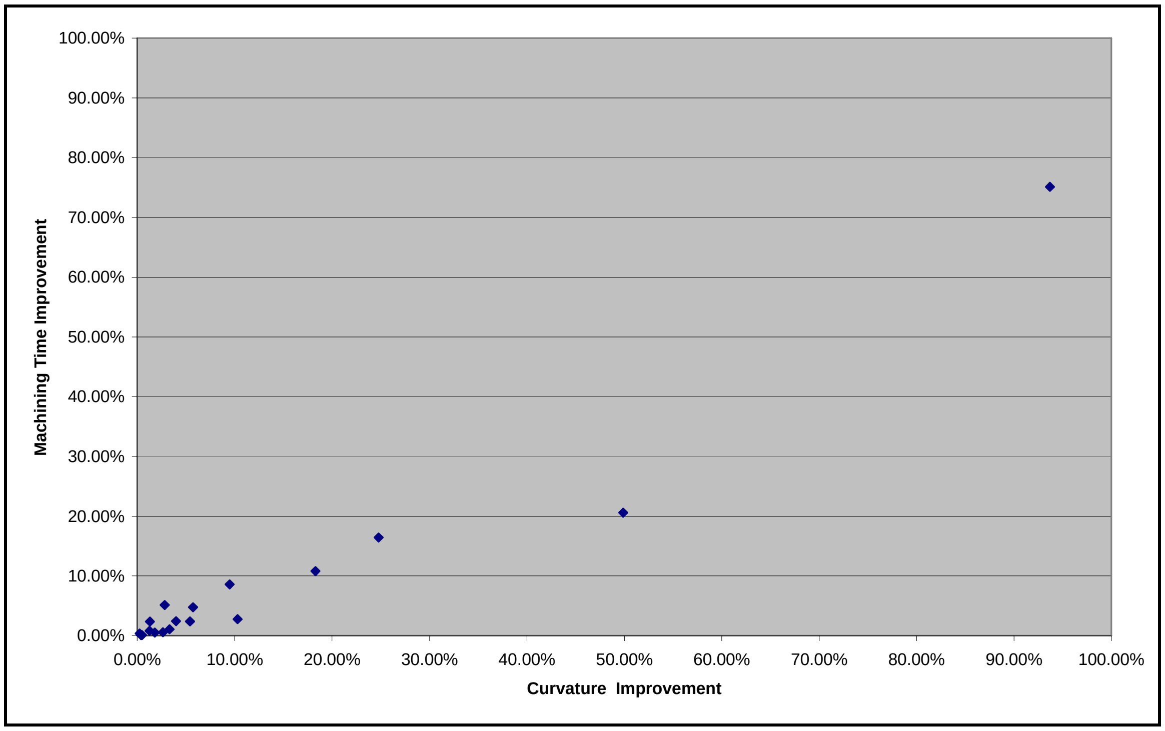

Figure 4-30 Percent decrease in curvature compared to percent decrease in machining time showing that decrease in curvature directly correlates to decrease in machining time.

Table 4-7 Pearson correlation values used to determine correlation between different parameters. Values in yellow represent a significant correlation.

		Machine Time Improvement %	Curvature Improvement %	Length Improvement %
Curvature Improvement %	Pearson correlation	0.9680		
	P-Value	0.0000		
Length Improvement %	Pearson correlation	-0.0280	-0.1110	
	P-Value	0.9110	0.6620	
Feedrate Improvement %	Pearson correlation	0.9720	0.9010	-0.1050
	P-Value	0.0000	0.0000	0.6770

The machining times were also affected by the reduction in tool path length. However, the decrease in machining time and the decrease in tool path length are not correlated. This is shown in Table 4-7 and Figure 4-31. This initially seems to be counterintuitive, but it can be understood by analyzing the nature of high-curvature tool paths.

