Realtime curve interpolators

M Shpitalni, Y Koren* and C C Lo*

The amount of geometric information that must be transferred between a CAD system and a computerized numerical control system creates a conflict between part precision on the one hand and feedrate fidelity and communications load on the other. This is the motivation for the development of new curve interpolation algorithms for CNC. The interpolation depends on the method of curve representation, i.e. the use of an implicit or a parametric form. Accordingly, the paper presents two realtime interpolation algorithms and compares them with existing CAD interpolators. With the new interpolators, the amount of geometric information transferred from the CAD system to the CNC system is reduced by orders of magnitude. Moreover, the contour errors caused by the new interpolators are much smaller than those caused by conventional CAD interpolators.

Keywords: interpolators, computerized numerical control, curves

Most computer aided design (CAD) systems provide the designer with tools for defining 2D and 3D curves and surfaces. In contrast, conventional computerized numerical control (CNC) machines generally support only the functions of straight line and circular interpolations [see, for example, References 1–4]. This gap between the well developed theory of representing curves and surfaces in CAD systems and the limited capabilities offered by numerical control interpolators imposes severe problems in relation to the rapid and accurate machining and cutting of surfaces and curves, indicating an obvious need for a general curve interpolator for CNC machines.

In this paper, various possibilities for implementing curve interpolators on CNC machines are analysed, and new schemes are offered. First, the method currently used for machining curves is discussed, and the difficulties associated with it are explored. Then, the need for a realtime interpolator for general curves is presented. Next, the dependencies of interpolation schemes on the methods for representing curves in a CAD system are discussed. Finally, two new schemes are proposed for realtime interpolators for general curves.

Figure 1 shows the current method for machining a part. A CAD system is used to define the geometry of the part, and the CNC machine must then drive the machine tool to machine this geometry. The part geometry is transferred to the CNC machine by means of a part program. The motion commands of the part program must be processed in realtime by the CNC interpolator in order to generate the reference commands for the control loops for execution (i.e. the driving of the machine tool).

The motion commands must fit the interpolator's capabilities so that the interpolator can process them in realtime. That is, a typical CNC interpolator can process only straight line and circular arc motion commands. Thus, to drive the machine tool along a curve, the CAD system breaks the curve into a set of line segments which approximate the curve to a desired accuracy (i.e. tolerance). Each line segment, in turn, is processed by the CNC's linear interpolator. The linear interpolator operates at sampling intervals of T s. At each iteration, the tool position command is incremented by VT mm, where V is the desired feedrate in millimetres per second. (Internally, basic length units (BLUs) are used instead of millimetres, and units of 0.01 mm and 0.001 in are common.) Thus, in order to machine a line of a length of l BLU at a feedrate of V BLU/s, the line is divided by the CNC interpolator into increments of VT BLU each. (Note that the division into increments is inherent in the CNC interpolation method, and is independent of the programmed acceleration or deceleration).

In the general case, however, the length of the line l is not an integer multiplication of VT. Therefore, the last increment is usually shorter than VT BLU. This last increment is also machined at T s, and this thus reduces the overall average feedrate along the line. This effect is practically negligible when 'normal' straight lines are machined. However, it becomes significant when the line is very short, as is the case in curve segmentation. In these cases, the contribution of the last and shorter increment does significantly affect the feedrate along the line.

The current method for machining curves creates a conflict in relation to the number of line segments into which a curve should be divided by the CAD system. On the one hand, the number of segments should be maximized, for two reasons:

- to better approximate the curve and reduce the contour error,
- to minimize the effect of segmentation, which causes discontinuities in the first derivatives along the path; these discontinuities, in turn, lead to deterioration of the smoothness of curves and surfaces, and this necessitates additional treatment (e.g. polishing).

Faculty of Mechanical Engineering, Technion, Haifa, Israel
*Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA
Paper received: 2 May 1993. Revised: 6 October 1993

0010-4485/94/11/0832–07 © 1994 Butterworth-Heinemann Ltd

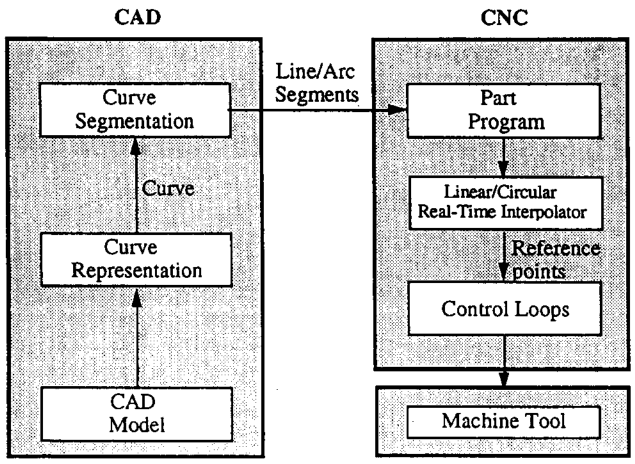


Figure 1 Current method for machining of curves

On the other hand, segmentation of the curve into many line segments causes the following problems:

