44.

Interpolator for a Computer Numerical Control System

YORAM KOREN

Abstract—A software interpolator which is comprised of linear and circular interpolations is compared with its hardware counterpart and with other circular interpolation methods. The software interpolator and the feed-rate control are contained in the numerical control (NC) program of a computer numerical control (CNC) system and enable a contouring control of the machine tool in any required feed-rate.

Index Terms—Adaptive control (AC), analog-to-digital processor (ADP), computer numerical control (CNC), digital differential analyzer (DDA), numerical control (NC), point-to-point (PTP), time base generator (TBG).

INTRODUCTION

THE computer numerical control (CNC) concept employs a digital computer, usually a minicomputer, for on-line control of the numerical control (NC) machine tool and eliminating, as far as possible, additional hardware circuits in the controller cabinet. The change from the use of a controller unit to CNC may be regarded as the most important advance in the philosophy of the design of NC systems that occurred during the first years of the seventies. The development of CNC systems has advanced as a result of the rapidly improving capabilities and falling prices of small computers, which make it suddenly attractive to use standard computers as part of the NC systems. Therefore, it is becoming increasingly evident that the cost of a CNC unit will be actually lower than its equivalent conventional NC counterparts.

Many functions of the conventional NC controller are replaced in a CNC system by a computer program denoted as the NC program. Naturally, the data processing, feed-rate calculations, and the interpolating between two data points are performed by software, while the controller contains only the position and velocity control loops.

A feed-rate control and an interpolator based on a simulation of digital differential analyzer (DDA) integrators are discussed in this paper. Although the principles of a hardware DDA integrator are well known [1]–[3], they are summarized at the outset since we found it the simplest way to represent the notations used in this paper. The interpolator is capable of linear and circular interpolation in accordance with instructions from the data tape. The hardware circuit is compared with its software counterpart. The latter is a part of a CNC system for a 3-axis milling machine which was developed at McMaster University, Hamilton, Ont., Canada.

The block diagram of the system is shown in Fig. 1. It includes five major components: a milling machine, a

Manuscript received June 15, 1974; revised December 15, 1974.
The author is with the Engineering Experiment Station, University of Wisconsin-Madison, Madison, WI 53706.

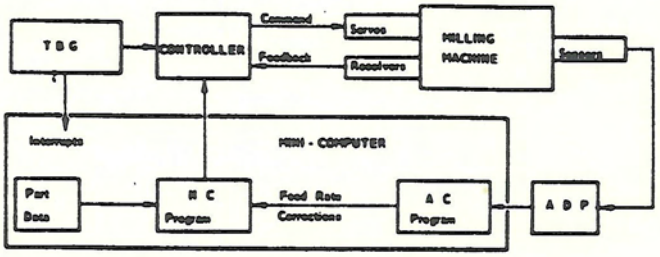

Fig. 1. Block diagram of the system.

minicomputer (HP 2100A), a data processing system [analog-to-digital processor (ADP)], a time base generator (TBG), and a controller which contains 3 control circuits for the 3 axes-of-motion of the milling machine.

The computer handles two programs: NC and adaptive-control (AC) programs.

The AC program accepts the sensor outputs and uses them to calculate a feed-rate correction which is supplied to the NC program. The interrupt system of the computer takes care of the simultaneous running of both programs.

The general structure of the NC program is shown in Fig. 2. The program contains five subroutines: point-to-point (PTP), feed, interpolator, output, and position. The NC program uses an additional routine denoted as the initiator routine. The main function of the initiator routine is the loading of a new data block to the memory locations which the NC program is using.

The feed routine generates interpolation commands in a rate dependent on the feed word (f) in the data block. The maximum rate of interpolation commands is equal to the frequency of the interrupt pulses. For every interpolation command which is produced by the feed routine, a single cycle of the DDA is simulated in the interpolator. If as the result of a DDA cycle an overflow pulse is generated either in one or two axes, the output routine sends a command pulse to the controller. For every command pulse, the position counter of the appropriate axis is decremented by one unit. The counters are contained in the position routine. A zero position check of both counters is performed. When both counters are zero (which means that the machining of the current segment has terminated), the program jumps to the initiator routine in order to load a new block and process its data.

DDA INTEGRATOR

A hardware DDA integrator consists of two registers of equal length, usually designated y and r, and a small additional register designated Δy. Usually, Δy is a 1-bit register and the maximum content of y and r is the number q. The DDA timing is controlled by an external clock, which has a frequency of w pulses per second and consequently produces pulses in intervals dt. Two actions take place for each clock pulse.

