ERROR-FREE INTERPOLATION OF PARAMETRIC SURFACES

J.G. Gan
Nanyang Technological Institute
Singapore

T.C. Woo
Dept. of Industrial & Operations Eng.
The University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 89-35

December 1989

Revised February 1991

ERROR-FREE INTERPOLATION OF PARAMETRIC SURFACES*

J.G. Gan
Nanyang Technological Institute
Singapore

T.C. Woo†
University of Michigan
1205 Beal Ave.
Ann Arbor, MI 48109-2117

December 12, 1989
Revised: February 1991

Abstract

A Digital Differential analyzer (DDA) is a cost effective implementation for interpolating parametric curves and surfaces. However, the problems of register overflow and integer round-off have prevented it from being adopted by industry. Contrary to intuition, an increase in register capacity is shown to have no bearing on the register overflow problem. It is shown that there is a minimum number of interpolation steps K_ℓ below which register overflow occurs. Correspondingly, it is also shown that there exists a maximum number of interpolation steps K_u above which the accumulation of integer round-off errors exceeds the acceptable limit. But, when $K_\ell > K_u$, conflict arises. A solution is given to resolve the possible conflict and to yield an error-free interpolation of curves and surfaces.

*To appear in ASME Transactions, J. of Engineering for Industry

†Please send correspondence to T.C. Woo at the above address

1 INTRODUCTION

A numerical control (NC) machine executes according to a part program which is converted into a set of low-level instructions. One of the instructions is to pulse the axis motors such that the cutting tool moves, with respect to the workpiece, along a specified path by *interpolation.* The available NC interpoators have been linear and circular. Under linear interpolation, up to three linear axes move simultaneously such that a straight line results [Milner 76, Koren 76]. The circular interpolator is a little more restrictive. Only two linear axis motors are pulsed such that the resulting arc lies in one of the three principal planes [Koren 81].

More recently, some of the NC machine controller manufactures are beginning to offer spline interpolation, by first *approximating* a spline with a series of line segments, as illustrated in Figure 1a. The *error* ϵ of such an approximation is a function of the maximum length of the line segments ℓ and the minimum radius of curvature ρ [Faux 79]:

$$\epsilon = \frac{\ell}{8\rho}$$

Since the resolution of modern NC machines has been improved to as high as 0.1 μm, it is reasonable to expect shorter line segment lengths for spline interpolation. Indeed, the NC machine controller manufacturers, with the aid of high-speed 32-bit microprocessors and multi-master-bus architecture, offer linear interpolation on the linear approximation of a spline, as shown in Figure 1b. This paper offers *error-free* interpolation of spline curves and surfaces, as illustrated by Figure 1c. We say that the interpolation is error-free when the error is no greater than $\frac{1}{2}$ of the available resolution of the NC machine.

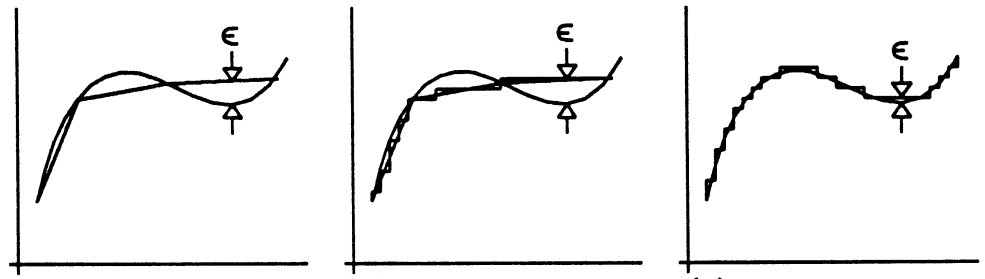

Figure 1 Approximation and Interpolation

It may be asked: Why don't the NC manufacturers approximate a curve to the resolution of the machine, hence achieving error-free interpolation by effectively by-passing linear approximation? Clearly, the error is dominated by approximation. As attempts of non-success are seldom reported, one can only surmise that there are difficulties. In this paper, we report the difficulties that have prevented successful implementation of error-free interpolation of curves and surfaces: *overflow* and *round-off* errors, as illustrated by Figure 2. We further present a solution that overcomes these two kinds of errors, using the Digital Differential Analyser as implementation.

