大韓機械學會論文集 第6卷 第4號, pp. 341~345, 1982.

<論 文>

2次元 輪廓制御를 위한 直線 및 圓弧補間

李 奉 珍*·盧 台 錫*

(1982年 7月 7日 接受)

Linear and Circular Interpolation for 2-dimensional Contouring Control

Bong Jin Lee and Tae Seok Nho

Abstract

The interpolator is usually built in hardware (logic circuitry), and the interpolator fabricated in a single LSI chip is recently made use of in most NC controllers, making the system more compact. However, the LSI interpolator not only has the technical difficulties but also requires high cost, in its fabrication.

To solve these problems, we tried to find the method of interpolating by software, and succeeded in developing a program which, executed by INTEL's 8085 microprocessor, can distribute the input pulses of up to 4.0 [Kpps] for the linear interpolation and 3.0 [Kpps] for the circular interpolation.

This paper presents the algorithm used to reduce the execution time and the flow chart of the interpolation program, and also shows the possibility of software interpolation. The interpolation program designed in assembly language is presented in the appendix.

1. 序 論

NC 工作機械나 産業用 로봇 等의 移送系 制御는 크게 位置決定制御(positioning control; point-to-point control)와 輪廓制御(contouring control; continuous path control)로 分類할 수 있다. 位置決定制御는 한 點(位置)에서 다른 點으로 移動함에 있어서 그 사이의 經路는 制御하지 않고 빠른 速力으로 終點의 位置만 正確하게 찾아가게 하는 形態의 制御이며[1] 이에 比해 輪廓制御는 주어진 經路를 따라 주어진 速力으로 移動하게 하는 보다 까다로운 形態의 御制이다. 輪廓制御에 있어서 經路의 制御는 通常 補間法이 利用되는데 補間이라 함은 始點과 終點의 座標値만으로 주어진 經路(直線, 圓弧 等)上에 있는 點들의 座標를 次例로 求하는 것을 말하며 經路의 種類에 따라 直線補間(linear interpolation), 圓弧補間(circular interpolation) 等으로 부른다.

補間方式으로는 MIT 方式, DDA 方式, 代數演算方式 等이 開發되어 있으며 이들의 原理와 回路의 블록 圖도 紹介되어 있다.[2)~8)] 그러나 이들은 모두 하드웨어(hardware) 方式으로서 그 回路를 TTL (transistor-transistor logic) 等의 凡用 IC (integrated circuit)로 꾸밀 境遇 200餘個 程度의 SSI(small-scale integ-

* 韓國科學技術院

rated circuit)가 所要되어 하드웨어가 너무 膨大해진다. 先進國에서는 이들 하드웨어 方式의 補間器(interpolator)를 LSI(large-scale integrated circuit)化하여 시스템을 小形化하고 있으나 이를 爲해서는 高度의 IC 製造(fabrication)技術이 必要함은 勿論이거니와 엄청난 開發費가 所要된다.

이런 點을 考慮하여 本硏究에서는 補間을 소프트웨어(software)로 處理하는 方法을 試圖하게 되었는데 MIT 方式과 DDA 方式은 소프트웨어로 處理하기가 困難하여 代數演算方式을 採擇했다. 이 方式은 日本의 東京大學에서 開發한 것으로 그 原理는 紹介되어 있으나[2)~4)] 모든 演算을 全部 하드웨어로 處理하고 있을뿐만 아니라 알고리듬(algorithm)도 演算速度가 極히 빠른 하드웨어를 前提로 한 것이어서 그것만으로는 마이크로프로세서(microprocessor)로 處理하기가 困難하다. 그래서 演算時間을 줄이기 爲한 알고리듬을 새로이 考案하여 그에 따라 어셈블리 語(assembly language)로 프로그램하여 實驗해 본 結果 소프트웨어에 依한 補間의 可能性을 確認했는 바 그 알고리듬과 그에 依한 補間프로그램의 흐름圖(flow chart)를 紹介하기로 한다. 그리고 어셈블리 語에 依한 補間프로그램은 附錄에 싣고자 한다.