- The average tool feedrate does not reach the desired feedrate V because of
  - the accumulated effect of the reduction in the feedrate in the last iteration of each line segment,
  - the machining of short lines, during which the tool may never reach the desired feedrate V because of the automatic acceleration and decleration applied at the beginning and end of each segment by the control loops.

  It has been suggested by Vickers and Bradley[5], on the basis of experimental measurements, that owing to the start/stop (acceleration/deceleration) effects, the full cutting speed is achieved for only 10% of the machining time. The result is that the feedrate along the curve is not constant, which in turn causes deterioration of the surface finish (during milling) or part dimension (during laser cutting). In addition, the machining time is increased because the mean feedrate is less than the desired rate.
- The CNC machine's memory is very small (and very expensive) in relation to the number of segments that must be stored for a part with complex surfaces (recall that the loading of the part program is an offline procedure).
- The communications load between the CAD system and the CNC machine, which may cause errors, should be reduced.

In practice, because of memory and communications constraints, the approach of minimizing the number of segments has been adopted by industry. The minimum number of segments is dictated by the permitted tolerance, and it depends on the curvature and length of the curve. However, the resulting discontinuities along the curve require additional treatment. Note that, even with the minimization approach, a huge number of segments is still needed in order to maintain a reasonable tolerance (e.g. 10 μm). Thus the current method is not adequate for the machining of curves and surfaces.

Consequently, a new method needs to be developed which will allow a high and constant feedrate along the curve. This concept is shown in *Figure 2*. The proposed approach is based on a new interpolator which can interpolate general curves in realtime, enabling the CAD system to transfer only information about the curve to the CNC machine; the CNC machine then interpolates it using the realtime curve interpolator.

Several researchers have developed realtime interpolators for curve generation. Koren[2] proposed an interpolator for standard parabolic curves; Sata *et al.*[6] developed a realtime interpolator for cubic Bézier curves which can, in fact, interpolate any cubic function. This interpolator can, of course, interpolate general parabolic curves by using only a 2nd-order Bézier curve. Makino[7,8] developed special interpolators for high-speed machines. Stadelmann[9] proposed a high-order interpolator for complex spatial geometry that requires a given velocity profile. In this interpolator, the transition between two successive segments is continuous. However, this interpolator does not guarantee that all interpolated points lie on the curve. A significant contribution has been made by Chou and Yang[10,11], who proposed an accurate offline interpolator for curves represented in their parametric forms. On the basis of their proposition, a realtime version of the interpolator was proposed by Huang and Yang[12,13], who solved the parametric interpolation (in their Equation 20) in the form of $u = g(t)$ by using the Euler method. The result they obtained is similar to Equation 23 in this paper.

In the following sections, two new approaches for realtime reference-word interpolators are proposed. These interpolators are capable of dealing with general curves. We assume that the given curve is the curve along which the centre of the tool must be driven. The proposed schemes are differentiated by the method used to represent the curve. The first scheme deals with the implicitly defined curve $f(x, y) = 0$, and the second scheme deals with the parametric curves $\mathbf{r}(u) = x(u)\mathbf{i} + y(u)\mathbf{j}$. (The extension to 3D is also discussed.)

The task of the realtime interpolator for a general curve can now be specified. Given (a) a curve $\mathbf{r}(u)$ or $f(x, y) = 0$, (b) the desired feedrate V mm/s along the curve, (c) the sampling time interval T s, and (d) the current reference point on the curve $R_k(x_k, y_k)$, find the next reference point to which the tool should be driven in T s (one sampling interval) while maintaining a constant feedrate V. This implies that (a) the next reference point $R_{k+1}(x_{k+1}, y_{k+1})$ must lie on the desired curve and must obey $\|R_{k+1} - R_k\| = VT$, and (b) the motion from R_k to R_{k+1}

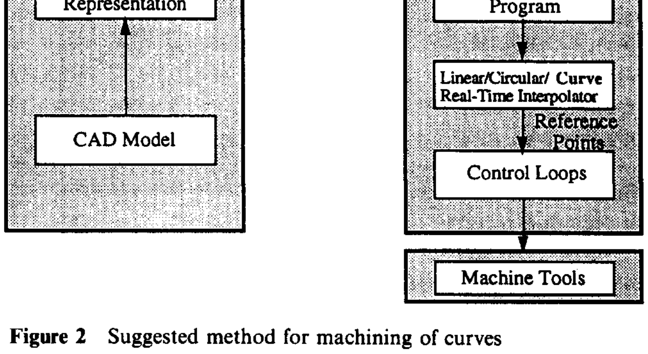


Figure 2 Suggested method for machining of curves

is of a point to point type and can therefore be executed by the existing linear interpolators available on CNC machines.