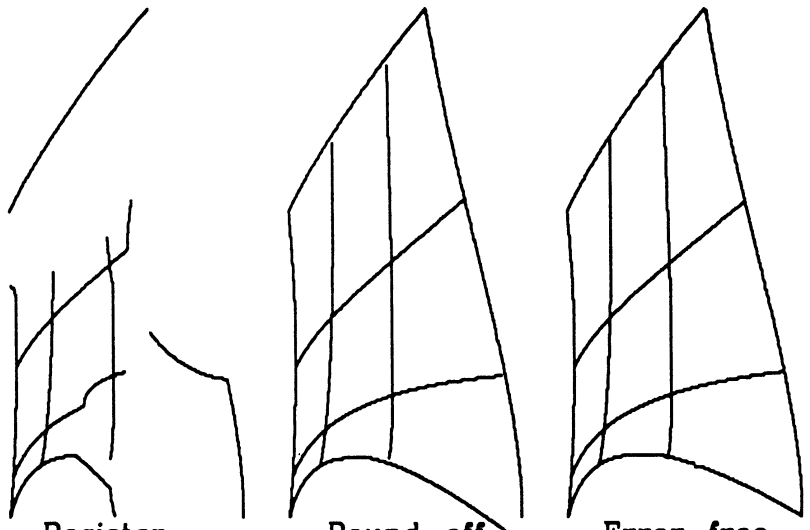

Figure 2 Interpolated Surface

The Digital Differential Analyzer (DDA) scheme can be used to perform interpolation of integral, differential, exponential, trigonometric and polynomial functions, etc. Previous work [Mayorov 64, Sizer 68] detailed how the interpolation of these functions could be implemented using hardware. Researchers [Koren 76, Milner 76, Koren 81] have introduced software implementation of the DDA technique for NC machine controller applications such as line and arc interpolations. In this paper we examine the use of DDA for interpolating

curves and surfaces of arbitrary degrees. We assume that a curve is represented parametrically as a polynomial in u.

$$\begin{aligned} X(u) &= a_0 + a_1 u + a_2 u^2 + \ldots + a_i u^i + \ldots + a_m u^m \\ Y(u) &= b_0 + b_1 u + b_2 u^2 + \ldots + b_i u^i + \ldots + b_m u^m \\ Z(u) &= c_0 + c_1 u + c_2 u^2 + \ldots + c_i u^i + \ldots + c_m u^m \end{aligned}$$

Stepping along the parameter $u \in [0, 1]$ traces the curve. A surface of two parameters, u and v, involves keeping one of the parameters constant while stepping along the other.

$$\begin{aligned} X(u,v) &= a_{00} + a_{10} u + \ldots + a_{i0} u^i + \ldots + a_{01} v + \ldots + a_{ij} u^i v^j + \ldots + a_{mm} u^m v^m \\ Y(u,v) &= b_{00} + b_{10} u + \ldots + b_{i0} u^i + \ldots + b_{01} v + \ldots + b_{ij} u^i v^j + \ldots + b_{mm} u^m v^m \\ Z(u,v) &= c_{00} + c_{10} u + \ldots + c_{i0} u^i + \ldots + c_{01} v + \ldots + c_{ij} u^i v^j + \ldots + c_{mm} u^m v^m \end{aligned}$$

Rational curves and surfaces are represented and traced similarly. Pushing the technology of DDA beyond lines and arcs offers the potential of high speed curve and surface interpolation with inexpensive hardware for numerical controllers and for computer graphics workstations. However, this is not without its difficulties.

The basic principle behind the DDA parametric interpolation scheme is the approximation of integration by additions. It can be viewed as a discrete version of the forward-differencing method [Ding 87, Foley 82], implemented using finite-capacity registers for data storage. Each interpolation *step* of a polynomial of degree m consists of the following successive addition operations:

$$\begin{aligned} \text{FOR } i: &= 1 \text{ TO } m \text{ DO} \\ (R_{m-i}) &\leftarrow (R_{m-i}) + (R_{m-i+1}) \end{aligned} \qquad (1)$$

where (R_{m-i}) represents the *content* of register R_{m-i}.

Figure 3 provides a schematic of the implementation of (1) for interpolating a cubic curve ($m = 3$) using four registers R_3, R_2, R_1 and R_0. The output of the interpolation for a polynomial function is in the form of overflow of the last register, R_0, in the cascade. It is used directly to instruct the motor to move a step in an open-loop control system, or to signal an error in the desired position in a closed-loop control system. Real-time computational speed achievable by DDA to perform the steps outlined in (1) is attributed to the usage of fixed-point variables and simple arithmetic operations such as increment, copy, compare and detect-overflow.

An examination of (1) reveals two problems in the implementation of the DDA parametric interpolator. Firstly, no register, other than R_0 should be allowed to overflow because if any of them does, the value stored is no longer valid. (An *overflow* results when the value stored exceeds the capacity.) Secondly, errors due to integer round-off of register values accumulate and propagate at an exponential rate [Kanatani 84]. These two problems were illustrated in Figure 2.