1) The number stored in the Δy register is added to or subtracted from the content of the y register:

$$y_{i+1} = y_i + \Delta y_i \quad \text{or} \quad y_{i+1} = y_i + (-\Delta y)_i. \tag{1}$$

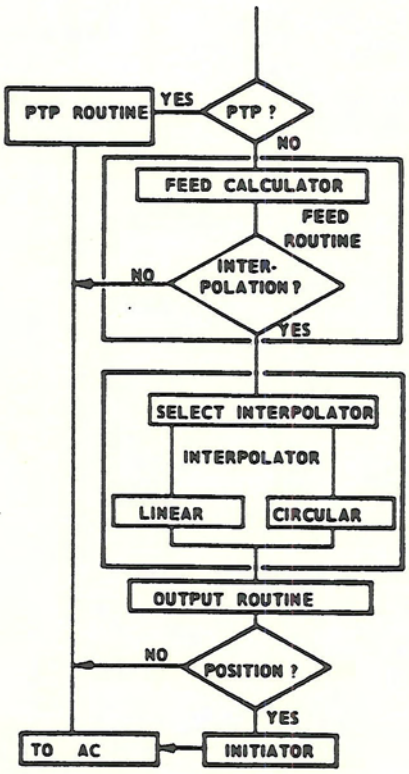

Fig. 2. Flowchart of the NC program.

2) The content of the y register is added to that of r and stored in the latter:

$$r_{i+1} = r_i + y_{i+1}. \tag{2}$$

Whenever $r > q$, an overflow pulse is applied by the integrator at the output Δz and the contents of r are reduced by q:

$$r_{i+1} = (r_i + y_{i+1}) - q. \tag{3}$$

The frequency w_z of the overflow pulses is given by the equation

$$w_z = \frac{dz}{dt} = \frac{y}{q} w. \tag{4}$$

Thus,

$$\Delta z \simeq dz = Sy\, dt, \tag{5}$$

where $S = w/q$ and called the scale factor. Equation (5) is very useful in solving differential equations by a DDA setup.

HARDWARE INTERPOLATOR AND FEED-RATE CONTROL

A hardware interpolator consists of a pair of DDA integrators (1 and 2 in Fig. 3). It is capable of linear and circular operation according to instructions from the punched tape. It controls simultaneously two axes which can be X and Y, or X and Z, or Y and Z. But for simplification in the further discussion, the two controlled axes will be denoted by X and Y. The feed-rate control comprises an additional DDA integrator (3 in Fig. 3) whose

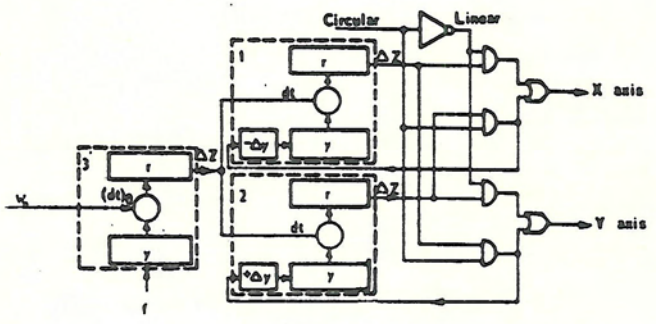

Fig. 3. Hardware interpolator and feed-rate control.

y register is fed by the number f, which is the feed-rate number, such that

$$w_{s_2} = S_0 f = f w_0 / q_0, \tag{6}$$

w_0 (supplied to $(dt)_0$ in Fig. 3) being a fixed external clock (which is replaced by a source of interrupt pulses in the software interpolator). w_{s_2} is fed as clock pulses to integrators 1 and 2 thereby enabling the motor speeds to be controlled, at the desired ratio, by f in both types of interpolation. That means that the scale factor S of integrators 1 and 2 is

$$S = w_{s_2}/q. \tag{7}$$

Substituting of (6) into (7) yields

$$S = f w_0 / q q_0. \tag{8}$$

Notice that it is not necessary that the maximum content of the interpolator registers (q) be equal to that of the registers of the feed-rate DDA (q_0).

The interpolator generates simultaneously two feed-rate commands V_x and V_y, which are supplied to two axes of the milling machine. The feed-rate command is measured in pulses per second. The respective position commands will be denoted as η_x and η_y, measured in pulses. Notice that (after linearization)

$$V_x = \dot{\eta}_x \qquad \text{and} \qquad V_y = \dot{\eta}_y.$$

In NC systems, the required distances are represented on the data tape in basic length units (BLU's). One BLU in the described system is 0.0001 in. The overflow pulses from the Δz outputs are fed directly to the control loops, serving as commands to the motors.