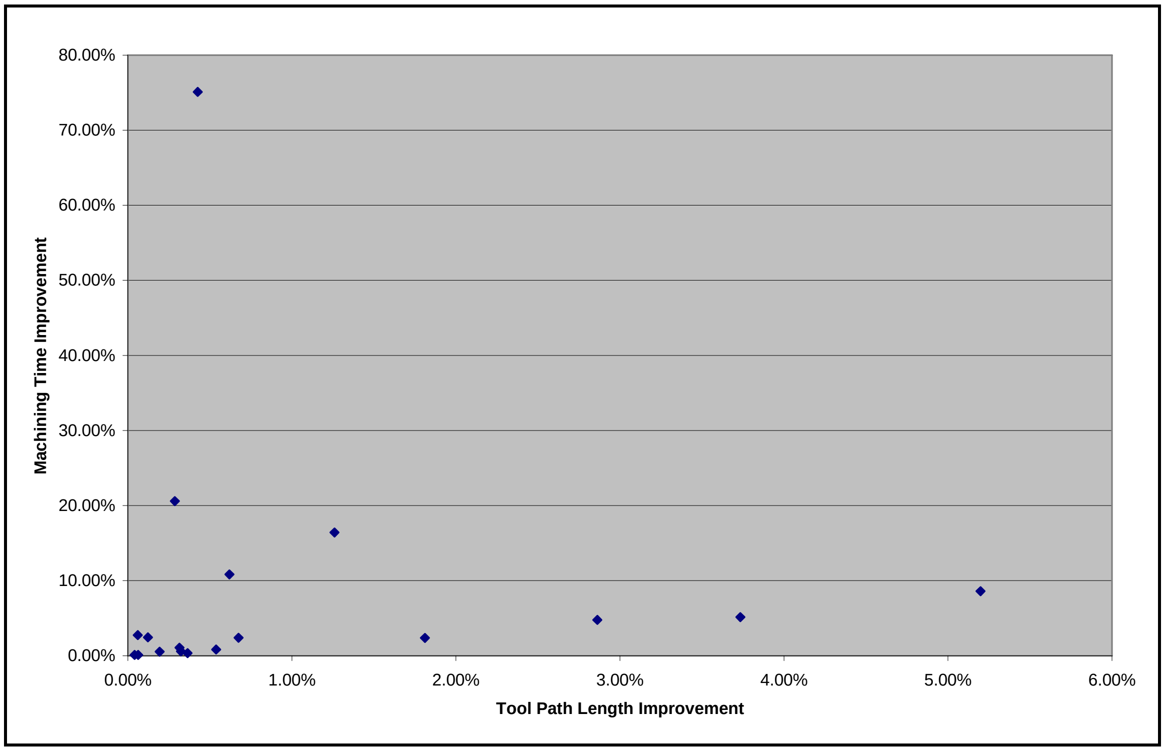

Figure 4-31 Percent decrease in tool path length compared to percent decrease in machining time showing that decrease in tool path length does not correlate to decrease in machining time.

The feedrate is coupled to the acceleration and jerk limits of the machine tool. It can be seen in the feedrate plots in Section 4.5. that the feedrate does not immediately return to its set value of 400 mm/s after reducing the feedrate in a high-curvature section. This is because the feedrate is limited by the acceleration and jerk limits. These limits impede the feedrate from returning to its desired value for a significant distance along the tool path. For example, in Figure 4-26 the feedrate drops to a minimum at the first section of high-curvature ripples. It does not have enough distance before the next high-curvature section to accelerate to 400 mm/s because of the acceleration and jerk limits. About half way between the high-curvature sections, the feedrate must again decelerate to a minimum value.

Due to the acceleration and jerk limits, decreasing the tool path length can have a minor adverse affect on feedrate peak values. If the tool path length is significantly shortened in a high-curvature section of the tool path, the machine tool will not have as much distance to accelerate before slowing for the next curve. The increase in machining time due to the reduced feedrate peaks may be offset by the reduction in machining time due to traversing the shortened path. These affects were not investigated. The reduction in feedrate at the beginning and ending of test case 4 in Figure 4-25 may be a result of this affect.

4.9.5 Tool Path Tolerance Radius

Increasing the path tolerance radius improves the results for curvature, tool path length, feedrate and machining time as shown in Table 4-6. A larger path tolerance allows the tool path to straighten further, which decreases the curvature as shown in the curvature graphs in Section 4.4. As the curvature decreases, the feedrate increases and the machining time decreases. The curvature does not decrease by large amounts because the path is only allowed to deviate within a small tolerance.

