A New Linear Interpolation Algorithm

Min Zeng [a, *], Yanjie Zhang [b] and Jian Huang [c]

School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou 510640, China.

[a]memzeng@scut.edu.cn, [b]memzeng@163.com, [c]969225065@qq.com

Abstract. The commonly used linear interpolation methods in the NC system are point by point comparison method, digital integration method and minimum deviation method. All of these interpolation algorithm need operation in each feeding period, which consume CPU time severely, and makes the stepping motor vulnerable to violent vibration. When the ratio of rotational speed of Y-axis motor over that of X-axis motor is identical to the slope of theoretical line, resultant velocity of the speed of the X-axis and the Y-axis is parallel to the theoretical line according to velocity synthesis principle. This paper demonstrated that by using this method, CPU only needed to operate once and can make the velocity of X-axis and the Y-axis unchanged, to complete linear interpolation. In addition, the algorithm error was shown smaller than 1 pulse equivalent by theoretical analysis and MATLAB. By using MATLAB to compare this algorithm presented here with point by point comparison algorithm in terms of speed and interpolation calculation time, it was indicated that the algorithm was promising in industry application.

Keywords: Linear interpolation algorithm; One interpolation operation; Stable velocity; Resultant velocity; Operate once.

1. Introduction

Most of the interpolation of circular arc [1], NURBS curve [2, 3], Spline curve [4], andother curves [5] in numerical control machine system includes two step interpolation algorithm-the first step is known as the coarse interpolation using a straight line to approximate the theoretical curve, it was executed in advance, and the maximum feed step was generated. The second step is fine interpolation using linear interpolation to complete the line generated in the first step and adjust the feed step. So the interpolation precision and the interpolation speed of the straight line as the simplest element [6] will determine the final interpolation precision and interpolation speed.

Point by point comparison method, digital integral algorithm and minimum deviation algorithm are the three most often used algorithms. The minimum deviation method divides the rectangular coordinate system into 8 octants by the four straight lines made up by X-axis, Y-axis, $y=x$, $y=-x$. The theoretical line is set into one of the octants by coordinate transformation. And at each feeding cycle, a smaller deviation direction is selected after the deviation of each direction is calculated. The relative direction is selected after a comparison between theoretical and practical lines in the point by point comparison principle at each feeding cycle, and then related information is sent to the pulse generator. The ratio of increment of Y-axis and X-axis is equal to the slope of the theoretical line, and both of the increment of the two axis is less than 1 in Digital Differential Analyzer, when the amount of the integrator is bigger than 1, the integrator overflows, then the related pulse generator sends a pulse[7, 8].

Minimum deviation algorithm shows the minimum error in these three algorithms, which can be as small as one half of equivalent pulse. And the other two algorithms error can be as small as one equivalent pulse. However, in order to get the accuracy mentioned above, all these three algorithms send one pulse each time, which will greatly increase the number of interpolation calculation, thereby consume the CPU time. As the traditional algorithms need one interpolation operation, the X-axis stepping motor drivers and the Y-axis motor drivers cannot get pulses per feed cycle, so the stepping motor and y stepping motor start and stop frequently and irregularly, which make the stepping motor vibrate severely and make the motor easy to lose steps giving rise to greater error.

LiW etal.[9] proposed a linear interpolation method by transforming the rectangular coordinates into polar coordinates. However, the algorithm error was high and the operation was complex. Wei Yang etal.[10] put forward an improved method by point comparison interpolation algorithm based on 8 directions, namely *+x*, *+y*,*-x*, *-y*, *+x+y*, *+x-y*, *-x+y*, *-x-y*. By maximum interpolation error analysis, analytical.solution,.numerical comparison, and computer computation, an error calculation method was obtained. Such method demonstrated the benefits of high interpolation precision, stable computation, and high computation speed. Dumitru D and Strajescu E [11] set up a lathe servo system, gained the linear interpolation error by Matlab/Simulink, and analysed the source of errors.

This paper put forward the linear interpolation based on the hardware device, containing a CPU and two pulse generators, which can be timers , external hardware equipment or other devices.