The case of implicit representation is analysed first; then, the case of parametric representation is presented and compared with that of the conventional interpolator.

INTERPOLATOR FOR IMPLICIT CURVE REPRESENTATION

A 2D curve along which the tool centre must be driven is frequently given implicitly by

$$f(x, y) = 0 \tag{1}$$

Let us examine the possible direct approaches to implementing a realtime interpolator for general curves as discussed in the above section.

Direct approaches

Assume that the current reference point $R_k(x_k, y_k)$ lies on the curve itself. The next reference point R_{k+1} must lie on the curve as well, and thus

$$f(x_{k+1}, y_{k+1}) = 0 \tag{2}$$

and the distance between the current reference point and the next one is specified, and thus

$$(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2 = (VT)^2 \tag{3}$$

Consequently, in order to obtain the next reference point $R_{k+1}(x_{k+1}, y_{k+1})$, two simultaneous equations, Equations 2 and 3, must be solved. The solution of these two equations, in general, must be iterative. However, iterative methods may need a large amount of computational time, and they are therefore appropriate only for offline CAD systems, but not for realtime CNC systems.

An alternative approach might be one that is based on calculating the intersection point between the curve and a circle of radius VT whose centre is at R_k. If the circle is represented parametrically, the following equations must be solved:

$$f(x_{k+1}, y_{k+1}) = 0 \tag{4}$$

$$x_{k+1} = x_k + VT \cos \theta_k \tag{5}$$

$$y_{k+1} = y_k + VT \sin \theta_k \tag{6}$$

Equations 5 and 6 can be substituted into Equation 4, resulting in a trigonometric equation in θ_k, $F(\theta_k) = 0$. When θ_k is obtained, it can be substituted into Equations 5 and 6 to obtain x_{k+1} and y_{k+1}. The solution of $F(\theta) = 0$ depends on the specific function $F(\theta)$, which is, in general, complicated, and must be solved iteratively. Although it will usually be easier to solve this equation rather than Equations 2 and 3, the solution will still not be suitable for realtime interpolation.

For the general implicit representation $f(x, y) = 0$, approximation methods that fit realtime interpolation may be proposed. One such approach has been suggested by Lo[14]. The proposed approach abandons the requirement that the next reference point must lie on the curve. Instead, it requires that R_{k+1} should be in the neighbourhood of $f(x, y)$, and that the deviation will be bounded and predictable.

Proposed interpolation method

Lo's interpolation method for implicit functions is based on approximating the nonlinear curve to a circle whose radius is equal to the radius of curvature ρ_k of the curve at the sampled point R_k. The method generates the next reference point R_{k+1} on the basis of the current reference point R_k and the geometric information, as shown in *Figure 3*. The motion displacement is split into two components: VT in the tangent direction, and $\overline{AB}$. The magnitude VT is given, and the magnitude of $\overline{AB}$ is determined below, where point A is a point along the tangent line whose distance from R_k is VT, and point B is the intersection between the line connecting A and C_k (the centre of curvature) and the circle whose centre is at C_k.

The geometric relation shown in *Figure 3* results in the following equations:

$$\sin \alpha_k = \frac{VT}{\rho_k + \overline{AB}} \tag{7}$$

$$\cos \alpha_k = \frac{\rho_k}{\rho_k + \overline{AB}} \tag{8}$$

Combining Equations 7 and 8 yields the following equation:

$$2\rho_k \overline{AB} + \overline{AB}^2 = (VT)^2 \tag{9}$$

Since the increments VT are small, α is also small. This implies that $\overline{AB} \ll \rho_k$ (typical values are $\rho_k = 20$ mm and $\overline{AB} = 0.005$ mm). Thus, Equation 9 can be rewritten as

$$\overline{AB} = \frac{(VT)^2}{2\rho_k} \tag{10}$$

In the proposed interpolation method, the magnitude VT is the incremental component in the tangent direction $\mathbf{t}_k$. The magnitude $\overline{AB}$ is used as the incremental component in the normal direction $\mathbf{n}_k$. Consequently, we have the