4.9.6 Percent Savings Versus Curvature

As expected, the test cases demonstrate that the algorithm yields better results for tool paths with higher curvature. The machining time for the tool path in Figure 4-2 was improved by a greater percent than the time in Figure 4-1 because the tool path in Figure 4-2 has a higher curvature distribution. The machining time for the tool path in Figure 4-3 was decreased by an even larger percent because the tool path has complex geometry with even higher curvature. This trend can be seen by comparing the curvature plots and machining times for each of the test cases.

4.9.7 Percent Savings Versus Discontinuities

High-curvature ripples caused by noise in the interpolation data provide considerable opportunity to decrease machining time. For example, if there were no noise in the tool path, the feedrate for the tool path in Figure 4-6 would remain at 400mm/s along the straight line and the tool path would be machined in 0.25 seconds. However, the high-curvature ripples cause the section to be machined in 1.34 seconds which is 536% of the ideal machining time. The smoothing algorithm decreases this time to 0.334 seconds, which is only 136% of the ideal time. The machining time does not reach the ideal time of 0.25 seconds, but it does improve the time by 75%, a significant improvement.

Anomalies in the CAD model, such as gaps and other discontinuities, also provide opportunities for improvement. For example, the machining times for the tool path in Figure 4-7 were improved by up to 16%. The curvature decreases significantly as the algorithm resolves the tangency discontinuities.

5 Conclusion

Tool paths that are designed with high levels of curvature can be smoothed with the smoothing algorithm presented in this thesis. In test cases 1 through 4, the machining times were decreased by 1% to 9% based on the degree of curvature. Resulting savings can be significant on complex parts. Tool paths with high-curvature ripples caused by discontinuities can also be smoothed using the algorithm. In test cases 5 and 6, the machining times were decreased by 16% to 75%. Anomalies should be removed before creating tool paths. But if they are not, the smoothing algorithm can 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. If these high-curvature ripples are smoothed with the algorithm proposed in this thesis, machining times can be decreased and manufacturing companies can save money.

5.1 Future Work

Opportunities to extend the work in this thesis include algorithms for smoothing closed tool paths and 3D tool paths. Brief explanations of closed tool paths and 3D tool paths are included in sections 5.1.1 and 5.1.2, respectively.

Investigations into using the smoothing algorithm as a filter for noisy data in reverse engineering of 3D surfaces could be performed. Smoothing the high-curvature ripples caused by noisy points would effectively filter out those points. This was illustrated in the tool path from Figure 4-6.

The algorithm's performance may be improved. The algorithm always decreases the overall machining time but sometimes it will increase the curvature of localized sections of the tool path. This is a rare case that sometimes occurs when the path tolerance is set too large. If the tolerance radius is too large, the tool path will form straight lines with small radius corners. For example, if the tool path had the shape of the letter "U" and the path tolerance was very large, the smoothed path could take the shape of the letter "V".

Even though the algorithm is fast, it has not been optimized to perform in a real-time environment. Work should be done to improve the calculation time of the smoothing algorithm.

The algorithm was developed to show generally that high-curvature sections in tool paths can be smoothed. The algorithm can now be adapted to a specific CAD, CAM, or NC system.

5.1.1 Closed Tool Paths

Although the smoothing algorithm works only with open-ended B-spline tool paths, closed tool paths are commonly used and should also be considered. An algorithm to smooth closed tool paths would be very similar to the open-ended algorithm. In fact, a closed tool path algorithm would be simpler than the open-ended one because the closed tool path has no true end. Therefore, there would be no endpoints to consider.

If a point was at the beginning of the array of control points, the algorithm would simply look at the other end of the array for the points preceding it in the polygon. This section will describe the modifications to the open-ended algorithm to convert it for use with closed tool paths.

Some of the math used for closed splines is slightly different than for open splines. See Dr. Sederberg's text for details on closed splines. The parsing would be almost exactly the same except that the knot vector would contain knot intervals instead of knot values. Knot intervals are the interval difference between the knot values. For example, if one knot is 5 and the next knot is 6, then the knot interval is 1. In a closed cubic B-spline, there is the same number of control points as knots.

