Jurnal Teknologi, 37(D) Dis. 2002: 67–78

REFERENCE PULSE CNC INTERPOLATOR BASED ON ENCLOSING LINE FOR 2D PARAMETRIC CURVE

ZAHURIN SAMAD[1] & WAN AZHAR WAN YUSOF[2]

Abstrak. Kertas kajian ini mengemukakan algoritma interpolasi CNC bagi lengkok parametrik umum 2D. Interpolasi yang dikemukakan ini menggunakan teknik rujukan-dedenyut ataupun teknik langkah-tambahan. Algoritma ini berasaskan kepada pencarian secara berterusan garisan-lingkungan yang bersilang dengan lengkok parametrik. Setelah garisan-lingkungan dikenalpasti, kawasan jarak-terdekat digunakan untuk memilih titik interpolasi tanpa melaksanakan pengiraan. Simulasi komputer secara *real-time* menunjukkan bahawa interpolasi yang dikemukakan ini telah meningkatkan halaju maksimum yang dibenarkan dan pada waktu yang sama mengekalkan kejituan yang tinggi.

Kata kunci: interpolator, CNC, rujukan-dedenyut, peralatan mesin.

Abstract. This paper proposes an efficient CNC interpolation algorithm for general 2D parametric curves. The proposed interpolator uses reference-pulse or incremental step technique. The algorithm is based upon finding the successive enclosing-lines that the parametric curve intersects. Once the intersected line is identified, the pre-defined closest distance region is used to select the interpolating point without resorting to calculation. Real-time computer simulations indicate that the proposed algorithm increases maximum allowable velocity while still maintaining high precision.

Key words: interpolators, computerized numerical control (CNC), reference pulse, machine tools

1.0 INTRODUCTION

Design parts are usually represented by Computer-Aided Design (CAD) system in parametric curves [1]. These curves are then approximated into successive cutter location (CL) paths based on tolerance requirements. These CL paths are in unit-less parametric form and thus need to be converted to time domain for machining. Using machining parameter such as cutting feed-rate, the CL paths are converted into successive motion trajectory path. Motion trajectory paths provide reference axial positions and velocities for each successive step to servo control loop for motion trajectory realization. This paper deals with the subject of converting CL paths into motion trajectory and is called interpolation or command generation.

Interpolation method is further divided into two different techniques namely "reference-word" and "reference-pulse" based on motion trajectory realization [2-6]. Reference-word or "sampled-data" technique uses digital value representing voltage

[1&2]Pusat Pengajian Kejuruteraan Mekanikal, Universiti Sains Malaysia, Kampus Kejuruteraan, 14300 Nibong Tebal Seberang Prai Selatan, P. Pinang. e-mel: zahurin@eng.usm.my

proportional to axial velocity to drive the servo systems. On the other hand, reference-pulse interpolation uses pulse frequency based on a single iteration of interpolation routine as a velocity command, in which each pulse increments one-step (one basic length unit or BLU) of motion. Reference-pulse interpolator is easier to implement but the maximum achievable feed-rate is limited by the interpolator execution time. Sampled-data technique implementation is more complex and the position error is larger [2,4,5]. For further information on interpolators that are based on sampled-data techniques, refer to references [7-11].

With the improvement of microprocessor clock-speed as well as efficient interpolator algorithms, the interpolator execution time for reference-pulse interpolator is getting shorter and therefore the maximum allowable feed-rate is increasing. Using reference-pulse interpolator technique, the maximum allowable feed-rate $V_{\max}$ is inversely proportional to the iteration time, T of a single iteration routine [4-6].

$$V_{\max} = f\,(\mathrm{BLU}) = \left(\frac{1}{T}\right)(\mathrm{BLU}) \tag{1}$$

where f is the output frequency and BLU is the machine basic-length-unit. Clearly, the smaller the iteration time T, the higher the maximum allowable feed-rate. For a given clock speed, the iteration time T is largely contributed by the interpolation calculation time in one iteration routine. This paper proposes an efficient interpolation algorithm for general 2D parametric curves that reduces interpolation calculation time while maintaining the same desired accuracy.