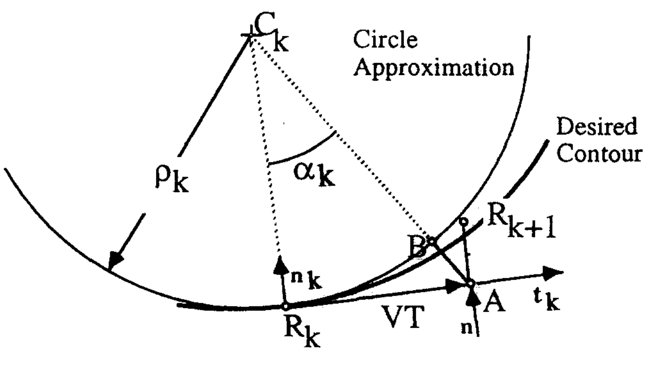


Figure 3 Geometric relation between current and next reference positions

following equation:

$$R_{k+1}=\begin{bmatrix}x_{k+1}\\y_{k+1}\end{bmatrix}=\begin{bmatrix}x_k\\y_k\end{bmatrix}+VT\begin{bmatrix}t_x\\t_y\end{bmatrix}_k+\frac{(VT)^2}{2\rho_k}\begin{bmatrix}n_x\\n_y\end{bmatrix}_k \qquad (11)$$

where R_k and R_{k+1} are two successive reference positions along the curve, $\mathbf{t}_k$ and $\mathbf{n}_k$ are the unit tangent and unit normal vectors, and ρ_k is the radius of the curvatuve at point R_k. The values of $\mathbf{t}_k$, $\mathbf{n}_k$ and ρ_k can be calculated at each step using the following equations:

$$\mathbf{t}_k=\begin{bmatrix}t_x\\t_y\end{bmatrix}_k=\begin{bmatrix}\dfrac{1}{\left(1+\left(\dfrac{dy}{dx}\right)^2\right)^{1/2}}\\ \dfrac{\dfrac{dy}{dx}}{\left(1+\left(\dfrac{dy}{dx}\right)^2\right)^{1/2}}\end{bmatrix}_k \qquad (12)$$

$$\mathbf{n}_k=\begin{bmatrix}n_x\\n_y\end{bmatrix}_k=\begin{bmatrix}-t_y\\t_x\end{bmatrix}_k \qquad (13)$$

$$\rho_k=\frac{\left(1+\left(\dfrac{dy}{dx}\right)^2\right)^{1.5}}{\dfrac{d^2y}{dx^2}} \qquad (14)$$

Discussion

Accurate realtime interpolation of curves represented implicitly as $f(x, y)=0$ cannot be guaranteed because, as mentioned above, the solution may need to be iterative. However, implicit smooth functions can be interpolated in realtime, on the basis of Equations 11–14, when the curves are approximated. In this case, the maximum contour error along the curve is[14] given by

$$E_{\max}<\frac{(VT)^2}{2}\left(\frac{1}{\rho_{\min}}-\frac{1}{\rho_{\max}}\right) \qquad (15)$$

which is typically of the order of 1 μm. Note, however, that the contour error is zero along straight lines ($\rho_{\min}=\rho_{\max}\rightarrow\infty$) and circles ($\rho_{\min}=\rho_{\max}$).

The following should be noted when implementing the proposed scheme:

- The derivative dy/dx can be evaluated by either coding its analytical expression in the source code, or using numerical procedures. Using the analytical expression is advantageous, and it should be used for common functions such as conic sections.
- The value of dy/dx tends to infinity when the tangent is vertical.

These cases (in which $x_{k+1}\rightarrow x_k$) should be identified, and the unit tangent vector should be set to $\mathbf{t}_k=[0, 1]^T$. Also, extension of the interpolation schemes of implicit representation to 3D is complicated, since, in the 3D case, a curve is defined as the intersection between two 3D surfaces $f(x, y, z)=0$ and $g(x, y, z)=0$. This is complicated to solve in the general case. To overcome the need for approximating the curve, to allow easy extension to 3D, and to avoid the difficulties associated with calculating dy/dx, parametric equations can be used.

Example

A parabolic contour was simulated with a feedrate of $V=20$ mm/s (47.2 in/min) and a sampling period of $T=0.01$ s. The parabolic equation is $y=0.05x^2$, where x and y are given in millimetres, and the tool moves from $(-20, 20)$ to $(20, 20)$.

