Precision Contouring Error Analysis
Curtis S. Wilson
Delta Tau Data Systems
9036 Winnetka Ave.
Northridge, Calif. 91324

INTRODUCTION

There are many potential error sources that must be considered in the design of ultraprecision motion control equipment. One source that gets little attention is the trajectory generation algorithm for the controller. Either this source is ignored in the analysis, or conservative assumptions that were valid twenty years ago are used, ignoring recent developments. With the advent of modern computational hardware, powerful techniques are available for keeping these errors to an insignificant amount at very reasonable costs. Still, computational resources must be utilized carefully, particularly in complex contouring applications, so tradeoffs must be struck between permitted errors and computational complexity.

BACKGROUND ANALYSIS

Conceptually, contouring in a digital controller is really an exercise in digital sampling of a continuous waveform, much like digital recordings of audio waveforms. Samples of the ideal contour are taken -- simultaneously for all axes, like the dual channels of a stereophonic system -- and in execution, a facsimile of the ideal waveform is created. The samples must be taken close enough together to permit reasonable recreation of the contour, but not so close together as to surpass either the storage or processing-speed capabilities of the system. In machine tool terms, the sample rate chosen is called the block rate, and an important figure of merit for any machine tool is its maximum block rate.

Digital sampling theory tells us that the minimum sample rate that permits perfect reconstruction is, the Nyquist frequency, which is just over twice the highest frequency contained in the original contour. It also tells us that the ideal reconstruction is to convolve each of the samples with the "sinc" function, which is:

\[ \text{sinc}(n) = \frac{\sin (\pi n)}{\pi n} \]  \hspace{1cm} (1)

where there is one unit of \(n\) for each sample of the waveform. There are several reasons why we cannot use this ideal reconstruction. First, it is an analog reconstruction, and we are using digital systems. This is not an important problem, because we can "reconstruct", or interpolate, at the servo sample rate of the system, which already has to be fast enough for control purposes to make any quantization problems here negligible. The interpolation algorithms are, in effect, digital oversampling algorithms, again much like those in audio systems, operating at the system's servo update rate.

The second reason why using the sinc function for reconstruction is not practical is that the computational complexity. Reduction to lookup table is really feasible only when the ratio of block rate to servo rate is fixed, which is not a tolerable constraint on machine tool operation. Real time
calculation of multiple sinc function values will remain prohibitively expensive for the foreseeable future.

The final reason that "sinc" reconstruction cannot be use directly is that it is an infinitely long function in both directions. Even though it gets vanishingly small before too many samples have passed, we cannot just truncate the function after a certain number of samples, because this would introduce integration errors. The area under the ideal sinc function curve is exactly 1, and if a truncated version were used that did not have this same area, position errors would creep in as integration errors.

PRACTICAL TECHNIQUES

The interpolation algorithm that we use must have several characteristics. It must keep interpolation errors to within tolerable bounds while keeping the required block rates within the system's limits. Its sample reconstruction must be finite in length, relatively easy to calculate, and not lead to accumulated errors. It must also lead to a physically realizable path for the axes of the system.

Most motion control systems use polynomial equations for their position update functions. Properly generated, these can serve as good interpolation routines for precision contouring applications. For a given system, decisions have to be made regarding both the order of the polynomial update equation used, and how the coefficients for the equations are generated from the rough move descriptions. The higher the order of the polynomial, the more the flexibility, but the higher the calculational requirements. There is a continuing trend toward higher-order polynomials; roughly speaking, in the 1970s, first-order equations were used; in the 1980s, second-order; and in the 1990s third-order.

The most basic technique is to use no interpolation between block sample points; this provides command position steps and command velocity impulses. Of course, these are impossible for a physical system to follow, so there must be significant low-pass filtering in the servo loop, which produces noticeable "servo lag". The block sample points must be quite close together to keep the impulse energy within manageable bounds. The only practical systems using this approach have been robotic ones, in which the sample points are the results of the inverse kinematic equations, computed at quite rapid intervals (often 5-30 msec), and fed directly to the commanded joint positions without interpolation.