In incremental-curve generation method, there are two sets of points: the interpolating point and the point on curve. The interpolating point is the point where the cutting tool is being moved in discrete step while the point on curve is the point used as reference for choosing the next interpolating point. The objective of the interpolator algorithm is to find the next interpolating point from all of the possible interpolating points such that the distance between the next interpolating point and the curve is the shortest. The existing method proposed by [3] employ two steps: selecting the right quadrant by finding the derivative of the curve at the current point, and finding the orthogonal distance to the curve as the closest distance criteria. This approach however requires finding of three points on the curve and the lengths of each normal vector from each point to the corresponding interpolating point. The equations used for reference [3] are listed below. The prime symbol represents the derivative with times and the subscripts "i" and "o" represent the next and the current interpolating points respectively. The capital letter represents the interpolating point while the small letter represents the point on curve.

$$u_i = u_0 - \frac{(X_i - x_0)\,x_0' + (Y_i - y_0)\,y_0'}{(X_i - x_i)\,x_0'' + (Y_i - y_0)\,y_0'' - x_0'^2 - y_0'^2} \qquad i = 1,2,3 \tag{2}$$

$$|n_i|^2 = (X_i - x_i)^2 + (Y_i - y_i)^2 \tag{3}$$

Equation 2 (utilized to find the next parametric value of the curve) and Equation 3 (the shortest distance criterion) have to be evaluated three-times in every iteration for 2D curve By inspection, the maximum error is half of the BLU and that is when the curve is a straight line passing through the middle of any two interpolating points.

2.0 PROPOSED 2D PARAMETRIC CURVE REFERENCE-PULSE INTERPOLATOR

Referring to Figure 1, let a parametrically defined curved is given by

$$\mathbf{r} = x(u)\mathbf{i} + y(u)\mathbf{j} \qquad u_{\text{start}} \le u \le u_{\text{final}} \tag{4}$$

Let the current interpolating point be given by $P_k = (X_k, Y_k)$ and the step size be given by S, then the set of all possible next interpolating points is given by

$$p = \{p_i = (X_i, Y_i)\} \qquad i = 1, 2, 3, \ldots, 8 \tag{5}$$

The next interpolating point is then defined by

$$P_{k+1} = (X_{k+1}, Y_{k+1}) \qquad P_{k+1} \in p \tag{6}$$

The proposed interpolator is based on the following principles:

- The closest distance regions are defined as $A = \{A_1, A_2, A_3, \ldots, A_8\}$, whereby if the next point on curve r_{k+1} is known and falls into one of the regions, the next interpolating point is identified.
- The next points on curve r_{k+1} are determined from the intersection points between the curve and the enclosing lines. By using the intersection points the following conditions are met.
  - The shortest distance from the possible next interpolating points to r_{k+1} is within the maximum allowable error, $|P_{k+1} - r_{k+1}| = E_{\max}$. The proposed interpolator is targeted to have $E_{\max} \le 0.5$ of the step size S.
  - Point r_{k+1} does not fall into the NULL region in which there is no corresponding next interpolating point P_{k+1} (note that the NULL region is the region that contains the current interpolating point P_k and the region is not a subset of A).

- The intersected enclosing line L_{k+1} is determined by using two criteria: tangent direction vector and the geometric factor.
- Knowing the intersected enclosing line L_{k+1}, the value of u_{k+1} can be determined by using tangent line approximation method.
- Using the value of u_{k+1}, the values of $x_{k+1}, y_{k+1}, DX_{k+1}$ and DY_{k+1} are obtained.
- The values of DX_{k+1} and DY_{k+1} are then used to determine the next interpolating point P_{k+1} based on the closest distance region.