The simulation was performed using two methods: the method using the interpolator proposed above, and the conventional method in which the contour is divided into many small segments, the number of which depends on the desired tolerance. The simulation results for the proposed interpolator are shown in *Figure 4*. The maximum contour error is given in *Table 1*.

With the proposed interpolator, only the start point and endpoint of the parabola and its parameters (e.g. 0.05) are transferred from the CAD system to the CNC machine. With the conventional method, a huge number of short line segments is needed. In addition to the large communications and storage loads, the large number of lines has another drawback. The feedrate along the curve is not constant. When 60 line segments are used, for example, the average length of each segment is 0.8 mm. Since, at each sampling period, the tool traverses $VT=0.2$ mm, each of the 60 segments is completed on average during only four sampling periods. In practice, this means that the tool never reaches the required feedrate. In contrast, as seen in *Figure 4*, the maximum feedrate error with the proposed interpolator is only 0.005%; in practice, there are no feedrate deviations.

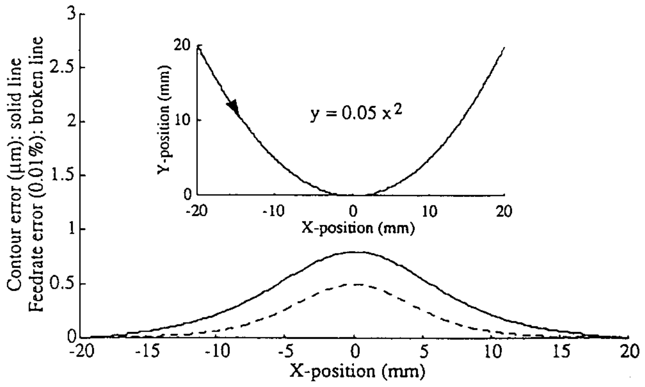


Figure 4 Simulation of proposed interpolator for parabola [——: contour error, – – –: feedrate error.]

Table 1 Maximum contour errors for example

Interpolation method	Maximum contour error, μm
Proposed	0.8
Conventional, 30 segments	22.0
Conventional, 60 segments	5.5
Conventional, 160 segments	0.8

INTERPOLATOR FOR PARAMETRIC CURVE REPRESENTATION

An alternative way of describing curves which treats the x and y axes symmetrically is the parametric form

$$x = x(u)$$
$$y = y(u) \tag{16}$$

where u is an arbitrary parameter, and usually $0 \leqslant u \leqslant 1$. Most CAD/CAM systems use parametric forms to represent curves. The parametric form is very convenient for controlling multiaxis machine tools, where each axis is individually driven. Furthermore, the parametric form enables x and y to be directly calculated as functions of u, and the extension from 2D to 3D is straightforward.

The straightforward approach for realtime interpolators might be the one used in CAD systems. Increment the parameter u uniformly, i.e. in equal small increments Δu, and calculate the corresponding $x_{k+1} = x(u_{k+1})$ and $y_{k+1} = y(u_{k+1})$ at each sampling period T.

This approach, however, has two main drawbacks:

- The length of the steps Δs_k, where $\Delta s_k = ((x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2)^{1/2}$, is not equal, but the tool transfers them in equal time intervals T. Consequently, the feedrate along the curve is not constant. This results in surface finish variations and an unnecessarily longer machining time.
- The optimal size of the increment Δu is not known. If Δu is too small, the resultant Δs_ks are too small as well, and the system slows down. If Δu is too big, the resultant Δs_ks are too big, and the position accuracy is not maintained.

As a consequence, a realtime interpolator cannot be based on a uniform segmentation of the curve according to u.

The key idea for realtime interpolators of curves represented in parametric form is that the segmentation should be based on curve segments of equal length rather than equal Δus. Accordingly, the proposed method determines successive values of u such that the curve segments Δs_k (machined at each sampling period T) are constants, which, in turn, guarantees a constant feedrate (i.e. velocity).

The feedrate $V(u)$ along the curve is defined by

$$V(u) = \frac{\mathrm{d}s}{\mathrm{d}t} = \left(\frac{\mathrm{d}s}{\mathrm{d}u}\right)\left(\frac{\mathrm{d}u}{\mathrm{d}t}\right) \tag{17}$$

or

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{V}{\mathrm{d}s/\mathrm{d}u} \tag{18}$$

where

$$\frac{\mathrm{d}s}{\mathrm{d}u} = ((x')^2 + (y')^2)^{1/2} \tag{19}$$

$$x' = \frac{\mathrm{d}x}{\mathrm{d}u}$$

$$y' = \frac{\mathrm{d}y}{\mathrm{d}u}$$

Substituting Equation 19 into Equation 18 yields

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{V}{(x'^2 + y'^2)^{1/2}} \tag{20}$$