This paper is organized as follows. Section 2 discusses the problem of register overflow and recommends a solution to prevent it. Section 3 discusses the problem of integer round-off error and how the error can be controlled to within tolerance. Section 4 provides an algorithm to satisfy simultaneously the conditions for preventing register overflow and limiting round-off error. We conclude in Section 5.

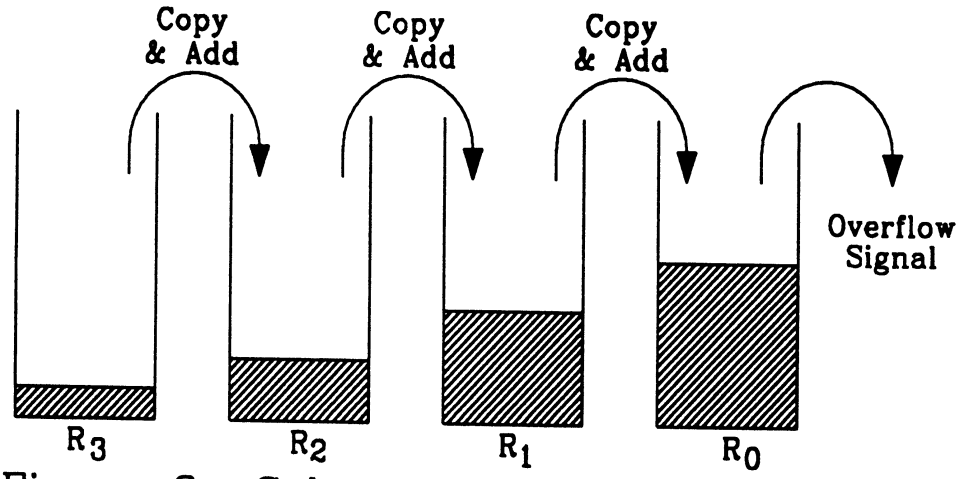

Figure 3 Schematic Drawing of DDA

2 REGISTER OVERFLOW

The conditions under which the registers overflow is first discussed. While intuition suggests that overflow is related to the capacity of a register, it is shown, however, that the register overflow problem cannot be overcome by increasing the capacities of registers. Specifically, there exists a lower bound on the number of interpolation steps which prevents register overflow.

2.1 Conditions for Overflow

The output from the DDA parmetric interpolator is the overflow signal from the last register, R_0, in the cascade. None of the other registers is allowed to overflow, otherwise the value stored would be incorrect, resulting in wrong interpolation results.

A DDA interpolates by performing successive copying of the content of register R_i, denoted by (R_i), and adding it to (R_{i-1}), the content of register R_{i-1}, as noted from (1). (R_{i-1}) is always greater than (R_i) for all $i > 1$. Hence, it follows that the register most likely to overflow is R_1. To prevent it from overflowing, we have to ensure that

$$(R_1)_{\max} < 2^n,$$

where n is the number of bits in R_1. Converely, the capacity of R_1 must be

$$n > \log_2 (R_1)_{\max}.$$

2.2 Register Capacity

It appears that a large n will prevent R_1 from overflowing. But, the following lemma shows that whether the R_1 register will overflow is independent of n.

Lemma 2.1 *Whether register R_1 overflows is independent of its capacity n.*

Proof: Let α_i be the initial value for register R_i. The content of R_1 after j integration steps is given by

$$(R_1) = \alpha_1 + j\alpha_2 + \frac{j(j+1)}{2!}\alpha_3 + \frac{j(j+1)(j+2)}{3!}\alpha_4 + \frac{(j+1)(j+2)(j+3)}{4!}\alpha_5 + \ldots \quad (2)$$

The values of the α's, their derivations given in Appendix I, are:

$$
\begin{aligned}
\alpha_5 &= \frac{5!2^n}{K^5 * D}(a_5 + \ldots) \\
\alpha_4 &= \frac{4!2^n}{K^4 * D}(a_4 + \ldots) \\
\alpha_3 &= \frac{3!2^n}{K^3 * D}(a_3 + \ldots) \\
\alpha_2 &= \frac{2!2^n}{K^2 * D}(a_2 - \frac{3a_3}{K} + \ldots) \\
\alpha_1 &= \frac{2^n}{K * D}(a_1 - \frac{a_2}{K} + \frac{a_3}{K^2} \ldots)
\end{aligned}
$$

where D = unit distance moved due to a single output pulse from DDA, and K = total number of steps to interpolate a curve. Substituting the values of the α's into (2), we have