2. 直線補間(Linear Interpolation)

Fig. 1(a)에서와 같이 始點 A에서 終點 B까지 直線을 따라 移動하는 境遇에 對해 생각해 보자. 數式을 簡單히 하기 爲해 Fig. 1(b)와 같이 始點 A가 原點 O와 一致하도록 平行移動시켜서 그 때의 終點 B의 座標를 (X_e, Y_e)라 하면, 直線 $\overline{AB}$의 方程式은 $y=(Y_e/X_e)\cdot x$가 된다. 그리고 動點 $P(x_i, y_i)$가 이 直線의 上部에 있기 爲한 條件은 $y_i>(Y_e/X_e)\cdot x_i$, 即 $X_e\cdot y_i - Y_e\cdot x_i>0$이다. 따라서

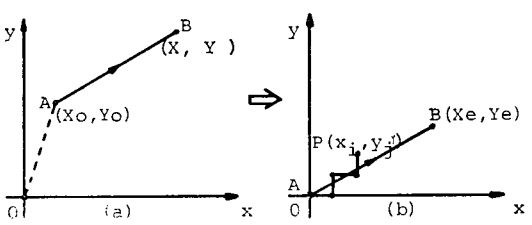

Fig. 1 Translation of coordinate for linear interpolation.

動點 $P(x_i, y_i)$가 直線 $\overline{AB}$의 上部에 位置하는가 下部에 位置하는가를 判別하기 爲한 判別式 $D_{i,j}$를

$$D_{i,j} \triangleq X_e\cdot y_j - Y_e\cdot x_i \tag{1}$$

라 두면,

i) $D_{i,j}>0$이면 動點 $P(x_i, y_i)$는 直線 $\overline{AB}$의 上部에 位置하게 되고,

ii) $D_{i,j}=0$이면 動點 $P(x_i, y_i)$는 直線 $\overline{AB}$ 上에 位置하게 되며,

iii) $D_{i,j}<0$이면 動點 $P(x_i, y_i)$는 直線 $\overline{AB}$의 下部에 位置하게 된다.

이것을 利用하여

i) $D_{i,j}\geqq 0$일 때는 x軸의 陽方向으로 1[pulse] 移動시키고,

ii) $D_{i,j}<0$일 때는 y軸의 陽方向으로 1[pulse] 移動시키면,

線分 $\overline{AB}$를 1[pulse] 以內의 誤差로 追跡할 수 있으며 $X_e=6$, $Y_e=5$인 境遇에 對한 直線補間 結果는 Fig. 2와 같다.

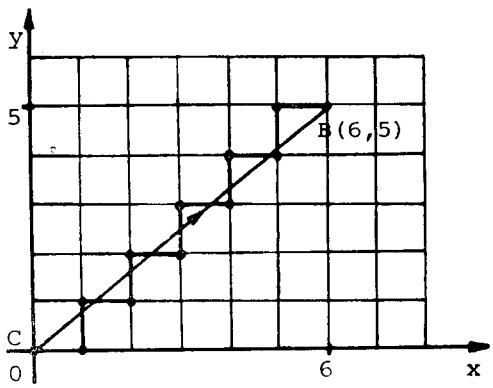

Fig. 2 The result of linear interpolation.