A solution of Equation 20 that is found in a short computation time is the heart of a realtime interpolator for curves given in parametric forms. The solution of Equation 20 for a constant V gives the required $u(t)$. However, because the solution of Equation 20 is difficult in the general case, we may use a recursive solution based on Taylor's expansion around $t = kT$:

$$u_{k+1} = u_k + T\dot{u}_k + (T^2/2)\ddot{u}_k + \text{higher order terms} \tag{21}$$

where $\dot{u}$ denotes $\mathrm{d}u/\mathrm{d}t$ and $\ddot{u}$ denotes $\mathrm{d}^2u/\mathrm{d}t^2$.

If T is very small and the curve does not have small radii of curvature, even a 1st-order approximation is adequate:

$$u_{k+1} = u_k + T\dot{u}_k \tag{22}$$

Substituting Equation 20 into Equation 22 yields the equation of the proposed interpolator:

$$u_{k+1} = u_k + \frac{VT}{(x_k'^2 + y_k'^2)^{1/2}} \tag{23}$$

This equation prescribes how the value of u_{k+1} can be calculated on the basis of the current value of u_k and the derivatives of the current position (x_k, y_k) with respect to the us. (Note that $x'(u)$ and $y'(u)$ are given explicitly.) Once the new value of u, u_{k+1}, has been obtained, it is substituted into Equation 16 to obtain the reference point x_{k+1}, y_{k+1}. This equation has been obtained independently by Huang[13] and Lo[14].

Discussion

In this case, the calculated reference points lie on the curve. The expression of the derivatives x_k' and y_k' (i.e. $(\mathrm{d}x/\mathrm{d}u)_k$ and $(\mathrm{d}y/\mathrm{d}u)_k$) should also preferably be expressed analytically in the code. In the parametric interpolator, the sensitive case is when both x_k' and y_k' are zero at the same time. However, this can only happen if the curve has a cusp and is thus not smooth, or when the parameterization is improper[15]. (In such a rare case, reparameterization will solve the problem.) This does not happen, for instance, when $x(u)$ and $y(u)$ are polynomials in u. For cases in which the curvatures are extremely small, Equation 21, rather than Equation 22, should be used. When Equation 22 is used, the computation load required to calculate u_{k+1} according to Equation 23 consists of two multiplications, two additions, one division and one square root. Once the new u_{k+1} is calculated, it should be substituted in the parametric equation of the curve (Equation 16). Since there is no limitation on Equation 16, the calculation of the new reference points is then dependent on the specific curve equation. An important advantage of the parametric representation is the straightforward extension to 3D.

Example

To evaluate the proposed interpolator, a 2D cubic polynomial curve was simulated and compared with the conventional method, which approximates this curve by line segments for which $\Delta u = \text{constant}$. The evaluation is given below.

The 2D cubic curve ($V = 1.2$ mm/min, $T = 0.01$ s) is given by

$$\begin{cases} x = 11.9u^3 - 29.8u^2 + 32.9u + 5.0 \\ y = 47.6u^3 - 41.7u^2 + 16.55u + 2.5 \end{cases} \quad 0 \leqslant u \leqslant 1$$

The contour shape of this 2D curve is shown in *Figure 5*. The variation of u as a function of t, which is the solution of Equation 23, is shown in *Figure 6*. This result is the basis of the proposed realtime interpolator. The proposed interpolator results in very small contour errors compared with those of conventional CAD interpolators, as shown in *Figure 7* and *Table 2*. The segmentation in the conventional CAD interpolator, based on small equal Δu (e.g. $\Delta u = 0.033$ for 30 segments), results in correspondingly short Δs segments (e.g. on average of 1 mm). The short segments cause feedrate errors. As expected, increasing the number of segments results in smaller contour errors, but it increases the average deviation from the programmed V ($V = 20$ mm/s in this example).