$$
\begin{aligned}
(R_1) &= \frac{2^n}{K * D}\{a_1 + \frac{a_2}{K}(2j-1) + \frac{a_3}{K^2}(3j^2 - 3j + 1) + \frac{a_4}{K^3}(\ldots) + \frac{a_5}{K^4}(\ldots) + \ldots\} \\
&= 2^n * f(a_i, j, K, D) \qquad (3)
\end{aligned}
$$

where

$$
f(a_i, j, K, D) = \frac{1}{K * D}\{a_1 + \frac{a_2}{K}(2j-1) + \frac{a_3}{K^2}(3j^2 - 3j + 1) + \frac{a_4}{K^3}(\ldots) + \frac{a_5}{K^4}(\ldots) + \ldots\}. \qquad (4)
$$

R_1 overflows when $(R_1) \geq 2^n$. From (3), it follows that in order to prevent R_1 from overflowing, $f(a, j, K, D) < 1$, or

$$
\frac{1}{K * D}\{a_1 + \frac{a_2}{K}(2j-1) + \frac{a_3}{K^2}(3j^2 - 3j + 1) + \frac{a_4}{K^3}(\ldots) + \frac{a_5}{K^4}(\ldots) + \ldots\} < 1 \text{for all } j \in [0, K]. \qquad (5)
$$

Since $f(.)$ is independent of n, whether R_1 register will overflow is independent of n. This completes the proof. ∎

2.3 Analysis of Overflow

The proof of Lemma 2.1 shows that the overflowing of a register is a function of K, the number of steps for interpolating a curve the coefficients of the curve a_i, as well as D, the resolution of the machine tool. D and a_i are fixed. It is, therefore, interesting to investigate the lower bound on K so as to avoid the register overflow problem. Determination of such a lower bound for interpolating a polynomial funciton of any degree m is in order.

Lemma 2.2 *There exists a minimum number of interpolation steps K_ℓ above which register overflow does not occur.*

Proof: It suffices to consider the condition to prevent register R_1 from overflowing as given by (5), reproduced below.

$$\frac{1}{K*D}\{a_1+\frac{a_2}{K}(2j-1)+\frac{a_3}{K^2}(3j^2-3j+1)+\frac{a_4}{K^3}(\ldots)+\frac{a_5}{K^4}(\ldots)+\ldots\} < 1 \quad \text{for all } j\in[0,K]. \tag{5}$$

From (5), we seek a relation between K and the given variables D and a_i. Since each of the terms can be bounded as follows:

$$\begin{aligned}\frac{a_2}{K}(2j-1) &< 2a_2 \quad \text{if } a_2 \geq 0\\ &> 2a_2 \quad \text{otherwise}\\ \frac{a_3}{K^2}(3j^2-3j+1) &< 3a_3 \quad \text{if } a_3 \geq 0\\ &> 3a_3 \quad \text{otherwise}\end{aligned}$$

and so on for all $j \in [0,K]$, we have

$$\begin{aligned}&\text{POS}(a_1)+2\ \text{POS}(a_2)+\ldots+m\ \text{POS}(a_m) < K*D\\ &\text{NEG}(a_1)+2\ \text{NEG}(a_2)+\ldots+m\ \text{NEG}(a_m) > -K*D\end{aligned}$$

where

$$\begin{aligned}\text{POS}(x) &= x \quad \text{if } x \geq 0\\ &= 0 \quad \text{otherwise}\end{aligned}$$

and

$$\begin{aligned}\text{NEG}(x) &= -x \quad \text{if } x < 0\\ &= 0 \quad \text{otherwise}\end{aligned}$$

A tight condition for bounding K is

$$\begin{aligned}K > \frac{1}{D}\ \text{MAX} \quad &\{[\text{POS}(a_1)+2\text{POS}(a_2)+\ldots+m\text{POS}(a_m)],\\ &-[\text{NEG}(a_1)+2\text{NEG}(a_2)+\ldots+m\text{NEG}(a_m)]\}\end{aligned} \tag{6}$$

Let K_ℓ denote the term on the right of (6), we have

$$K_\ell = \frac{1}{D}\text{MAX}\left\{\left[\sum_{i=1}^{m} i\text{POS}(a_i)\right],\left[-\sum_{i=1}^{m} i\text{NEG}(a_i)\right]\right\} \tag{7}$$

which is bounded. This completes the proof that there exists a minimum number of steps K_ℓ above which no register overflow occurs. ∎

3 INTEGER ROUND-OFF

The source and propagation of round-off errors are now discussed. With the aid of numerical experiments, the control of such errors to within the tolerance limits is then established in Section 3.2.