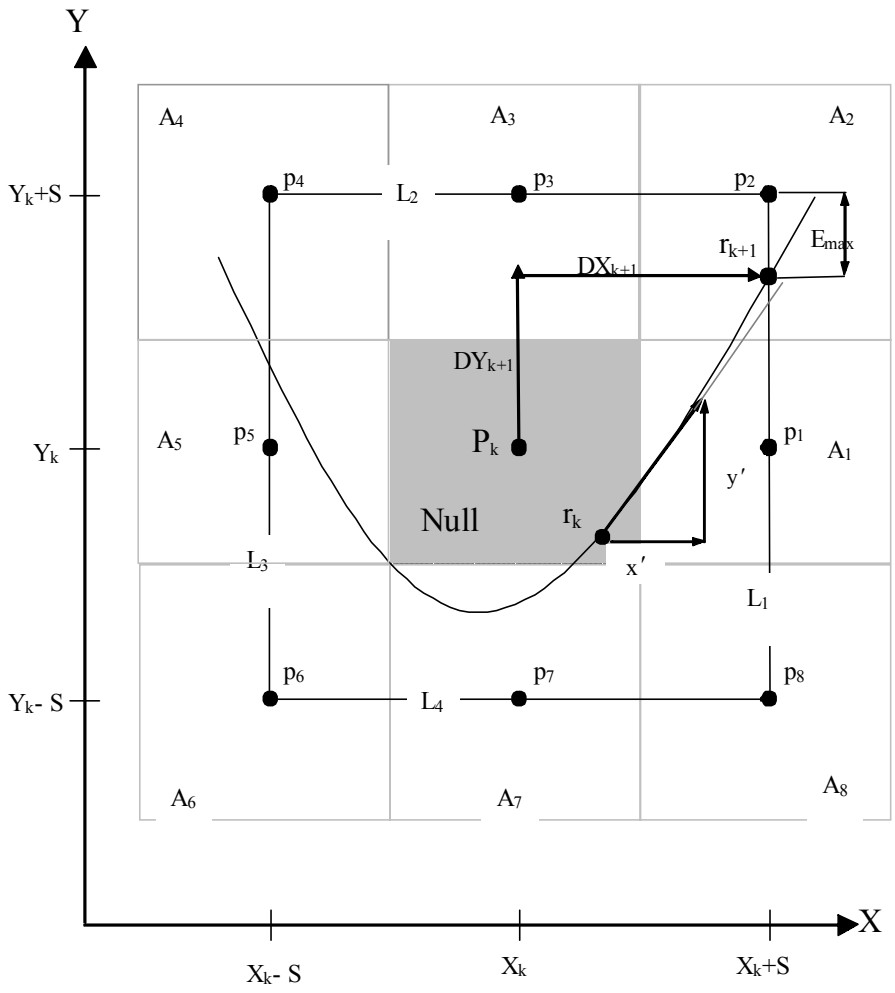
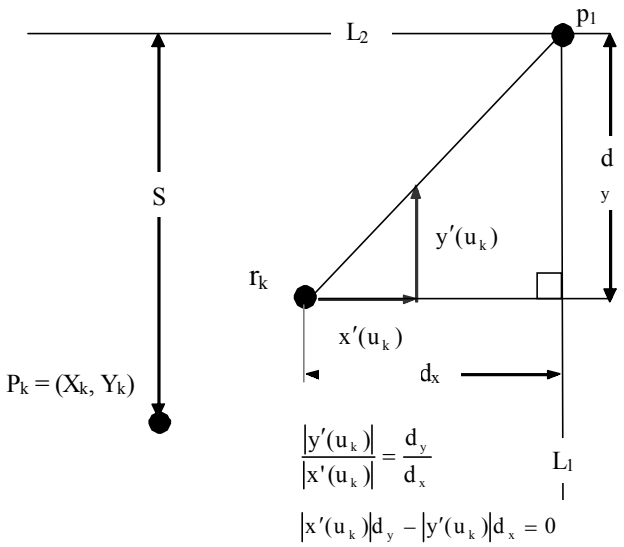


Figure 1 The proposed interpolation algorithm that uses the predefined closest-distance regions $\{A_1, A_2, A_3, \ldots, A_8\}$ and the intersected line $\{L_1, L_2, L_3, L_4\}$.



The values of d_x and d_y can be determined as follows:

Conditions	$\geq 0(+)$	$< 0(-)$
$x'(u_k)$	$(X_k + S) - x_k$	$x_k - (X_k - S)$
$y'(u_k)$	$(Y_k + S) - y_k$	$y_k - (Y_k - S)$

Figure 2 Using geometric factor to determine the intersected line.

Step 1: Finding L_{k+1} based upon tangent direction vector

Based upon the component of the tangent direction vector (the derivative along the X and Y axes), Table 1 below indicates that the number of possible intersected lines is reduced from four to two.

Table 1 Set of possible L_{k+1} due to tangent direction vector factor.