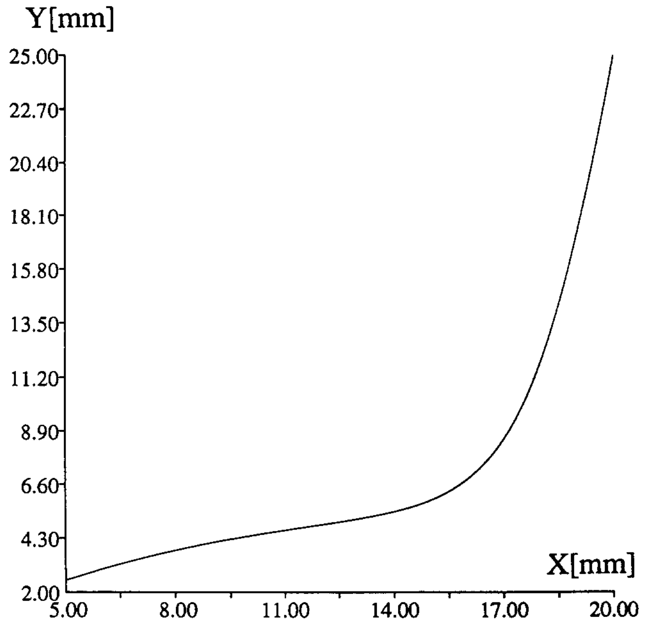


Figure 5 2D cubic polynomial curve

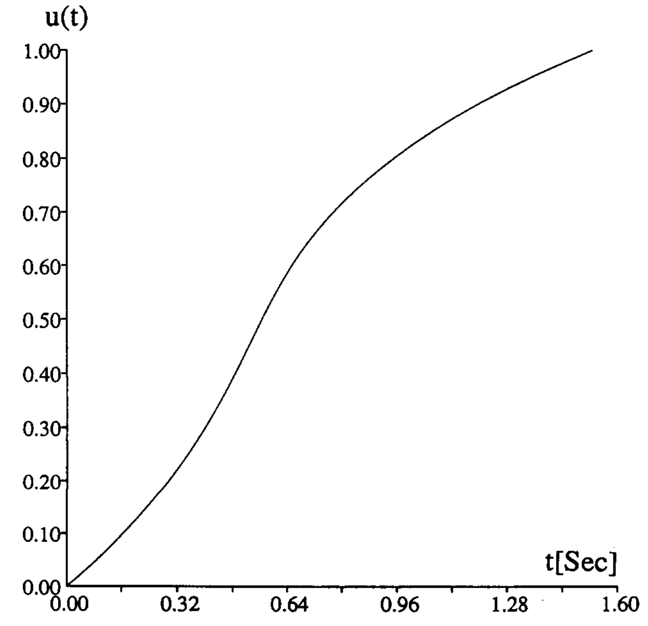


Figure 6 Variation of parameter u with time

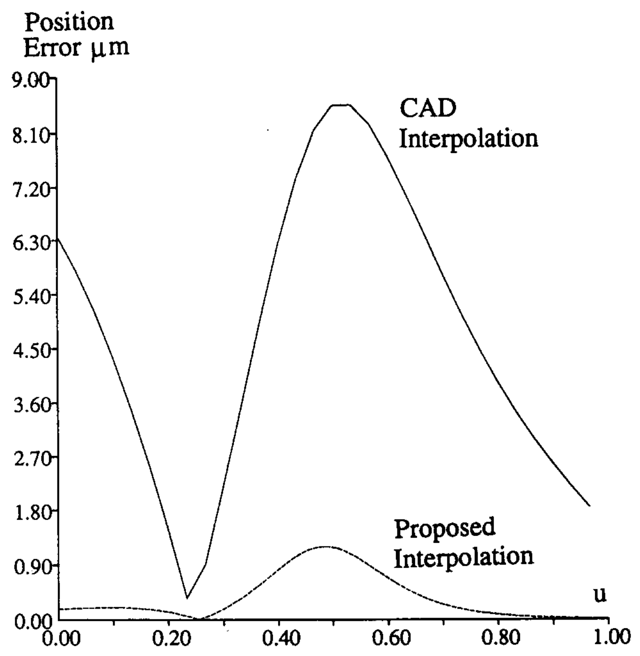


Figure 7 Contour errors caused by proposed interpolator and CAD interpolator

Table 2 Interpolation of 2D cubic curve

Interpolator	Realtime	CAD, 30 segments	CAD, 10 segments
ε_{max}, μm	1.20	8.80	77.00
Average V, mm/s	19.98	18.14	19.50

CONCLUSIONS

The amount of geometric information transferred from the CAD system to the CNC machine must be minimized, but it must still enable any general curve to be machined. These conflicting requirements can be satisfied only by the development of realtime interpolators for general curves. Curves may be presented as either implicit functions or parametric forms. Accordingly, two new types of CNC interpolator which can handle general curves have been introduced. The only assumption made is that the first derivatives exist, as is the case for smooth curves. The performances of these interpolators have been evaluated in terms of precision and feedrate deviations.