By controlling the ratio of the velocity of *y*-axis over that of the *x*-axis equivalent to the slope of the theoretical line, the resultant velocity will be parallel to the theoretical line according to velocity synthesis principle. As start point is the same, so in this way, the synthesis track of X-axis and the Y-axis coincide with CPU to carry out a calculation, then the number of the X-axis and the Y-axis and the frequency the X-axis and the theoretical line. Therefore, the algorithm only needs the Y-axis will be sent to the relevant pulse generator.

2. Interpolation Priciple

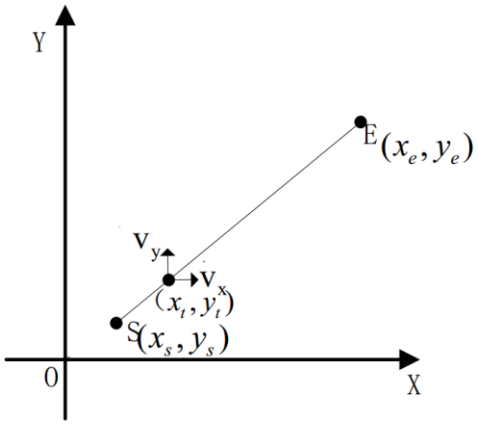


Fig. 1 Linear sketch map

As shown in Fig. 1, there is a line segment in a straight line (1).

$$y - y_s = k(x - x_s) \tag{1}$$

The coordinate of start point A was (x_s,y_s), and that of end point B was (x_e,y_e). Assuming the velocities of X-axis and Y-axis were v_{xt} and v_{yt} respectively, the displacements at *t* moment were as followed according to the displacement formula, x_t, y_t were the displacement of X-axis and of Y-axis respectively.

$$\begin{cases} x_t = x_s + \int_0^t V_{xt}\,dt \\ y_t = y_s + \int_0^t V_{yt}\,dt \end{cases} \tag{2}$$

So when

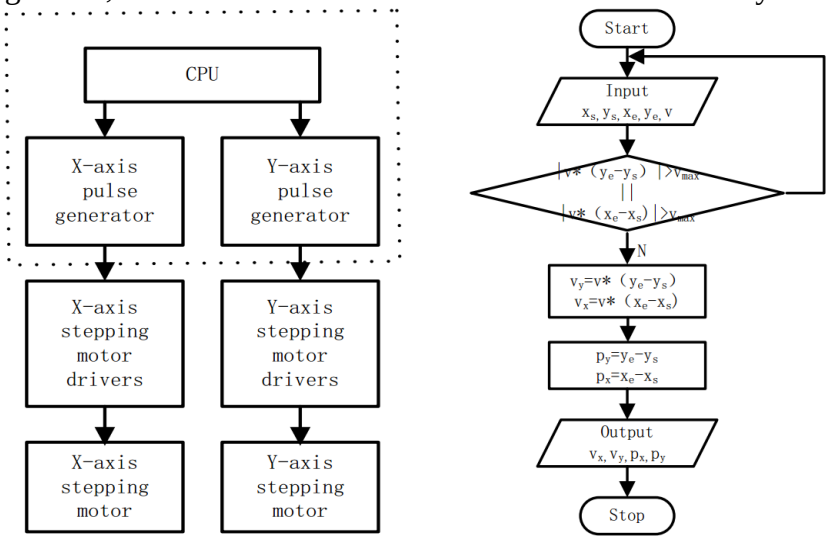
$$v_{yt} = k \times v_{xt} \tag{3}$$

substituted formula (3) into formula (2) and could achieve the formula (4).

$$\begin{aligned} y_t &= y_s + \int_0^t k \times v_{xt}\,dt \\ y_t &= y_s + k\int_0^t v_{xt}\,dt \\ y_t &= y_s + k(x_t - x_s) \end{aligned} \tag{4}$$

From the formula (4), it can be concluded that the point (x_t,y_t) was in the line.Therefore,, through control the ratio of the speed of the motor of the X-axis over that of the Y-axis in theory, the point

after a certain time should also fall on the line. And the linear interpolation algorithm was viable theoretically. Before describing the algorithm in this paper, it is necessary to present the hardware structure of this algorithm, which is also the basic hardware structure of many traditional algorithm.