그런데 이와 같은 補間過程에서 1펄스(pulse) 移動할 때마다 그 點에서의 새로운 判別式 $D_{i,j}$를 計算하여 그 符號를 判定하지 않으면 안된다. 式(1)에 依하면 判別式 $D_{i,j}$의 값은 X_e와 y_j의 곱에서 Y_e와 x_i의 곱을 빼어서 求할 수 있지만 이 計算은 마이크로프로세서로 處理하기에는 너무 時間이 걸리는 일이며 그 理由는 다음과 같다. 大槪의 工作機械의 스트로우크(stroke)는 10[m] 未滿이지만 最小指令單位가 1[μ]인 境遇에 對해 생각해 보면 座標値를 16 빗 (bit) 데이터로 處理하면 最大制御 스토로우크가 65.535[mm]밖에 되지 않아서 座標値를 24 빗(bit) 데이터로 處理하여 最大制御 스트로우크를 16777.215[mm]로 잡을 수 밖에 없다. 即 機械의 스트로우크를 充分히 커버하기 爲해서는

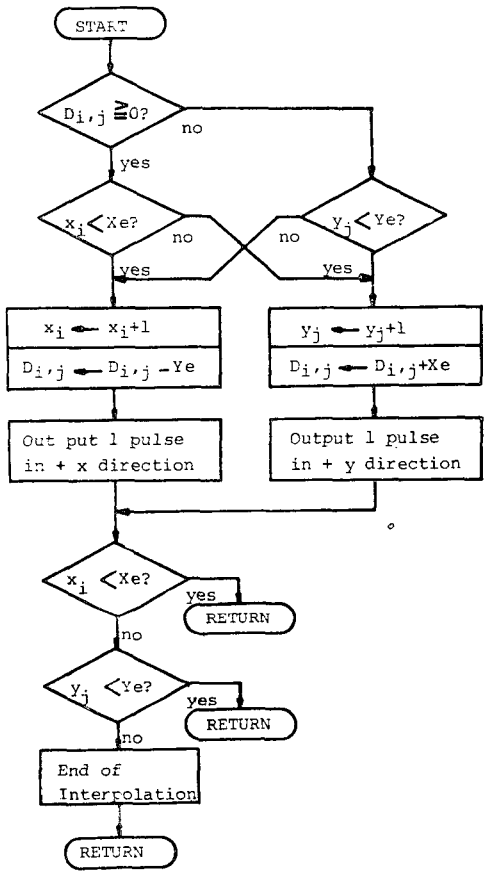

Fig. 3 The flow chart of linear interpolation subprogram.

座標値는 3바이트(byte) 데이터로 되어야 하므로 X_e, Y_e, x_i, y_j는 全部 3 바이트 데이터로 된다. 따라서 $D_{i,j}$의 計算에는 2번의 24 빗 곱셈과 1번의 24 빗 뺄셈이 必要하므로 이것을 簡單히 計算하는 方法을 模索하지 않으면 안된다.

위의 補間過程에서 變數 x_i는 x軸의 陽方向으로 1〔pulse〕移動했을 때 1增加하고 變數 y_j는 y軸의 陽方向으로 1〔pulse〕移動했을 때 1增加하며 始點에서의 x_i, y_j, $D_{i,j}$ 即 初期値 x_0, y_0, $D_{0,0}$는 全部 0임이 分明명하므로 이 點에 着眼하면 다음과 같은 結論을 얻을 수 있다.

i) x_i, y_j, $D_{i,j}$의 初期値는 全部 0이다.

ii) $D_{i,j}\geq 0$일 때는 x軸의 陽方向으로 1〔pulse〕移動하므로 移動後의 새로운 x座標 x_{i+1}과 判別式 $D_{i+1,j}$는 다음과 같다.

$$x_{i+1}=x_i+1 \tag{2}$$

$$D_{i+1,j}=D_{i,j}-Y_e \tag{3}$$

iii) $D_{i,j}<0$일 때는 y軸의 陽方向으로 1〔pulse〕移動하므로 移動後의 새로운 y座標 y_{j+1}과 判別式 $D_{i,j+1}$은 다음과 같다.

$$y_{j+1}=y_j+1 \tag{4}$$

$$D_{i,j+1}=D_{i,j}+X_e \tag{5}$$

以上과 같이 하여 1〔pulse〕移動後의 새로운 判別式을 簡單히 計算할 수 있으며 이에 依한 直線補間 서브프로그램(subprogram)의 흐름圖는 Fig. 3과 같다.