Tangent Vector Component Factor		**Direction**	**Set of Possible L_{k+1} due to tangent direction vector**
$x'(u_k)$	$y'(u_k)$		
+	+	↗	$\{L_1, L_2\}$
+	–	↘	$\{L_1, L_4\}$
–	+	↗	$\{L_2, L_3\}$
–	–	↖	$\{L_3, L_4\}$

Step 2: Finding L_{k+1} based upon geometric factor

Referring to Figure 2, let d_x and d_y are defined as the distances between the current point on curve r_x with the corresponding enclosing lines. For the shown case in Figure 2, geometrically, relation $|x'(u_k)|dy - |y'(u_k)|d_x = 0$ can be used to determine the intersected enclosing lines. We can conclude that if $|x'(u_k)|dy - |y'(u_k)|d_x$ is greater than zero (positive) than the curve intersects with L_1. Otherwise, the curve intersects with L_2. Similarly, by taking into considerations the values of d_x and d_y, the set of possible L_{k+1} due to geometric factor is obtained as summarizes in Table 2.

Table 2 Set of possible L_{k+1} due to geometric factor

Factor $\left(\|x'(u_k)\|d_y - \|y'(u_k)\|d_x\right)$	**Set of Possible L_{k+1} due to geometric factor**
+	$\{L_1, L_3\}$
–	$\{L_2, L_4\}$

Step 3: Finding L_{k+1} based upon tangent direction vector and geometric factor

Combining the set of possible L_{k+1} based upon the tangent direction vector and the geometric factor will produce the intersected line L_{k+1} as shown in Table 3.

Table 3 The intersected line L_{k+1}

Tangent Direction Vector Factor		**Geometric Factor**	**Resulting L_{k+1}**
$x'(u_k)$	$y'(u_k)$	$x'(u_k)d_y - y'(u_k)d_x$	
+	+	+	L_1
+	+	–	L_2
+	–	+	L_1
+	–	–	L_4
–	+	+	L_3
–	+	–	L_2
–	–	+	L_3
–	–	–	L_4

Step 4: Finding the value of u_{k+1}, x_{k+1}, y_{k+1}, DX_{k+1} and DY_{k+1}

Once the intersected line is determined, the value of u_{k+1} can be can be obtained by using tangent line approximation method [12]. Once u_{k+1} is known, x_{k+1}, y_{k+1}, DX_{k+1} and DY_{k+1} can be calculated as per Table 4.

Table 4 Finding the values of u_{k+1}, x_{k+1}, y_{k+1}, DX_{k+1} and DY_{k+1}

L_{k+1}	u_{k+1}	x_{k+1}	y_{k+1}	DX_{k+1}	DY_{k+1}
L_1	$u_{k+1} = u_k + \frac{d_x}{x_k'}$	$x(u_{k+1})$	$y(u_{k+1})$	$x_{k+1} - X_k$	$y_{k+1} - Y_k$
L_2	$u_{k+1} = u_k + \frac{d_y}{y_k'}$	$x(u_{k+1})$	$y(u_{k+1})$	$x_{k+1} - X_k$	$y_{k+1} - Y_k$
L_3	$u_{k+1} = u_k + \frac{d_x}{x_k'}$	$x(u_{k+1})$	$y(u_{k+1})$	$x_{k+1} - X_k$	$y_{k+1} - Y_k$
L_4	$u_{k+1} = u_k + \frac{d_y}{y_k'}$	$x(u_{k+1})$	$y(u_{k+1})$	$x_{k+1} - X_k$	$y_{k+1} - Y_k$

Step 5: Finding the next interpolating point X_{k+1}, Y_{k+1}

From the pre-defined closest distance region, Table 5 indicates that the following conditions can be derived.

Table 5 The next interpolating point

Range	Range > 0.5S	– 0.5S d" Range d" 0.5S	Range < 0.5S
DX_{k+1}	$X_{k+1} = X_k + S$	$X_{k+1} = X_k$	$X_{k+1} = X_k - S$
DY_{k+1}	$Y_{k+1} = Y_k + S$	$Y_{k+1} = Y_k + S$	$Y_{k+1} = Y_k - S$

3.0 COMPUTER IMPLEMENTATION AND DISCUSSION

Figure 3 illustrates the computer implementation procedure for the proposed interpolation algorithm. Notice that the required calculation terms are small. To evaluate the efficiency of the proposed interpolator, comparison with the existing general parametric curve interpolator by Kiritsis [3] is made. The existing algorithm employs Newton-Raphson iterative method to locate orthogonal distance point on curve as in Equation 2. Then the distances are then compared using Equation 3.

```
BEGIN
    INITIALIZE PARAMETERS
    WHILE  u_start ≤ u_final DO
        EVALUATE [x'(u_k), y'(u_k)]
        EVALUATE [d_x, d_y]
        EVALUATE | x'(u_k) | d_y − | y'(u_k) | d_x
        DETERMINE TANGENT DIRECTION FACTOR – TABLE 1
        DETERMINE GEOMETRIC FACTOR – TABLE 2
        EVALUATE  [u_k+1] – TABLE 4
        DETERMINE NEXT INTERPOLATED POINT p_k+1 – TABLE 5
        SET NEXT STEP = CURRENT STEP
    NEXT STEP
END
```