3.1 Source and Propagation of Errors

Since the registers have finite capacities, there are representation errors. As integers are stored and the fractional parts are lost, there are the round-off errors at initialization. During each interpolation step, these round-off errors accumulate and propagate.

Let ϵ_i be the integer round-off error in register R_i when the register is initialized. After K steps, the error in R_0 register is given by

$$\epsilon_0 + K\epsilon_1 + \frac{K(K+1)}{2!}\epsilon_2 + \frac{K(K+1)(K+2)}{3!}\epsilon_3 + \ldots + \frac{K(K+1)\ldots(K+m-1)}{m!}\epsilon_m \quad (8)$$

where m is the degree of the curve. This well-known problem of exponential growth of cumulative error with the number of incremental steps illustrates the major weakness of conventional incremental approach [Kanatani 84].

Since every 2^n in R_0 corresponds to a single output pulse, the number of pulses ζ erroneously sent out due to the initial register round-off errors is given by

$$\zeta = [\epsilon_0 + K\epsilon_1 + \frac{K(K+1)}{2!}\epsilon_2 + \frac{K(K+1)(K+2)}{3!}\epsilon_3 + \ldots + \frac{K(K+1)\ldots(K+m-1)}{m!}\epsilon_m]^* 2^{-n} \quad (9)$$

We discuss in the next section the control of ζ.

3.2 Controlling Round-off Errors

In Appendix I, the derivation of the initial value, α_i, to be stored in register R_i shows that α_i is a function of the coefficient of the curve a's, number of interpolation steps K, and the register size n. Since ϵ_i is the error due to the integer round-off of α_i, ϵ_i is also a function of a's, K and n. From (9), we deduce that ζ is also a function of a's, as well as K and n.

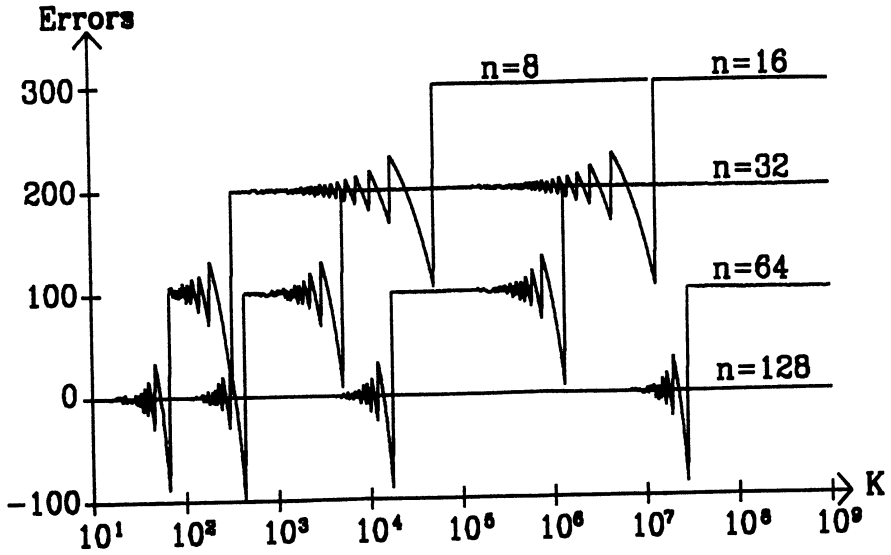

Figure 4 Round-off Errors

Consider a specific cubic function

$$x(u) = 100u + 100u^2 + 100u^3 \tag{10}$$

Figure 4 shows the number of erroneous pulses ζ verses the number of interpolation steps K for registers of sizes 8-bit, 16-bit, 32-bit, 64-bit and 128-bit. The plots suggest that there is an upper limit to the number of steps that can be used for DDA interpolator with different register capacities. In the case of interpolating the function (10), the upper limits for K are approximately 14, 75, 4073, 6760830 and some value greater than 10^9 when the sizes of the registers are respectively 8-bit, 16-bit, 32-bit, 64-bit and 128-bit. We seek a theoretical upper limit for K.

Lemma 3.1 *There exists a maximum number of interpolation steps K_u above which the accumulation of integer round-off errors exceeds the acceptable limit.*

Proof: Suppose $\frac{1}{2}$ is the acceptable number of erroneous pulses ζ. Since ϵ is due to integer round-off, it may assume any value between 0 and $\frac{1}{2}$. In the worst case, every ϵ is $\frac{1}{2}$. By substituting $\frac{1}{2}$ into every ϵ, and bounding ζ to $\frac{1}{2}$, the condition given by (9) becomes

$$1 + K + \frac{K(K+1)}{2!} + \frac{K(K+1)(K+2)}{3!} + \ldots + \frac{K(K+1)\ldots(K+m-1)}{m!} < 2^n.$$