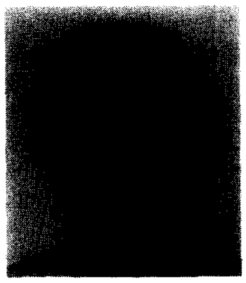
ACKNOWLEDGEMENTS

This research was partially sponsored by the US National Science Foundation grant DDM-9114131, and by the

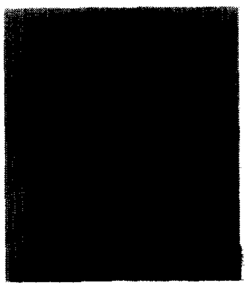
Japan TS Research Fund through the Technion, Haifa, Israel, Vice President for Research grant 033-0861. The paper was written while Y Koren was a Lady Davis Visiting Professor at the Technion, Haifa, Israel. The authors acknowledge the support of the Lady Davis Foundation.

REFERENCES

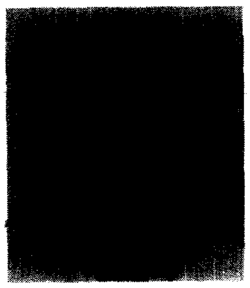
1 Koren, Y *Computer Control of Manufacturing Systems* McGraw-Hill, USA (1983)
2 Koren, Y 'Interpolator for a computer numerical control system' *IEEE Trans. Comput.* Vol C-25 No 1 (1976) pp 32-37
3 Koren, Y and Masory, O 'Reference-word circular interpolators for CNC systems' *Trans. ASME J. Eng. Indust.* Vol 104 (Nov 1982) pp 400-405
4 Papaioannou, S G 'Interpolation algorithms for numerical control' *Comput. Indust.* Vol 1 (1979) pp 27-40
5 Vickers, G W and Bradley, C 'Curved surface machining through circular arc interpolation' *Comput. Indust.* Vol 19 (1992) pp 329-337
6 Sata, T, Kimura, F, Okada, N and Hosaka, M 'A new method of NC interpolation for machining the sculptured surface' *Ann. CIRP* Vol 30 No 1 (1981) pp 369-372
7 Makino, H 'Clothoidal interpolation — a new tool for high-speed continuous path control' *Ann. CIRP* Vol 37 No 1 (1988)
8 Makino, H and Ohde, T 'Motion control of the direct drive actuator' *Ann. CIRP* Vol 40 No 1 (1991) pp 375-378
9 Stadelmann, R 'Computation of nominal path values to generate various special curves for machine' *Ann. CIRP* Vol 38 No 1 (1989) pp 373-376
10 Chou, J J and Yang, D C H 'Command generation for three-axis CNC machining' *Trans. ASME J. Eng. Indust.* Vol 113 (Aug 1991) pp 305-310
11 Chou, J J and Yang, D C H 'On the generation of coordinated motion of fine-axis CNC/CMM machines' *Trans. ASME J. Eng. Indust.* Vol 114 (Feb 1992) pp 15-22
12 Huang, J T and Yang, D C H 'A generalized interpolator for command generation of parametric curves in computer controlled machine' *Jap./USA ASME Flexible Automation — Vol 1* (1992) pp 393-399
13 Huang, J T 'Design and application of a new CNC command generator for CAD/CAM integration' *PhD Thesis* University of California at Los Angeles, USA (1992)
14 Lo, C C 'Cross-coupling control of multi-axis manufacturing' *PhD Thesis* University of Michigan, USA (1992)
15 Faux, I D and Pratt, M J *Computational Geometry for Design and Manufacture* Ellis Horwood, UK (1979)



Moshe Shpitalni received a BSc (1972), an MSc (1975) and a PhD (1980) in mechanical engineering from the Technion–Israel Institute of Technology. In 1983, after two years at Rensselaer Polytechnic Institute, USA, he joined the faculty of the Technion, where he is now an associate professor and head of the Laboratory for Interactive Computer Graphics and CAD. His research interests are in the applications of geometrical modelling and reasoning to manufacturing systems, CIM, automatic process planning, and CNC. He is particularly interested in the manufacture of sheet metal products and automatic assembly.



Yoram Koren received a BSc and an MSc in electrical engineering and a DSc (1970) in mechanical engineering from the Technion–Israel Institute of Technology. He is the Paul G Goebel Professor of Mechanical Engineering and Applied Mechanics at the University of Michigan, USA. He is the author of many publications on automated manufacturing, and the inventor of work covered by three US patents in robotics.



Chih-Ching Lo received a BS in power mechanical engineering from the National Tsing-Hua University, Taiwan, in 1984. He received an MS and a PhD in mechanical engineering from the University of Michigan, USA, in 1989 and 1992, respectively. He later joined the staff of the Mechanical Engineering Department, Feng-Chia University, Taiwan.