First order interpolation has been used extensively in NC machine tools. This produces constant velocity commands between block sample points on each axis, which in Cartesian systems leads to chordal approximations of contours. In analytical terms, the velocity impulse is reconstructed as a rectangular velocity profile by convolving the impulse with a unit rectangle. With the standard techniques for generating these chords, the endpoints of the chord are on the ideal contour, and the worst error is at the midpoint of the chord. For a curve with local radius \( R \), and a chord subtending an angle \( \theta \) of the curve, the effective radius at the midpoint of the chord is:

\[
R' = R \cos (\theta/2)
\]

which creates an error
\[ E = R - R' = R \left[ 1 - \cos \left( \theta/2 \right) \right] \quad (3) \]

Since the arc distance for a segment, \( R \) times \( \theta \), is also the vector velocity \( V \) multiplied by the segment time \( T_a \), and using the Taylor series approximation \( \cos x = 1 - x^2/2! + x^4/4! + \ldots \), we can approximate the error as

\[ E \equiv R \left[ 1 - \cos \left( VT_a/2R \right) \right] \equiv V^2 T_a^2 / 8 \ R \equiv R \ \theta^2 / 8 \quad (4) \]

However, this method requests step changes in velocity at block boundaries, which are not physically realizable, so substantial servo lag is required, which produces significant errors of its own. For a system with servo lag due to uncompensated mechanical time constant \( T_m \), this error can be approximated as

\[ E \equiv V^2 T_m^2 / 2 \ R \quad (5) \]

With second-order position update polynomials, it is possible to introduce acceleration control, which largely eliminates the need to maintain servo lag (modern servo techniques make it possible to eliminate this lag, but the interpolation techniques must permit it). Now the velocity impulse reconstruction appears as a triangle, not a rectangle; the velocity rectangle of the first-order technique is convolved with a unit acceleration rectangle, resulting in the triangle. Because the axis velocities are no longer constant in the general case, the path segments are no longer straight. Instead of chords, we get curves joining the midpoints of the chords that are approximately parallel to the ideal curve. Now the error as the reconstructed curve passes the sample points with approximately the same error as at the midpoints shown in Equation 4.

This technique requests step changes in acceleration, and hence, step changes in motor current, which are not possible due to motor inductance. With third-order interpolation, acceleration can be kept continuous as well. The velocity impulse reconstruction now looks roughly bell-shaped, the result of convolving the triangular response with a unit rectangular "jerk" \((da/dt)\) response, and a cubic spline is produced. This spline passes through "via points" \( V P_n \) on each axis

\[ V P_n = (P_{n-1} + 4 \ P_n + P_{n+1}) / 6 \quad (6) \]

On a curve of local radius \( R \) and subtended angle \( \theta \), the effective radius is

\[ R' = \left[ R \ \cos (-\theta) + 4 \ R + R \ \cos (\theta) \right] / 6 = R \ (2 + \cos \ \theta) / 3 \quad (7) \]

producing an error of

\[ E = R \ (1 - \cos \ \theta) / 3 \equiv V^2 T_a^2 / 6 \ R \equiv R \ \theta^2 / 6 \quad (8) \]

The error at the midpoints is the same to second order approximation. That this error is slightly larger than the second-order reconstruction is the "cost" of the added smoothness. The process of increasing the polynomial order and creating "smoother" impulse reconstructions by more convolutions can be continued indefinitely; in the limiting case a Gaussian impulse reconstruction is generated, and the error produced is

\[ E = V^2 T_a^2 / 2 \ R \equiv R \ \theta^2 / 2 \quad (9) \]
CORRECTED POLYNOMIAL TECHNIQUES

All of the above techniques produce predictable errors to the inside of the curve. Since they are predictable, good pre-compensation algorithms are possible. This paper presents simple techniques that are very effective at reducing error by modifying the axis sample points $P_n$ for each axis that are on the ideal curve to points $P'_n$ that are outside this curve.

For the first-order chordal approximation the compensation that cuts the maximum interpolation error in half by producing equal inside and outside errors is

$$P'_n = (-P_{n-1} + 18 P_n - P_{n+1}) / 16 \quad (10)$$

Different precompensation equations can be used if it is also desired to compensate for the servo lag error shown in Equation 5; the exact equation will depend on the ratio between $T_a$ and $T_m$.