The functions for categorizing points and determining direction vectors would not need to consider endpoints, as endpoints do not exist in closed splines. Control points would be categorized and direction vectors would be calculated using the same functions as for the open-ended smoothing algorithm. The functions to calculate sensitivity values, maximum allowable distance, move distances, and the smooth control polygon and termination would be essentially the same as for the open-ended spline algorithm.

5.1.2 3D Tool paths

In addition to 2D tool paths, 3D tool paths are commonly used. The algorithm in this thesis could be extended to 3D tool paths. 3D B-splines are essentially the same as 2D B-splines. Even though the B-splines math is similar between 2D and 3D, the algorithm needed to smooth the 3D tool path would be more complex. This section describes suggestions on how the 2D algorithm could be extended to use 3D splines. The techniques in this section have not been tested.

The 3D algorithm would incorporate the same basic principles and steps as the 2D version. The 3D algorithm would also move the control points towards forming a straight line so that the spline would have zero curvature. A line in 3D space has zero curvature just like a line in 2D space. The path tolerance would form a cylindrical offset tube around the path instead of a two-dimensional band. This would allow the spline to deviate in one more dimension than the 2D case.

The 3D algorithm would parse all of the same data. The control points would be in 3D instead of 2D. The rest of the parameters would be the same as for 2D.

Sorting the control points into end, ripple and smooth points would be the most complex part of a 3D conversion. The control points would no longer be in a single plane, which means that lines and distances to lines would no longer be sufficient to distinguish between smooth and ripple points. The first and last points would continue to be categorized as endpoints.

The 3D algorithm would still categorize the points using the shape of the control polygon. In the 3D case, lines would not be sufficient to categorize points. Two planes would be used in place of each of the two lines that are used to categorize each point in the 2D algorithm. This would produce four planes total per control point.

To distinguish between ripple and smooth points, the algorithm would first calculate the equations of two orthogonal planes that would go through the point being considered and the point just before it. The intersection of the two orthogonal planes would be the same line that is used in the 2D algorithm to categorize points. The two planes would split the space into quadrants. The rotation of the planes would need to be determined. Next, the points immediately before and after the points used to make the planes would be checked to determine their quadrant.

The 3D tool path would have direction vectors that are essentially the same as the 2D path. The direction vectors would be calculated using the same algorithms as the 2D, except the 3D tool path would use 3D vectors. Direction vectors would be calculated using three points that would lay in a plane, so the math would be the same as in the 2D algorithm. The sensitivities, maximum distance, move distances, smooth control polygon, and termination would be essentially the same for 3D as for 2D.

6 References

[1] Beckman, B. "Physics of Racing Series." n.d. http://www.miata.net/sport/Physics/index.html (accessed July 30, 2007).

[2] Berg, J. "Anytime Path Planning and Replanning in Dynamic Environments." *IEEE International Conference on Robotics and Automation* (May 2006): 2366-2371.

[3] 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).*

[4] Blinkhorn, R. "Driving Techniques." 2006, http://www.gpracing.net192.com/drivers/techniques.cfm (accessed July 30, 2007).

[5] Bohez, E. "Compensating for Systematic Errors in 5-Axis NC Machining." *Computer-Aided Design* 34 (2002): 391-403.

[6] Choi, B.; Jun, C. "Ball-End Cutter Interference Avoidance in NC Machining of Sculptured surfaces." *Computer-Aided Design* 21 (1989): 371-378.

[7] Eilers, P.; Marx, B. "Flexible Smoothing with B-Splines and Penalties." S*tatistical Science* 11, no. 2 (May 1996): 89-121.

[8] Erkorkmaz, K.; Altintas, Y. "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.

[9] Fleisig, R.; Spence, A. "A Constant Feed and Reduced Angular Acceleration Interpolation Algorithm for Multi-Axis Machining." *Computer-Aided Design* 33 (2001): 1-15.

[10] Geraerts, R.; Overmars, M. "The Corridor Map Method: Real-Time High-Quality Path Planning." *IEEE International Conference on Robotics and Automation* (April 2007): 1023-1028.

[11] Higashi, M.; Aoki, N.; Kaneko, T. “Application of Haptic Navigation to Modify Free-Form Surfaces Through Specified Points and Curves.” *Journal of Computing and Information Science in Engineering* 2 (December 2002): 265-276.