Dividing both sides by K gives

$$1 + \frac{K+1}{2!} + \frac{K^2+3K+2}{3!} + \ldots + \frac{K^{m-1} + \frac{m(m+1)}{2}K^{m-2} + \ldots + (m-1)}{m!} < \frac{(2^n-1)}{K}$$

Rearranging in decreasing order of K yields

$$K^{m-1} + \frac{m(m+1)}{2}K^{m-2} + \ldots + (\ldots)K + (1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{m})m! < \frac{(2^n-1)m!}{K}, \text{ or}$$

$$K^{m-1}[1 + \frac{m(m+1)}{2K} + \ldots + \frac{(\ldots)}{K^{m-2}} + (1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{m})\frac{m!}{K^{m-1}}] < \frac{(2^n-1)m!}{K} \quad (11)$$

Since K is usually very large (in the order of 10^5) compared to m (the degree of the curve), we can safely assume that the first term in (11) is much greater that the second term, and so on. The terms within the square brackets in (11) sum to less than 2. The condition to restrict round-off error to 0.5 of an output pulse can therefore be simplified to

$$K^m < \frac{(2^n-1)m!}{2}$$

Therefore, the maximum number of integration steps K_u is given by taking the mth root,

$$K_u = \left(\frac{(2^n-1)m!}{2}\right)^{1/m}. \quad (12)$$

Since n, the number of bits, and m, the degree of the curve, are bounded, K_u is finite.
∎

In order to limit the number of erroneous pulses, care must be taken to ensure that the number of interpolation steps used is lower than K_u. If this upper limit is too low, overflow (as discussed in Section 2) can still occur. This is the subject of the next section.

4 COMBINED SOLUTION

Lemma 2.2 asserts that, to prevent overflow, the number of interpolation steps K must exceed a lower bound

$$K > K_\ell = \frac{1}{D} MAX \left\{ \left[\sum_{i=1}^{m} iPOS(a_i) \right], \left[-\sum_{i=1}^{m} iNEG(a_i) \right] \right\} \tag{7}$$

and Lemma 3.1 states that, to prevent round-off errors, K must not exceed an upper bound

$$K < K_u = \left(\frac{(2^n - 1)m!}{2} \right)^{1/m} \tag{12}$$

For error-free interpolation, both conditions must be satisfied simultaneously, i.e., $K_\ell < K < K_u$:

$$\frac{1}{D} MAX \left\{ \left[\sum_{i=1}^{m} iPOS(a_i) \right], \left[-\sum_{i=1}^{m} iNEG(a_i) \right] \right\} < \left(\frac{(2^n - 1)m!}{2} \right)^{1/m}$$

The necessary number of bits n to ensure no overflow and no more than $\zeta = \frac{1}{2}$ of an erroneous pulse is given by

$$n > 1 - \log m! + m \log \left[\frac{1}{D} MAX \left\{ \left[\sum_{i=1}^{m} iPOS(a_i) \right], \left[-\sum_{i=1}^{m} iNEG(a_i) \right] \right\} \right] \tag{13}$$

since $2^n \gg 1$. If ζ is $\frac{1}{2^q}$, then the "1" on the right hand side of (13) is replaced by q.

With condition (13), we are ready to summarize our findings in a procedural form.

PROCEDURE DDA - CURVE

Step 1 {*Determine the minimum K that ensures no register overflow problem*}

$$K \leftarrow \frac{1}{D} * MAX\{\sum_{i=1}^{m} iPOS(a_{ix}), \sum_{i=1}^{m} iPOS(a_{iy}), \sum_{i=1}^{m} iPOS(a_{iz}),$$
$$-\sum_{i=1}^{m} iNEG(a_{ix}), -\sum_{i=1}^{n} iNEG(a_{iy}), -\sum_{i=1}^{m} iNEG(a_{iz})\}$$

Step 2 {***Find n***}

$$n \leftarrow \lceil 1 - \log m! + m \log K \rceil$$

Step 3 {*Calculate initial register values*}

FOR each axis DO

$$(R_0) \leftarrow ROUND\ (2^n/D\ *\ (a_0 - D\ *\ TRUNC\ (a_0/D)))$$
$$(R_1) \leftarrow ROUND\ (2^n/(K * D)\ *\ (a_1 - a_2/K + a_3/K^2))$$

$(R_2) \leftarrow ROUND\ (2!\ 2^n/(K^2 * D)\ *\ (a_2 - 3a_3/K))$
$(R_3) \leftarrow 3!\ 2^n/(K_3 * D)\ *\ a_3$
ENDDO