(a) (b)

Fig. 2 (a) Basic hardware structure of the algorithm; (b) Technological process of the algorithm

Fig. 2 (a) illustrates the basic hardware structure of the algorithm, inculding CPU, X-axis pulse generator, Y-axis pulse generator, X-axis stepping motor drivers, Y-axis stepping motor drivers, X-axis stepping motor and Y-axis stepping motor. The pulse generators, for which frequency and number of the pulses can be set, and can be chip peripherals, such as timers, or off chip equipment. CPU sents instructions containing the information of frequency and the number of the pulses to the relative pulse generator. Then the generator produced the pulses meeting the requirements of instructions to the related drivers. At last, the related driver initiated the rotation of the motor.

Fig. 2 (b) illustrates the method of linear interpolation, and it was based on the principle and the structure mentioned above. Since the stepping motor has a maximum rotation, it should be ensure that the last velocity is not bigger than the maximum velocity v_{max}. Otherwise, they should be entered again. When entering the program correctly, the start point coordinate(x_s,y_s), the end point coordinate(x_e,y_e), and the common factor of the velocity of the X-axis and Y-axis should be input. In the algorithm, the time unit was the equivalent pulse duration, and the velocity unit was pulses per second in order to facilitate error analysis. As the max velocity vmax was very high, in most case, the input numbers will meet the requirements. If the length of the line goes beyond the requirements, the line will be divided into small lines. After the input of numbers, v_y standing for the velocity of Y-axis can be calculated from formula (5).

$$v_y = v(y_e - y_s) \tag{5}$$

And v_x standing for the velocity of X-axis can be solved from formula (6).

$$v_x = v(x_e - x_s) \tag{6}$$

In addition, p_y which was the symbol of the numbers of pulses sent by the Y-axis generators can be obtained from formula (7).

$$p_y = y_e - y_s \tag{7}$$

And p_x which was the symbol of the numbers of pulses sent by the X-axis generators can be obtained from formula (8).

$$p_x = x_e - x_s \tag{8}$$

As x_s, y_s, x_e, y_e and v were all integers, so v_y,v_x,p_x,p_y were all integer. There were no errors in the calculation. From Fig. 2(b), for most of the lines, the algorithm mentioned in this paper only needs the CPU to do one interpolation to get the frequency and the numbers of the pulses. The resultant track of

the X-axis and the Y-axis will be an approximate straight-line , it pulse generators according to the CPU instruct.

3. Error Analysis

In this paper, the error referred in particular to the error caused by the interpolation algorithm. So the error caused by the error of the mechanical and error caused by stepping motor was ignored.

Assuming that there was a line segment, the start point was (x_s,y_s), and the end point coordinate was (x_e,y_e). And v was set the common factor of the last speed. The unit of v was pulses per second, and the length unit was equivalent pulse. v_x and v_y were set as the symbols of velocity of X-axis motor and velocity of Y-axis respectively, which can be regarded as the frequency of X-axis pulse generator and Y-axis pulse generator. When the stepping motor takes continuous step motion, the speed of the rotation of the stepping motor is strictly dependent on the frequency input. However, in order to facilitate analysis, it was assumed that the stepper motor driver received a falling edge or a rising edge of the motor to rotate a pulse. Therefore, if the clock frequency of the pulse generator was f, and the duty cycle of the pulse generator was 50% at the moment t.

$$x_t = round(\frac{v_x t}{f}) + x_s \tag{9}$$

$$y_t = round(\frac{v_y t}{f}) + y_s \tag{10}$$

And $0 \le x_t \le p_x$, $0 \le y_t \le p_y$ round was a function that returns the rounding of the data. The conclusion will be obtained by classified discussion. When $y_e=y_s$, y_t was equal to y_s all the time, so (x_t, y_s) must be a point of the straight line $y=y_s$. And from$0 \le x_t \le p_x$, the error was 0.When $x_e=x_s$, the error was zero similarly. When the formula was (11), assuming that there was a point (x_t,y_t) in the line whose coordinate was and the formula (12) could be achieved.

$$y_e - y_s > x_e - x_s > 0 \tag{11}$$

$$(x_e - x_s)(y_t - y_s) = (y_e - y_s)(x_t^{'} - x_s) \tag{12}$$

According to the rounding rule, from the formula (10), it was concluded formula (13) and formula (14).