여기서 Fig. 3이 意味하는 프로세스(process)의 흐름에 對해 說明하고자 한다. 먼저 補間指令 펄스가 마이크로프로세서의 인터럽트(interrupt) 端子에 加해지면 現在位置에서의 判別式의 符號를 判定하여 나아갈 方向(x軸 또는 y軸)을 決定한 다음 判別式의 符號에 依해 決定된 方向으로 더 나아갈 餘分이 있는지 없는지를 判定하여 餘分이 있을 境遇에는 그 方向으로 나아가고 餘分이 없을 境遇에는 다른 軸方向으로 나아가게 된다. 그런데 該當 方向으로 出力 펄스를 내기 前에 移動後의 새로운 座標値와 判別式의 값을 計算해 둔다. 다음 該當方向으로 出力 펄스를 내보내고서 終點에 到達했는지를 判定하여 到達하지 않았을 境遇에는 그냥 리턴(return)하고 到達하였을 때는 補間終了 프로세스를 執行하고서 리턴한다.

3. 圓弧補間(Circular Interpolation)

圓弧補間은 時計方向(CW)과 反時計方向(CCW)의 2가지가 있으나 原理가 같기 때문에 여기서는 反時計方向의 圓弧補間에 對해서만 記術하고자 한다. Fig. 3-(a)에서와 같이 C點을 中心으로 하는 圓弧를 따라 A點에서 B點까지 移動하는 境遇에 對해 생각해 보자. 直線補間時와 마찬가지로 數式을 簡單히 하기 爲하여 Fig. 3-(b)와 같이 中心 C가 原點 O와 一致하도록 平行移動시켜 그 때의 A點과 B點의 座標를 各各(X_s,

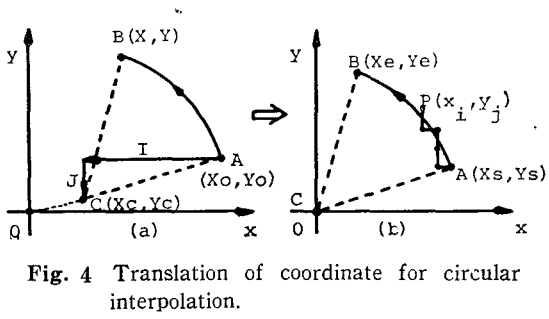

Fig. 4 Translation of coordinate for circular interpolation.

Y_s), (X_e, Y_e)라 하면 圓弧 $\widehat{AB}$의 方程式은 $x^2+y^2=X_e^2+Y_e^2$이 된다. 動點 $P(x_i, y_j)$가 이 圓弧의 外部에 있기 爲한 條件은 $x_i^2+y_j^2>X_e^2+Y_e^2$, 即 $x_i^2+y_j^2-(X_e^2+Y_e^2)>0$이다. 따라서 動點 $P(x_i, y_j)$가 圓弧 $\widehat{AB}$의 外部에 位置하는가 內部에 位置하는가를 判別하기 爲한 判別式 $D_{i,j}$를

$$D_{i,j}\triangleq x_i^2+y_j^2-(X_e^2+Y_e^2) \tag{6}$$

라 두면,

i) $D_{i,j}>0$이면 動點 $P(x_i, y_j)$는 圓弧 $\widehat{AB}$의 外部에 位置하게 되고,

ii) $D_{i,j}=0$이면 動點 $P(x_i, y_j)$는 圓弧 $\widehat{AB}$上에 位置하게 되며,

iii) $D_{i,j}<0$이면 動點 $P(x_i, y_j)$는 圓弧 $\widehat{AB}$의 內部에 位置하게 된다.