In the controlled-acceleration interpolation algorithm, the interpolation error can be dramatically reduced, eliminating the second-order error term, by the precompensation

$$P'_n = (-P_{n-1} + 10 P_n - P_{n+1}) / 8 \quad (11)$$

The maximum resulting error is

$$E \equiv V^4 T_a^4 / 64 R^3 \equiv R \theta^4 / 64 \quad (12)$$

For the cubic-spline reconstruction, the precompensation that eliminates the second-order error term is

$$P'_n = (-P_{n-1} + 8 P_n - P_{n+1}) / 6 \quad (13)$$

The maximum resulting error in this case is

$$E \equiv V^4 T_a^4 / 36 R^3 \equiv R \theta^4 / 36 \quad (14)$$

A DIFFERENT APPROACH

A relatively new approach that is gaining in popularity samples not only axis positions at each block boundary, but axis velocities as well. The reconstruction algorithm takes the beginning and ending positions and velocities, and the block time, and computes the unique cubic position equation coefficients for the segment that meet these constraints (the ending position and velocity are used to compute the beginning acceleration and constant jerk for the segment).

In this approach, the commanded trajectory passes exactly through the sampled points with continuous velocity, although not necessarily with continuous acceleration. The worst error is at the segment midpoints; this error can be approximated as

$$E \equiv V^4 T_a^4 / 384 R^3 \equiv R \theta^4 / 384 \quad (15)$$
This error is almost an order of magnitude smaller than the errors in any of the other techniques, but note that because it requests step changes in acceleration, and hence motor currents, the resulting errors in actual position may be larger than the pre-compensated cubic spline.

**SUMMARY**

The following table summarizes the errors for each of the techniques shown above. The formulas for via point errors and variations in errors between via points and segment midpoints are given, as are the worst case errors for sampling a subtended angle of $10^9$ on the curve.

<table>
<thead>
<tr>
<th>Method</th>
<th>Via Point Error</th>
<th>Error Variation</th>
<th>$10^9$ Segment Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Order (Chordal)</td>
<td>0</td>
<td>$V^2 T_a^2 / 8 R$</td>
<td>$3.8 \times 10^{-3} R$</td>
</tr>
<tr>
<td>Corrected 1st Order</td>
<td>$-V^2 T_a^2 / 16 R$</td>
<td>$V^2 T_a^2 / 8 R$</td>
<td>$1.9 \times 10^{-3} R$</td>
</tr>
<tr>
<td>2nd Order</td>
<td>$V^2 T_a^2 / 8 R$</td>
<td>$V^4 T_a^4 / 144 R^3$</td>
<td>$3.8 \times 10^{-3} R$</td>
</tr>
<tr>
<td>Corrected 2nd Order</td>
<td>$V^4 T_a^4 / 64 R^3$</td>
<td>$V^4 T_a^4 / 144 R^3$</td>
<td>$1.4 \times 10^{-5} R$</td>
</tr>
<tr>
<td>3rd Order</td>
<td>$V^2 T_a^2 / 6 R$</td>
<td>$V^4 T_a^4 / 384 R^3$</td>
<td>$5.1 \times 10^{-3} R$</td>
</tr>
<tr>
<td>Corrected 3rd Order</td>
<td>$V^4 T_a^4 / 36 R^3$</td>
<td>$V^4 T_a^4 / 384 R^3$</td>
<td>$2.6 \times 10^{-5} R$</td>
</tr>
<tr>
<td>Position-Velocity</td>
<td>0</td>
<td>$V^4 T_a^4 / 384 R^3$</td>
<td>$2.4 \times 10^{-6} R$</td>
</tr>
</tbody>
</table>

**CONCLUSIONS**

Interpolation errors are a potentially important component of the error budget for a precision machine. The interpolation method and the point spacing must be considered carefully in view of the computational capabilities of the machine and the precision requirements. Modern techniques utilizing higher-order interpolation equations and pre-compensation methods permit intelligent trade-offs between required block rates and interpolation complexity. These techniques render the traditional method of evaluating the chordal error obsolete. The flexibility they bring allow much higher performance for a given computational capability, without sacrificing precision.
Contouring Interpolation Methods

Steps
(0th Order Update)

Constant Velocity
(1st Order Update)

Constant Accel
(2nd Order Update)

3-Point Spline
(3rd Order Update)
Contouring Impulse Reconstructions -- Uncorrected

- Ideal
- 1st Order
- 2nd Order
- 3rd Order
Contouring Impulse Reconstructions -- Corrected

- Ideal
- 1st Order
- 2nd Order
- 3rd Order