Step 4 {*Position motor*}
*Move motor to position (D * TRUNC (a_0/D))*

Step 5 {*Generating Output Pulses*}
For i := 1 TO K DO
Wait for a system clock pulse
FOR each axis DO
$(R_2) \leftarrow (R_2) + (R_3)$
$(R_1) \leftarrow (R_1) + (R_2)$
$(R_0) \leftarrow (R_0) + (R_1)$
IF R_0 overflows THEN
Send an output pulse to motor
ENDDO
ENDDO
ENDDO {ELSE}
ENDPROCEDURE

PROCEDURE DDA - Surface
Approximate surface by a set of constant parametric curves
For each curve DO
Call DDA-Curve
ENDPROCEDURE

It may be noted that Procedure DDA-curve has two phases: a preprocessing phase (involving Steps 1 to 3 that need to be done only once for a given curve) and the actuation phase (Step 5 which is repetitive and arithmetically simple). The preprocessing phase can be done in software and the actuation phase implemented in hardware.

The interpolated surface in Figure 2 was generated entirely by software. The bicubic equations used for generating the surfaces plotted are as follows:

$$
\begin{aligned}
x(u,w) = {} & 100 + 50\,u + 100\,u^2 + 100\,u^3 \\
& + 0\,w + 50\,uw + 50\,u^2w - 100\,u^3w \\
& + 50\,w^2 - 50\,uw^2 + 0\,u^2w^2 - 200\,u^3w^2 \\
& - 50\,w^3 + 50\,uw^3 + 0\,u^2w^3 + 100\,u^3w^3
\end{aligned}
$$

$$
\begin{array}{lcrcrcrcrcrcr}
y(u,w) & = & 50 & & + & 100 & u & + & 50 & u^2 & + & 50 & u^3 \\
 & + & 0 & w & + & 50 & uw & + & 50 & u^2w & + & 200 & u^3w \\
 & + & 200 & w^2 & - & 50 & uw^2 & + & 0 & u^2w^2 & - & 100 & u^3w^2 \\
 & - & 50 & w^3 & + & 100 & uw^3 & + & 50 & u^2w^3 & - & 100 & u^3w^3 \\
z(u,w) & = & 100 & & + & 50 & u & + & 100 & u^2 & + & 100 & u^3 \\
 & + & 0 & w & + & 50 & uw & + & 50 & u^2w & - & 100 & u^3w \\
 & + & 50 & w^2 & - & 50 & uw^2 & + & 0 & u^2w^2 & - & 200 & u^3w^2 \\
 & - & 50 & w^3 & + & 50 & uw^3 & + & 0 & u^2w^3 & + & 100 & u^3w^3
\end{array}
$$

5 SUMMARY

The DDA has been shown to be an efficient scheme for interpolating not only lines [Milner 76] and arcs [Koren 81], but also parametric curves and surfaces of arbitrary complexity. However, because of the complexity, the two sources of errors, overflow and round-off, can no longer be overlooked.

It is shown in this paper that overflow has nothing to do with the register size n. It is also shown the round off error is a function of n (as well as m, the degree of the curve or surface). To overcome both sources of error, the choice of n is rationalized as the simultaneous satisfaction of two constraints on the number of interpolation steps K.

REFERENCES

1. Ding, Q. and B.J. Davis, "Surface Engineering Geometry for Computer-Aided Design and Manufacture," Ellis Horwood, UK, 1987.

2. Faux, I.D. and M. Pratt, "Computational Geometry for Design and Manufacture," Halsted Press, NY, 1979.

3. Foley, J.D., and A. VanDam, "Fundamentals of Interactive Computer Graphics," Addison-Wesley, Reading, MA, 1982.

4. Kanatani, K., "Errors of the Incremental Method for Curves," Comp Vis Grap I Porc, Vol. 26, pp. 130-133, 1984.

5. Koren, Y., "Interpolator for a Computer Numerical Control System," IEEE Trans. on Computers, Vol. C-25, No. 1, pp. 32-37, Jan. 1976.

6. Koren, Y., and O. Masory, "Reference-Pulses Circular Interpolators for CNC Systems," Trans. ASME, J. Eng. Ind., Vol. 103, No. 1, pp. 131-136, Feb. 1981.

7. Mayorov, F.V., "Digital Differential Analyzers," Iliffe Books, London, England, 1964.

8. Milner, D.A., "Some Aspects of Computer Numerical Control with Reference to Interpolation," Trans. ASME, J. Eng. Ind., pp. 883-889, Aug. 1976.

9. Press, W.H., B.P. Falnnery, S.A. Teukolsky and W.T. Vetterling, "Numerical Recipes-The Art of Scientific Computing," Cambridge U. Press, Cambridge, England, 1986.