$$\frac{v_y t}{f} < y_t - y_s + 0.5 \tag{13}$$

$$\frac{v_y t}{f} \ge y_t - y_s - 0.5 \tag{14}$$

And from formula (13) and formula (14), it was concluded formula (13) and formula (14).

$$\frac{y_e - y_s}{x_e - x_s}\frac{v_x t}{f} < \frac{y_e - y_s}{x_e - x_s}(x_t^{'} - x_s) + 0.5 \tag{15}$$

$$\frac{y_e - y_s}{x_e - x_s}\frac{v_x t}{f} \ge \frac{y_e - y_s}{x_e - x_s}(x_t^{'} - x_s) - 0.5 \tag{16}$$

The formula (13) and the formula (14) were simplified and could conclude formula (17) and the formula (18).

$$\frac{v_x t}{f} < x_t^{'} - x_s + 0.5\frac{x_e - x_s}{y_e - y_s} \tag{17}$$

$$\frac{v_x t}{f} \ge x_t^{'} - x_s - 0.5\frac{x_e - x_s}{y_e - y_s} \tag{18}$$

According to the rule of rounding and formula (9), it was concluded formula (19) and formula (20).

$$\frac{v_x t}{f} < x_t - x_s + 0.5 \tag{19}$$

$$\frac{v_x t}{f} \geq x_t - x_s - 0.5 \tag{20}$$

Combined the two formula and formula (17), formula (18), we can get the formula (21) and the formula (22).

$$x_t - x_s - 0.5 < x_t^{'} - x_s + 0.5\frac{x_e - x_s}{y_e - y_s} \tag{21}$$

$$x_t - x_s + 0.5 > x_t^{'} - x_s - 0.5\frac{x_e - x_s}{y_e - y_s} \tag{22}$$

The formula (21) and formula (22) could be simplified to the formula (23) and the formula (24).

$$x_t - x_t^{'} < 0.5 + 0.5\frac{x_e - x_s}{y_e - y_s} \tag{23}$$

$$x_t - x_t^{'} > -0.5 - 0.5\frac{x_e - x_s}{y_e - y_s} \tag{24}$$

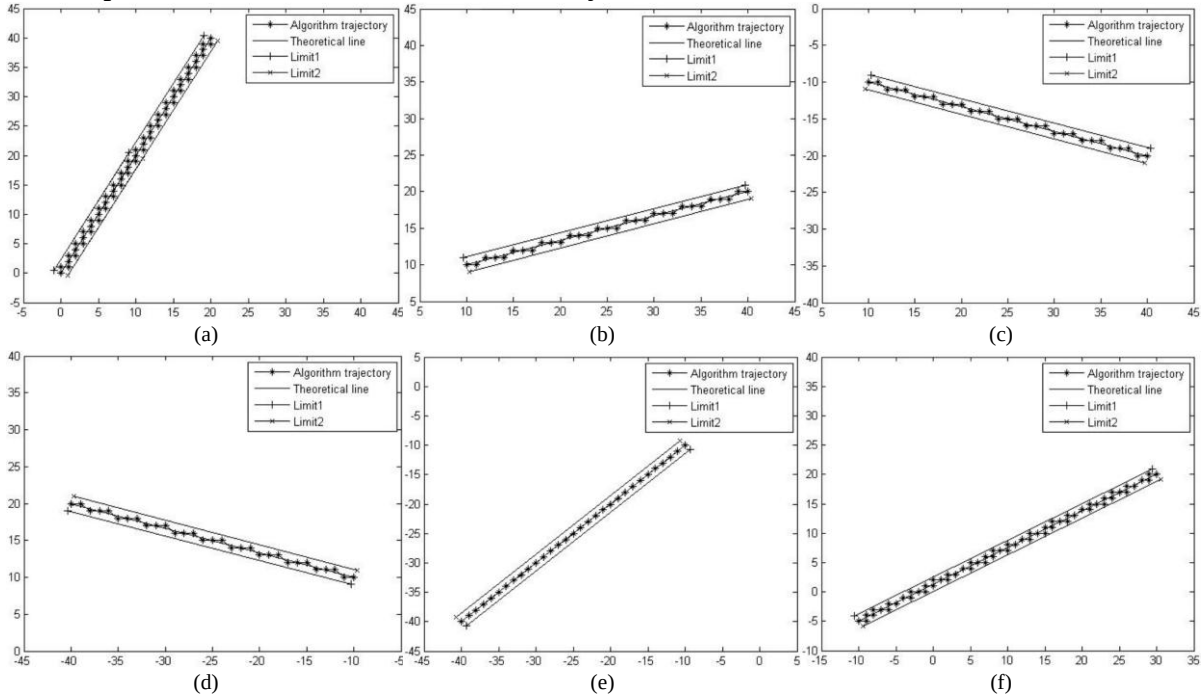
From the formula (11), $-1 < x_t - x_t^{'} < 1$, so the distance between and was smaller than 1. It can be concluded that the distance between point and theoretical line was smaller than 1. It also can conclude that the error was smaller than one.