Figure 3 Computer Implementation Procedure for the Proposed Interpolator

Using this algorithm, in every iteration, Equation 2 and Equation 3 have to be evaluated three times. Table 6 presents the number of evaluation terms and the mathematical operations required between the proposed interpolator and the existing interpolator. For the proposed interpolator, equation for geometric factor in Table 2 $\left(\left|x'(u_k)\right|dy-\left|y'(u_k)\right|d_x\right)$ and equation to find d_x and d_y in Figure 2 (for example $d_x=\left(X_k+S\right)-s_k$) are considered while for the existing interpolator Equation 2 and Equation 3 are considered (which are evaluated three times for each iteration). Notice that, the terms for evaluating the current and the updating of the next values are same and are not considered. Clearly, the evaluation terms and the mathematical operations required for the proposed interpolator is smaller compared with the existing method.

Table 6 Comparisons of evaluation terms and mathematical operations required

Interpolator	Evaluated Terms	Operations required per iteration		
		+ and –	x and /	Square and $\sqrt{}$
Ref [3]	$x, y, x', y', x'', y'', x'^2, y'^2$	36	15	12
Proposed	x', y', d_x, d_y	5	2	0

4.0 EVALUATION EXAMPLE

To evaluate the proposed interpolator, real-time computer simulation using LabVIEW V6.1 was implemented on a Compaq Presario with Intel Celeron 500 MHz processor

under Windows XP environment. Two examples are chosen: a circle and a 2D cubic polynomial curve.

Circle:	$\boldsymbol{r}(u) = \cos(u)\boldsymbol{i} + \sin(u)\boldsymbol{j}$	$0 \le u \le 2\pi$
Polynomial:	$\boldsymbol{r}(u) = (1.22u - u^2)\boldsymbol{i} + (5.25 - u^2)\boldsymbol{j}$	$0 \le u \le 4.0$

Figure 4 and Figure 5 present the plots of the examples (with exaggerated BLU of 1.0 mm) while Table 7 and Table 8 show the simulation results compared to existing interpolator. The results indicate that the average interpolation time has decreased significantly (around 60%) while still maintaining the accuracy.