Each output pulse from the interpolator will cause a motion of one BLU in the appropriate direction. This enables a discussion about distances in terms of pulses. A detailed explanation of a hardware interpolator is given in [4]–[6]. However, a short explanation and the principal equations are presented below.

For linear milling with paths a and b along the X and Y axis, respectively, we have

$$\Delta z_1 = d\eta_x = V_x\,dt = (V/L)a\,dt \tag{9a}$$

$$\Delta z_2 = d\eta_y = V_y\,dt = (V/L)b\,dt, \tag{9b}$$

where V is the required feed-rate (velocity) along the cutting path and

$$L = (a^2 + b^2)^{1/2}. \tag{10}$$

Comparing (9a) and (9b) with (5), we see that the number a must be fed as the initial condition to the y register of integrator 1, and the number b to that of integrator 2. In this case (V/L) serves as the scale factor S, which yields from (8)

$$\frac{f w_0}{q q_0} = \frac{V}{L}. \tag{11}$$

The condition prescribed by (11) must be fulfilled in order to control the tool in the required feed-rate.

In circular interpolation, the y register of integrator 2 is fed intially with the number j ($j = R \sin \omega t$) and its output is connected to the X axis, while integrator 1 is fed initially with the number i ($i = R \cos \omega t$) and its output in turn is connected to the Y axis. The content $R \sin \omega t$ of the y register of integrator 2 is updated throughout the circular operation by an increment $\Delta y = d(R \sin \omega t)$ obtained from the output of integrator 1, which is supplied simultaneously also the Y axis. Similarly, the output of integrator 2, $-d(R \cos \omega t)$, is connected to $-\Delta y$ of integrator 2 for updating the $R \cos t$ value in its y register, and at the same time it is connected to the X axis as well.

For generating a circular arc, the following conditions must be fulfilled:

$$\Delta y_2 = d(R \sin \omega t) = R \cos \omega t\,dt = V_y\,dt = \Delta z_1 \tag{12a}$$

$$-\Delta y_1 = -d(R \cos \omega t) = R \sin \omega t\,dt = V_x\,dt = \Delta z_2. \tag{12b}$$

In this case, the angular velocity ω is the scale factor S, which means by using (8) that

$$\omega = f w_0 / q q_0. \tag{13}$$

The feed-rate number f must be programmed in such a way as to fulfill the condition given in (13).

FEED-RATE INSTRUCTION

The feed-rate number f in NC systems which use hardware interpolators is calculated by the "inverse time method." This method provides the system with a code which is the reciprocal of interpolation time in minutes. The f in linear motion is calculated by the formula

$$f = 10 V_0 / l, \tag{14}$$

where V_0 is the velocity along the path-of-motion measured in inches per minute and l is the incremental length of the tool-path measured in inches, the formula for f in a circular cutting motion is

$$f = 10 V_0 / r. \tag{15}$$

V_0 is measured in inches per minute, and r, the radius of the arc, is in inches.

By converting the distance and time units, (14) and

(15) can be rewritten as follows:

$$\text{For linear motion} \qquad f = 600V/L, \tag{16}$$

$$\text{For circular motion} \qquad f = 600V/R, \tag{17}$$

where V is in pulses per second and L or R in pulses. Substituting f from (16) into (11) yields

$$w_0 = qq_0/600. \tag{18}$$

Bearing in mind that $V = \omega R$, the same result is obtained by substituting f from (17) into (13), i.e., for the circular interpolator.

Notice that q_0 is measured in the same units as f, which are [1/min] in the discussion of this section, and that q is measured in pulses. The clock frequency obtained from (18) is measured in pulses per second.

According to the Electronic Industries Association (EIA) standards [7], f is given by a four-digit number, thus the minimum value of q_0 is 10 000. Let us assume that the maximum incremental motion (and the maximum radius) is limited to 12 in and that one BLU is 0.0001 in. For these data, the frequency of the external clock is 2 MHz. The clock frequency can reach much higher frequencies when using a serial DDA instead of the parallel type which was discussed so far.

SOFTWARE INTERPOLATOR

The interpolator in a CNC system is a software one. The software interpolator is a computer program which simulates a single cycle of the hardware interpolator and the feed-rate control. The external clock w_0 is replaced by a source of interrupt pulses which are supplied to the computer. This enables the computer to run simultaneously on both the NC and the AC program. Normally, the computer runs on the AC program, but whenever an interrupt occurs the computer starts to execute the NC program. When the latter is terminated, the computer control returns to the AC program and continues to perform it from the point of interruption.