이것을 利用하여

i) $D_{i,j}\geq 0$일 때는 x軸의 陰方向으로 1[pulse] 移動시키고,

ii) $D_{i,j}<0$일 때는 y軸의 陽方向으로 1[pulse] 動시키면,

圓弧 $\widehat{AB}$를 1[pulse] 以內의 誤差로 追跡할 수 있으며 $X_s=6$, $Y_s=0$, $X_e=0$, $Y_e=6$인 境遇에 對한 圓弧補間結果는 Fig. 5와 같다.

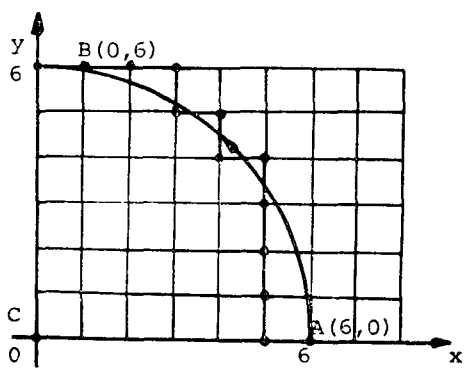

Fig. 5 The result of circular interpolation.

그런데 여기서도 直線補間時와 마찬가지로 새로운 判別式 $D_{i,j}$를 計算하여 그 符號를 判定해야 하므로 式(6)을 簡略化하는 方法을 찾아야 한다. 變數 x_i는 x軸의 陰方向으로 1[pulse] 移動했을 때 1減少하고 變數 y_j는 y軸의 陽方向으로 1[pulse] 移動했을 때 1增加하며 始點 A에서의 $x_i, y_j, D_{i,j}$ 即 初期値 $x_0, y_0, D_{0,0}$는 $x_0=X_s$, $y_0=Y_s$, $D_{0,0}=0$이므로 다음과 같은 結論을 얻을 수 있다.

i) $x_i, y_j, D_{i,j}$의 初期値 $x_0, y_0, D_{0,0}$는 다음과 같다.

$x_0=X_s$, $y_0=Y_s$, $D_{0,0}=0$

ii) $D_{i,j}\geq 0$일 때는 x軸의 陰方向으로 1[pulse] 移動하므로 移動後의 새로운 x座標 x_{i+1}과 判別式 $D_{i+1,j}$는 다음과 같다.

$$x_{i+1}=x_i-1 \tag{7}$$

$$D_{i+1,j}=D_{i,j}-(2x_{i+1}+1) \tag{8}$$

iii) $D_{i,j}<0$일 때는 y軸의 陽方向으로 1pulse] 移動하므로 移動後의 새로운 y座標 y_{j+1}과 判別式 $D_{i,j+1}$은 다음과 같다.

$$y_{j+1}=y_j+1 \tag{9}$$

$$D_{i,j+1}=D_{i,j}+(2y_j+1) \tag{10}$$

以上과 같이 하여 1[pulse] 移動後의 새로운 判別式의 값을 簡單히 計算할 수 있으며 이에 依한 圓弧補間 서브프로그램의 흐름 圖는 Fig. 6과 같다. 그리고 Fig. 6

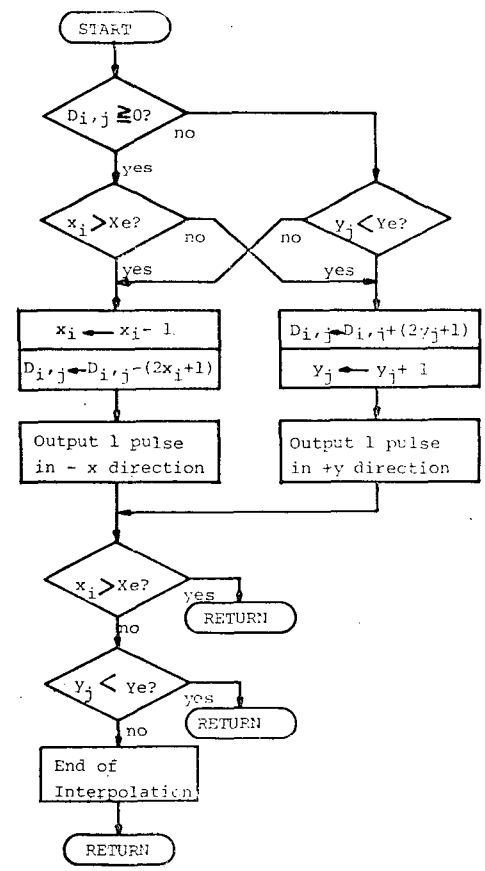

Fig. 6 The flow chart of CCW circular interpolation subprogram.