10. Sizer, T.R., "The Digital Differential Analyser," Chapman & Hall, London, England, 1968.

APPENDIX I

Calculation of Initial Register Values

Consider the DDA interpolation of a parametric curve

$$X(u) = a_0 + a_1 u + a_2 u^2 + \ldots + a_m u^m \qquad (A1)$$

successive addition of the contents of the registers are accumulated in R_0. Now, suppose a total of K steps is necessary to complete the curve, then after the completion of K steps, the content of register R_0 is given by

$$\begin{aligned}(R_0) &= \alpha_0 + K\alpha_1 + \frac{K(K+1)}{2!}\alpha_2 + \frac{K(K+1)(K+2)}{3!}\alpha_3 + \frac{K(K+1)(K+2)(K+3)}{4!}\alpha_4 \\ &+ \frac{K(K+1)(K+2)(K+3)(K+4)}{5!}\alpha_5 + \ldots\end{aligned}$$

where α_i is the initial value of R_i.

Rearranging, we have

$$\begin{aligned}(R_0) &= \alpha_0 + \{\alpha_1 + \frac{\alpha_2}{2} + \frac{\alpha_3}{3} + \frac{\alpha_4}{4} + \frac{\alpha_5}{5} + \ldots\}K \\ &+ \{\frac{\alpha_2}{2!} + \frac{(1+2)}{3!}\alpha_3 + \frac{[(1*2)+(2*3)+(3*1)]}{4!}\alpha_4 + \ldots\}K^2 \\ &+ \{\frac{\alpha_3}{3!} + \frac{(1+2+3)}{4!}\alpha_4 + \frac{[(1*2)+(1*3)+(1*4)+(2*3)+(2*4)+(3*4)]}{5!}\alpha_5 + \ldots\}K^3 \\ &+ \{\frac{\alpha_4}{4!} + \frac{(1+2+3+4)}{5!}\alpha_5 + \ldots\}K^4 \\ &+ \{\frac{\alpha_5}{5!} + \ldots\}K^5 + \ldots + \{\ldots\}K^m \qquad (A2)\end{aligned}$$

By comparing (A1) and (A2), and bearing in mind the relationship that 2^n in the register R_0 corresponds to one unit distance (D) due to a single pulse from the DDA in the linear distance, we obtain

$$\begin{aligned}\frac{(\alpha_1 + \frac{\alpha_2}{2} + \frac{\alpha_3}{3} + \frac{\alpha_4}{4} + \frac{\alpha_5}{5} + \ldots)}{2^n}K * D &= a_1 \\ \frac{(\alpha_2 + \alpha_3 + \frac{11}{12}\alpha_4 + \frac{5}{6}\alpha_5 + \ldots)}{2!2^n}K^2 * D &= a_2 \\ \frac{(\alpha_3 + \frac{3}{2}\alpha_4 + \frac{7}{4}\alpha_5 + \ldots)}{3!2^n}K^3 * D &= a_3 \\ \frac{(\alpha_4 + 2\alpha_5 + \ldots)}{4!2^n}K^4 * D &= a_4 \\ \frac{(\alpha_5 + \ldots)}{5!2^n}K^5 * D &= a_5\end{aligned}$$

Solving for $\alpha_1, \alpha_2, \alpha_3$, and so on, we find the values to be

$$\begin{aligned}
\alpha_5 &= \frac{5!2^n}{K^5 * D}(a_5 + \ldots) \\
\alpha_4 &= \frac{4!2^n}{K^4 * D}(a_4 + \ldots) \\
\alpha_3 &= \frac{3!2^n}{K^3 * D}(a_3 + \ldots) \\
\alpha_2 &= \frac{2!2^n}{K^2 * D}(a_2 - \frac{3}{K}a_3 + \ldots) \\
\alpha_1 &= \frac{2^n}{K * D}(a_1 - \frac{a_2}{K} + \frac{a_3}{K^2} + \ldots)
\end{aligned}$$

The value of α_0 can be obtained by observing that the initial position is a_0 ideally. However, due to the fact that the position can only be in multiples of D, the actual initial position is only TRUNC$(a_0/D)^*D$, where TRUNC is a truncating operation where only the integral part of a real number is kept and the fractional part discarded. The balance of a_0 which is not translated into initial-position command is stored in register R_0 as α_0. The value of α_0 can be computed by using the following equation:

$$\alpha_0 = \frac{a_0 - D^*TRUNC(\frac{a_0}{D})}{D} * 2^n$$

In the case of a rational curve, the determination of the initial values for the registers in numerator and denominator are performed separately.