Fig. 4 shows the flowcharts of the feed calculator, the linear interpolator, and the circular interpolator. Simpler DDA's with $\Delta y = 0$ are used in the simulation of the feed calculator and the linear interpolator, while full software DDA's are used for the circular interpolator. Notice that q and q_0 represent in a hardware DDA the maximum content of the registers of the interpolator and the feed calculator, respectively.

A simple calculation in the previous section had proved that the frequency of the external clock in the hardware interpolator is in the range of megahertz. However, since the cycle time of a minicomputer is about 1 μs (and an execution time of one instruction is 2 μs), it is impossible to use such a high frequency for the interrupt pulses. This leads to another definition for q, and therefore the software interpolator is slightly different from a direct simulation of a hardware one.

In order to obtain the required feed-rate V in linear motion, the condition which was prescribed by (11) must

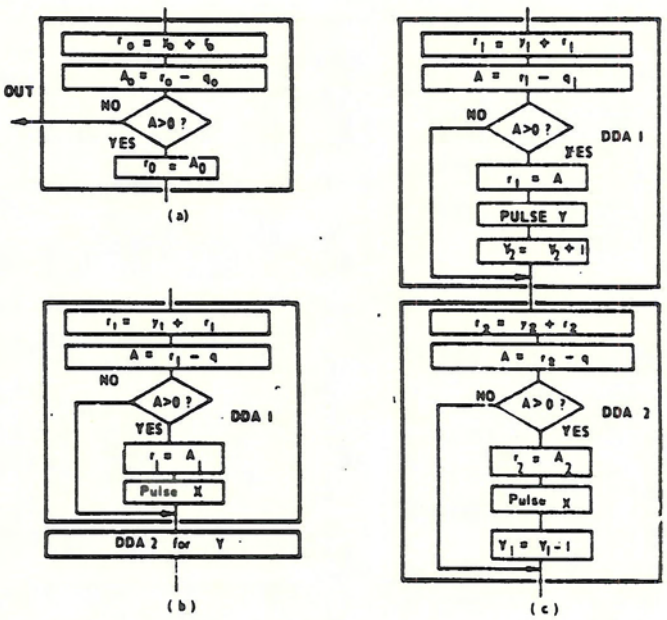

Fig. 4. Flowcharts of the (a) feed-rate calculator, (b) linear interpolator, (c) circular interpolator.

be fulfilled:

$$V = \frac{fw_0}{q_0}\left(\frac{L}{q}\right). \tag{19}$$

For a circular motion, L in (19) is replaced by the arc radius R, as can be seen from (13).

The efficiency of the hardware interpolator, which is given by the ratio (L/q), might be very low. L/q represents the ratio between the actual distance of motion (L) and the maximum allowable one (q). An incremental motion of a relatively small distance with a high feed-rate prescribes the requirement of the high frequency w_0. Hence, the frequency w_0 can be decreased by increasing the efficiency of the interpolator. In computer software, this can be easily done by using a variable q rather than a fixed one as in a hardware interpolator, thus increasing the efficiency to 1.

The value of q is calculated at the beginning of each segment.

$$\text{For linear motion} \qquad q = L.$$

$$\text{For circular motion} \qquad q = R.$$

L is given in (10) while $R = (i^2 + j^2)^{1/2}$. The initial values of i and j are known since they are used as the initial conditions for the y registers of the interpolator. By using the variable q, (11), (13), and (19) reduce to

$$V = w_0 f/q_0. \tag{20}$$

Since w_0 is a constant, the term f/q_0 represents the ratio between the required feed-rate and the maximum allowable one. But as this ratio is equal to V/w_0, the clock frequency w_0 is the maximum allowable velocity measured in pulses per second. By this method, much lower clock frequencies are obtained. In the described system, a maximum feed-rate of 30 in/min and a BLU of 0.0001 in were

chosen. For these data, the frequency of the interrupt pulses is 5000 pps. Such a frequency allows time intervals of 200 μs for the execution of the NC program. In fact, the maximum execution time of the NC program is 170 μs (2 μs/instruction) which always permits enough time for the execution of the AC program.