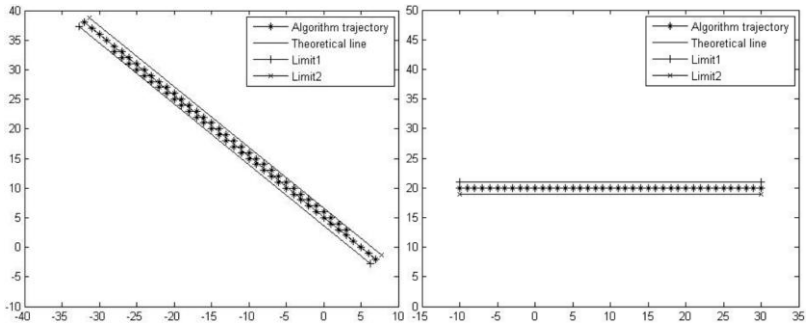
4. Simulation Analysis

Herein, MATLAB was used to simulate the algorithm of a line segment, and the trajectory of the algorithm was controlled within an error band to ensure that the algorithm error was smaller than 1. The simulation result was further compared with that simulated using point by point algorithm in terms of speed and interpolation calculation times.

4.1 Error simulation.

According to the principle of linear interpolation proposed in this paper, MATLAB was used to simulate this algorithm, taking Line 1 with the start point at coordinate (0,0), end point at (20, 40), Line 2 with start point at (10, 10), end point at (40, 20), Line 3 with start point at (10, -10), end point at (40, -20), Line 4 with start point at (-10,10), end point at (-40,20), Line 5 with start point at (-10,-10), end point at (-40, -40), Line 6 with start point at (-10, -5), end point at (30, 20), Line 7 with start point at (7, -2), end point at (-32, -38) and Line 8 with start point at (-10, 20), end point at (30, 20) as an example. The common factor of the velocity was set as 5000.





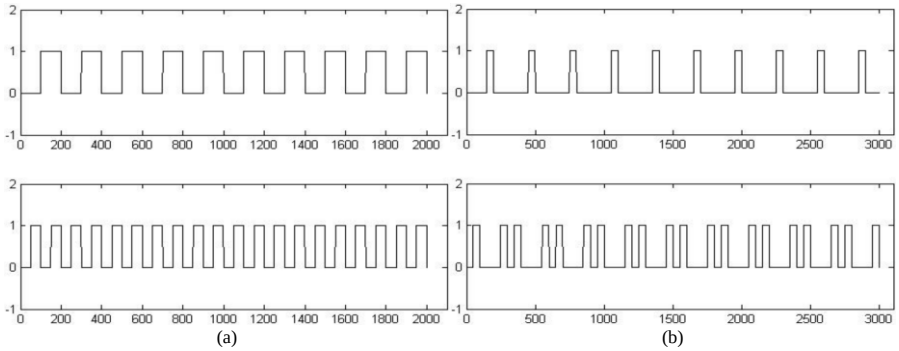
(g) (h)

Fig. 3 Algorithm trajectory: (a) trajectory for Line1; (b) trajectory for Line2; (c) trajectory for Line3; (d) trajectory for Line4; (e)trajectory for Line5; (f) trajectory for Line6; (g) trajectory for Line7; (h) trajectory for Line8

In Fig. 3, the red line was the algorithm trajectory of the resultant track of X-axis and Y-axis. The green line was the theoretical line. Limit1 and Limit2 referred to the two lines which were displaced from the theoretical line with the distance of 1. From Fig. 3, the error of the end point was zero. All the trajectory was between the Limit1 and Limit2. It can be conclude that the error of the algorithm was smaller than 1.