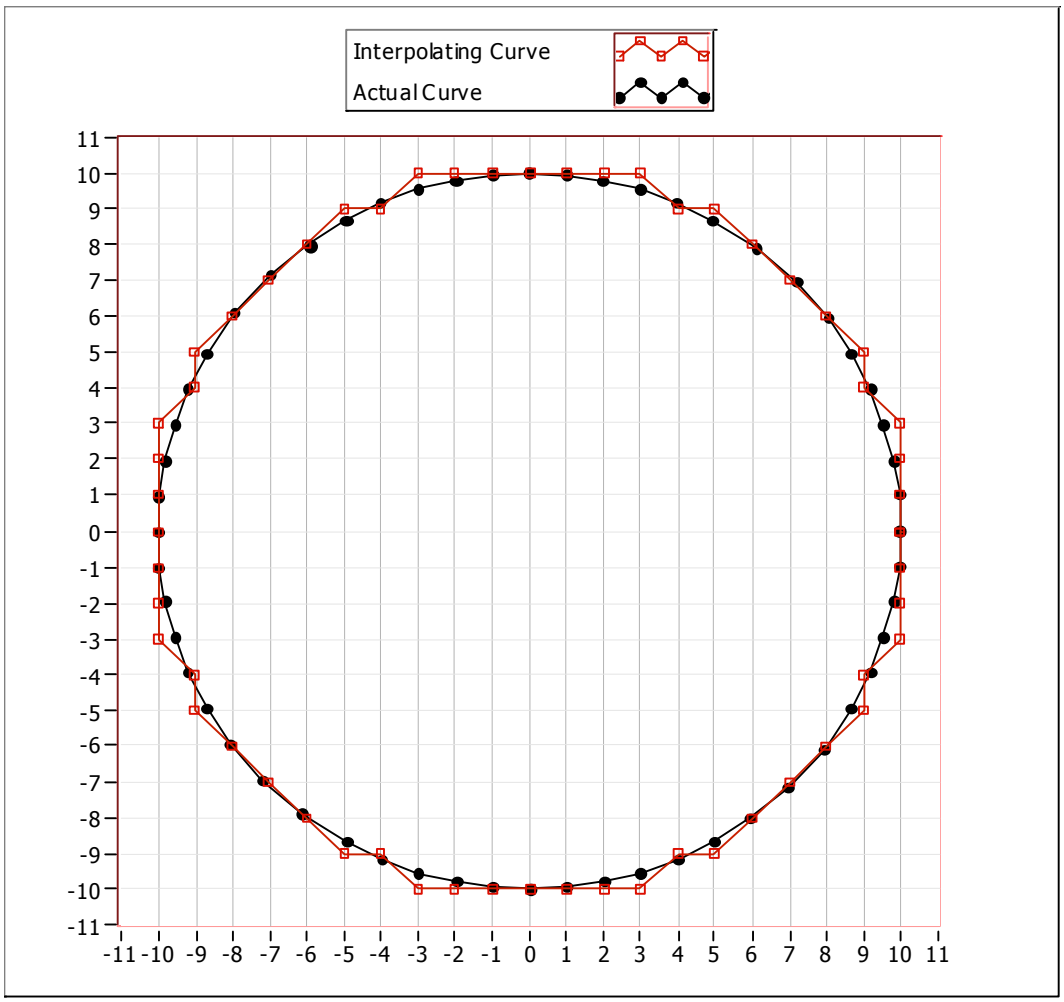


Figure 4 Plot of circle using the proposed interpolator. Circle example is $\boldsymbol{r}(u) = \cos(u)\boldsymbol{i} + \sin(u)\boldsymbol{j}$ for $0 \le u \le 2\pi$ and the BLU used is 1.0 mm.