의 프로세스의 흐름은 直線補間時와 同一하므로 說明은 略하기로 한다.

4. 結 論

補間을 소프트웨어로 處理하는 方法을 試圖하여 위에서 다룬 바와 같은 알고리듬과 흐름 圖에 따라 어셈블리 語로 프로그램하여 附錄에 실은 補間 프로그램을 얻었으며 이 프로그램의 命令(instruction)數, 사이클數, 執行時間은 Table 1과 같다. 그리고 이 프로그램

Table 1 The execution time of interpolation program.

	Number of instructions	Number of clock cycles*	Execution time	Remarks
Linear interpolation	99	737	246〔μsec〕	
Circular interpolation	126	917	306〔μsec〕	

(* The frequency of the clock is 3 〔MHz〕.)

을 8085 마이크로프로세서로 執行시켜 본 結果 直線補間은 4.0〔Kpps〕까지, 圓弧補間은 3.0〔Kpps〕까지의 補間指令 入力 펄스를 處理할 수 있음이 確認되었다. 이 處理速度는 最小設定單位가 10〔μ〕인 시스템에서 〔mm/sec〕로 換算해 보면 各各 40〔mm/sec〕, 30〔mm/sec〕가 되어 大部分의 切削加工은 커버할 수 있는 速度이다. 그러나 피치(pitch)가 긴 나사加工이 不可能하고 더우기 最小設定單位가 1〔μ〕인 시스템에서는 各各 4.0〔mm/sec〕, 3.0〔mm/sec〕 밖에 되지 않아서 困難하다. 이로 미루어 보아 어떠한 시스템에도 適用할 수 있는 滿足스러운 結果를 얻으려면 計算時間을 數十倍程度로 短縮시킬 必要가 있는데 이것은 마이크로프로세서의 性能이 數十倍 向上되지 않으면 안되는 일이므로 凡用 마이크로프로세서 보다는 빗슬라스 마이크로프로세서 設計技法(bit-slice microprocessor design technique)을 活用하여 24 빗(bit) 專用 마이크로프로세서를 만들어서 마이크로프로그램(microprogram)으로 處理하는便이 낫겠고 그렇게 하면 250〔Kpps〕까지는 充分히 處理할 수 있으리라 본다.

參 考 文 獻

1) Bong Jin Lee & Tae Seok Nho; "The Positioning Control of Robot using Microcomputer", The 11th International Symposium on Industrial Robots Proceedings, pp. 237~244, Oct. 1981.

2) 李奉珍 ; "數値制御", 賢文出版社, 1978.

3) 稻葉 清右衛門 ; "やさしい NC讀本", 日本能率協會, 1977.

4) 山岸正謙 : "NC 工作機械", 日刊工業新聞社(日本), 1975.

5) Roger S. Pressman & John E. Williams; "Numerical Control & Computer Aided Manufacturing", John Wiley & Sons, Inc., 1977.

6) Yoram Koren, Alexander Shani & J. Ben-Uri; "Numerical Control of a Lathe", IGA-6, No. 2, pp. 175~179, Apr. 1970.

7) Per E. Danielsson; "Incremental Curve Generation", C-19, No. 9, pp. 783~793, Sept. 1970.

8) Yoram Koren; "Interpolator for a Computer Numerical Control System", C-25, No. 1, pp. 32~37, Jan. 1976.