[12] International TecheGroup Incorporated, “CAD Model Quality" (September 2003)

[13] Jacobs, P.; Canny, J. “Planning Smooth Paths for Mobile Robots.” *IEEE Conference on Robotics and Automation* (1989): 2-7.

[14] Jung, S.; Jang, E. “Collision Avoidance of a Mobile Robot Using Intelligent Hybrid Force Control Technique.” *IEEE International Conference on Robotics and Automation* (April 2005): 4418-4423.

[15] Kanayama, Y.; Hartman, B. “Smooth Local Path Planning for Autonomous Vehicles.” *International Journal of Robotics Research* 16 (1997): 263-284.

[16] Langeron, J.; Duc, E.; Lartigue, C.; Bourdet, P. “A New Format for 5-Axis Tool Path Computation, Using Bspline Curves.” *Computer-Aided Design* 36 (2004): 1219-1229.

[17] Lee, C.; Haralick, R.; Deguchi, K. “Estimation of Curvature from Sampled Noisy Data.” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (June 1993): 536-541.

[18] “Line (racing).” n.d. http://en.wikipedia.org/wiki/Line_(racing) (accessed July 30, 2007).

[19] Papadopoulos, E.; Tortopidis, I.; Nanos, K. “Smooth Planning for Free-Floating Space Robots Using Polynomials.” *IEEE International Conference on Robotics and Automation* (April 2005): 4272-4277.

[20] “Racing line.” n.d. http://en.wikipedia.org/wiki/Racing line (accessed July 30, 2007).

[21] Red, W. “Constant Jerk Equations for a Trajectory Generator” http://www.et.byu.edu/~ered/ME537/Notes/Ch5.pdf (accessed Oct 10, 2007).

[22] Sadoyan, H.; Zakarian, A.; Avagyan, V.; Mohanty, P. 2006. “Robust Uniform Triangulation Algorithm for Computer Aided Design.” *Computer-Aided Design* 38 (2006): 1134-1144.

[23] Sahin, T.; Zergeroglu, E. “A Computationally Efficient Path Planner for a Collection of Wheeled Mobile Robots with Limited Sensing Zones.” *IEEE International Conference on Robotics and Automation* (April 2007): 1074-1079.

[24] Scherer, S.; Singh, S.; Chamberlain, L.; Saripalli, S. “Flying Fast and Low Among Obstacles.” *IEEE International Conference on Robotics and Automation* (April 2007): 2023-2029.

[25] Scheuer, A.; Fraichard, T. “Continuous-Curvature Path Planning for Car Like Vehicles.” *IEEE Int. Conf. on Intelligent Robots and Systems* 2 (September 1997): 997-1003.

[26] Sederberg, T. “Computer Aided Geometric Design.” April 5, 2007 http://tom.cs.byu.edu/~557/ (accessed July 30, 2007).

[27] Sellen, J. “Planning Paths of Minimal Curvature.” *IEEE International Conference on Robotics and Automation* (1995): 1976-1982.

[28] Tang, C.; Medioni, G. “Robust Estimation of Curvature Information from Noisy 3-D Data for Shape Description.” *IEEE International Conference on Computer Vision* (September 1999).

[29] “The Racing Line.” n.d. http://www.driversdomainuk.com/motorsport/racingline.php (accessed July 30, 2007).

[30] Velenis, E.; Tsiotras, P. “Minimum Time vs Maximum Exit Velocity Path Optimization During Cornering.” *IEEE International Symposium on Industrial Electronics* (June 2005).

[31] Velenis, E.; Tsiotras, P. “Optimal Velocity Profile Generation for Given Acceleration Limits: Receding Horizon Implementation.” *American Control Conference Portland, OR, USA* (June 2005): 2147-2152.

[32] Wang, F.; Yang D. “Nearly Arc-Length Parameterized Quintic Spline Interpolation for Precision Machining.” *Computer-Aided Design* 25 (1993): 281-288.

[33] Wilfong, G. “Shortest Paths for Autonomous Vehicles.” *Proceedings of the IEEE International Conference on Robotics and Automation* (1989): 15-20.

[34] Zarrabi-Zadeh, H. “Path Planning Above Polyhedral Terrain.” *IEEE* International Conference on Robotics and Automation (May 2006): 873-876.