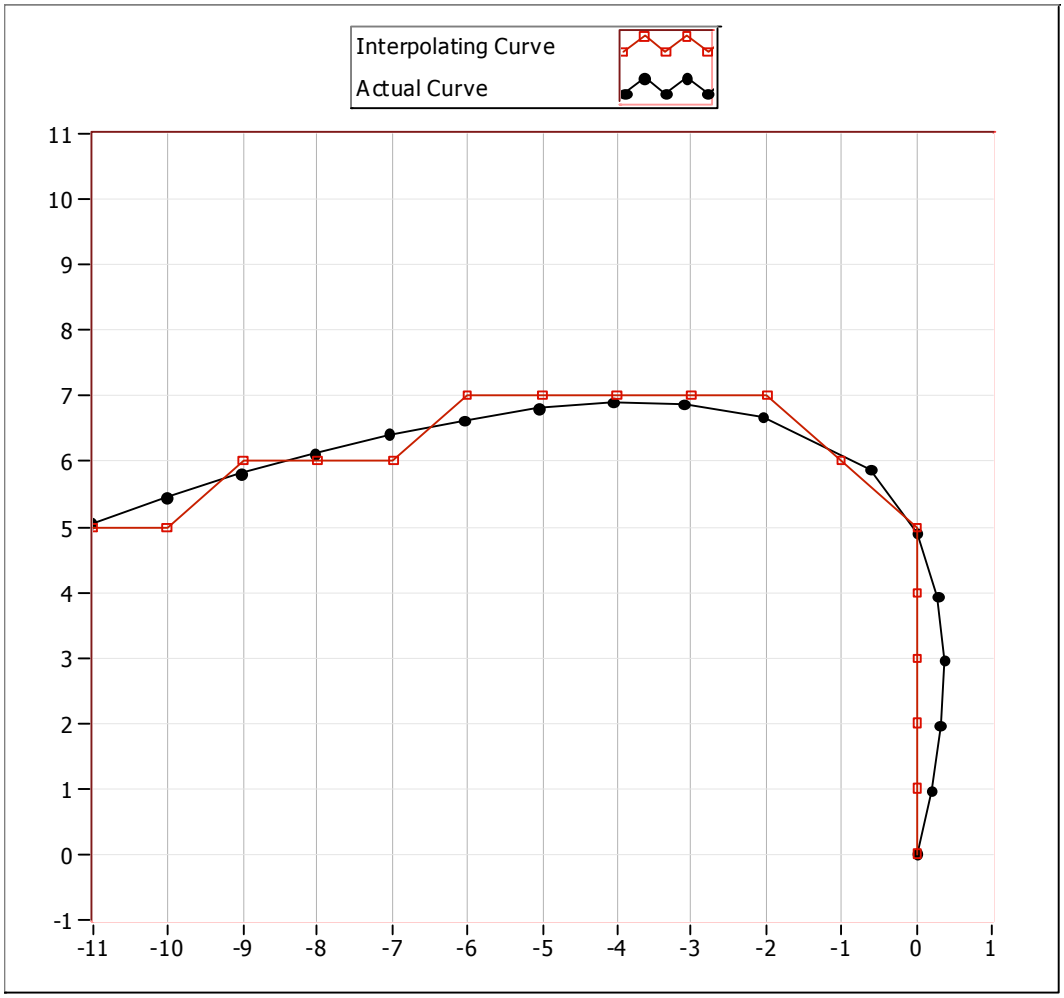


Figure 5 Plot of polynomial curve using the proposed interpolator. The curve is $\boldsymbol{r}(u) = (1.22u - u^2)\boldsymbol{i} + (5.25 - u^2)\boldsymbol{j}$ with $0 \leq u \leq 4.0$ and BLU of 1.0 mm.

Table 7 Comparisons of performance between the proposed interpolator with existing interpolator for the circle example

Interpolators	Maximum Error (E_{max} mm) BLU = 0.001mm	Average computational speed per iteration (Microseconds)
Interpolator [3]	0.00049996 mm	12.14
Proposed interpolator	0.00049999 mm	4.13
Performance increase	< ½ BLU	65.98 %

Table 8 Comparisons of performance between the proposed interpolator with existing interpolator for the polynomial curve example

Interpolators	Maximum Error (E_{max} mm) BLU = 0.001mm	Average computational speed per iteration (Microseconds)
Interpolator [3]	0.000499898 mm	13.39
Proposed interpolator	0.000499990 mm	5.20
Performance increase	< ½ BLU	61.17%

5.0 CONCLUSION

In short, we have proposed an efficient algorithm that is highly accurate and easy to control. The maximum allowable feed-rate has been increased significantly compared to the existing interpolator while maintaining the accuracy. With the trend of increased microprocessor clock speed, the future maximum feed-rate using reference-pulse technique will be higher. Further work will be performed on reference-pulse interpolator for general 3D parametric curve.

REFERENCES

[1] Yang, D. C. H. and T. Kong. 1994. Parametric interpolator versus linear interpolator for precision CNC machining. *Computer-Aided Design.* 26(3): 225-234.

[2] Min, Y. Y. and P. H. Won. 2001. A PC-NC milling machine with new simultaneous 3-axis control algorithm. *Int. J. of Mach. Tools & Manufacture.* 41: 555-566.

[3] Kiritsis, D. K. 1994. High precision interpolation algorithm for 3D parametric curve generation. *Computer-Aided Design.* 26(11): 850-856.

[4] Koren, Y. and O. Masory. 1981. Reference-pulse circular interpolators for CNC systems. *Trans. ASME J. Eng. Indust.* 103: 131-136.

[5] Koren, Y. 1979. Design of Computer Control for Manufacturing Systems. *Trans. ASME J. Eng. Indust.* 101: 326-332.

[6] Koren, Y. 1976. Interpolator for a computer numerical control system. *IEEE Transactions on Computers*. C-25: 32-37.

[7] Farouki, R. T. and T. Y. Feng. 2001. Exact Taylor series coefficients for variable-feedrate CNC curve interpolators. *Computer-Aided Design.* 33: 155-165.

[8] Yeh, S. S. and P. L. Hsu. 1999. The speed-controlled interpolator for machining parametric curves *Computer-Aided Design.* 31: 349-357.

[9] Shpitalni, M., Y. Koren and C. C. Lo. 1994. Realtime curve interpolators. *Computer-Aided Design.* 26(11): 25-234.

[10] Farouki, R. T. and S, Shah. 1996. Real time CNC interpolators for Phythagorean-Hodograph curves. *Computer Aided Geometric Design.* 13: 583-600.

[11] Koren, Y. 1997. Control of Machine Tools. *Journal of Manuf Science and Engineering.* 119: 749-755.

[12] Ellis R. and D. Gulick. 1986. *Calculus With Analytic Geometry*. New York: HBJ Publishers Third Edition.