Another advantage of a software interpolator which uses $q = L$ or $q = R$ (depending on the interpolation type) is in calculating the feed-rate number f. Instead of using the inverse time method, the feed-rate number is programmed directly in inches per minute and multiplied by a constant depending on the required resolution. In the described system, a resolution of 0.01 in/min was chosen, and since the maximum feed-rate is 30 in/min the value of q_0 is 3000 and f is calculated according to the formula

$$f = 100V_0, \tag{21}$$

where V_0 is the required feed-rate in inches per minute.

COMPARISON WITH OTHER METHODS

The circular interpolator which was presented applies a method of solving a second-order differential equation by converting it to a set of two difference equations. It might be interesting to compare the proposed algorithm with other known methods from the literature. Two errors might appear in any solution: the truncation error which can be determined analytically, and the round-off error which has a nonlinear effect on the solution. When dealing with difference equations, it is convenient to signify the time interval (or step size) by T rather than dt. The radius of the circle is assumed to be one in the following discussion.

An algorithm which is not affected by truncation error is the state-transition method [8]. In this method, the difference equations to be solved are

$$\begin{aligned}\cos \omega(n+1)T &= \cos \omega T \cos \omega nT - \sin \omega T \sin \omega nT \\ \sin \omega(n+1)T &= \cos \omega T \sin \omega nT + \sin \omega T \cos \omega nT.\end{aligned} \tag{22}$$

Since the initial conditions are known, (22) generates the sequence of points which approximate a circle. Notice that the values of $\cos \omega T$ and $\sin \omega T$ must be precalculated for each ω by a Taylor's series expansion. The accuracy of the solution depends upon the accuracy to which the series expansion was computed.

Consider a case in which the following expansion is taken:

$$\begin{aligned}\cos \omega T &= 1 - (\omega T)^2/2 = A \\ \sin \omega T &= \omega T = B.\end{aligned} \tag{23}$$

Equation (22) becomes

$$\begin{aligned}\cos \omega(n+1)T &= A \cos \omega nT - B \sin \omega nT \\ \sin \omega(n+1)T &= A \sin \omega nT + B \cos \omega nT.\end{aligned} \tag{24}$$

Equation (24) was analyzed in [9] as an example of Heun's method, in [10] as an example of Adam's method, in [11] as an example to trapezoidal integration, and in [12]. The precalculation of A and B is relatively simple, and if T is sufficiently small this approximation provides accurate results. It can be shown that the radius truncation error resulting in this case decreases linearly with T^4 [9], [12].

A similar method is the Tustin method [8] which yields the same equations as (24) but with the following definitions for A and B:

$$A = \frac{1 - (\omega T/2)^2}{1 + (\omega T/2)^2} \qquad B = \frac{\omega T}{1 + (\omega T/2)^2}. \tag{25}$$

The truncation error is of the same order of magnitude as in the previous one, but since the constants A and B are more complicated, the round-off error can be slightly greater.

A more accurate result can be achieved by using the Runge–Kutta method. A solution using this method for our case can be found in [8] and [9]. The truncation error in this case decreases with T^6. The Runge–Kutta method requires an enormous number of calculations and therefore cannot be considered as an on-line control method.

The approach presented in [4]–[6] using hardware DDA's is equivalent to using Euler's method. Euler's method as applied to the given problem is discussed in [8]–[10]. The differential equations are given in (12); the corresponding difference equations are as in (24) with the following definitions for A and B:

$$A = 1 \qquad B = \omega T. \tag{26}$$

The Euler method yields a truncation error which decreases linearly with a decrease of T [9].

Now let us analyze the software interpolator proposal in this paper. Considering the flowchart in Fig. 4, one sees that for calculating the sine in the nth step the new value of the cosine is used, which means

$$\begin{aligned}\cos \omega(n+1)T &= \cos \omega nT - \omega T \sin \omega nT \\ \sin \omega(n+1)T &= \sin \omega nT + \omega T \cos \omega(n+1)T \\ &= (1 - \omega^2T^2) \sin \omega nT + \omega T \cos \omega nT.\end{aligned} \tag{27}$$

Thus, a notable improvement over the hardware method is achieved. The approximation of $\cos \omega T$ is not 1 as for the hardware interpolator but is 1 in the first equation and $(1 - \omega^2T^2)$ in the second equation, with an average of $1 - \omega^2T^2/2$ as in (23). Thus the radius truncation error decreased with T^4 rather than with T^2 as for the hardware DDA.

To summarize the discussion up to this point, four types of methods have been presented.

1) Euler method, which is a first-order approximation method and is used in hardware DDA's.

2) Second-order approximation methods, which include the Tustin method, the Adams or Heun method, and the proposed software interpolator method. All produce truncation errors of the same order of magnitude.