4.2 Velocity simulation.

The rotation speed of the stepping motor is in proportion to the frequency of the pulses input when the motor is making continuous rotation. We took the linear interpolation as a continuous rotation when we carried out the simulation. The velocity of the motor was calculated from the frequency of the pulses input into the related stepping motor driver, which was equal to the frequency of pulses generated by the related pulse generator. MATLAB was used to simulate the pulse generator and compare the linear interpolation algorithm put forward in this paper with the point by point comparison method. Taking line with the start point at (0,0), end point at (10, 20) as example, the following wave Figs were produced.



(a) (b)

Fig. 4 Waveform: (a) this paper algorithm; (b) point by point comparison method

The upper graph of Fig. 4(a) and Fig. 4(b) was the waveform of the X-axis pulses generator. The lower graph of Fig 4(a) and Fig 4(b) was the waveform of the Y-axis pulses generator. Fig. 4(a) showed that the frequency of the pulses generated by X-axis and Y-axis pulse generators was maintained the same during the whole linear interpolation procedure. The duty ratio was kept at 50%. It can be concluded that the velocities of X-axis and Y-axis stepping motor were kept the same all the time. Fig. 4(b) showed that the frequency of pulse generated by Y-axis pulse generator was changed during the whole linear interpolation procedure. It can be concluded that the velocity of Y-axis stepping motor was changed.

In the comparison of Fig. 4(a) with Fig. 4(b), when the velocities were set at the same value, the time used by the linear interpolation algorithm proposed in this paper was shorter than the time used by point by point comparison method. It can be conclude that the linear interpolation algorithm proposed in this paper was faster than point by point comparison method.

4.3 Interpolation Operation Times.

From the Fig. 2(b), the algorithm proposed in this paper only took one interpolation operation during the whole process to get the information the pulses generators need, namely the related frequency of pulses, and the related number of pulses, and send to the related pulsed generator.Taking Line 1 with start point at (0,0) and end point at (20,40) as example, MATLAB was used to simulate the operations times consumed by point by point comparison method.

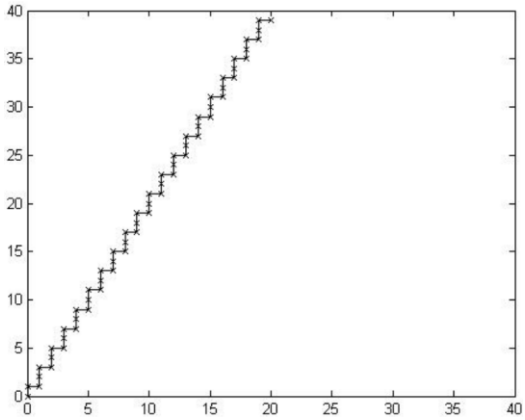


Fig. 5 Simulation of operation times point by point comparison method

The symbol " *" represented one interpolation operation. In the Fig. 5, there were 60 "*". The operation times consumed by point by point comparison method was 60 to interpolate Line 1, indicating hight consumption of CPU sources.

5. Conclusion

This paper presents a new method that keeps the ratio of the velocity of the X-axis motor and the Y-axis motor being equal to the slope of theoretical line to complete linear interpolation based on the principle of velocity synthesis.And it was confirmed that the theoretical line was less than one pulse equivalent by theoretical analysis and MATLAB Simulation. The velocities of X-axis and Y-axis stepping motor of the linear interpolation algorithm proposed were kept the same during the whole linear interpolation procedure. The velocities simulated by point by point comparison method were changed during the linear interpolation. The average velocities of point by point comparison method were hard to control. And the linear interpolation algorithm proposed in this paper was faster than point by point comparison method at the same pre-set velocity. The linear interpolation algorithm proposed in this paper only needed one interpolation operation during the whole linear interpolation procedure.

Acknowledgments

This work was financially supported by the Science and Technology Department of Guangdong Province of China (grants number 2014A010104006) and China scholarship council.

References

[1] Liang H, Hong H, Svoboda J, etal. "A combined 3D linear and circular interpolation technique for multi-axis CNC machining" [J]. TRANSACTIONS-AMERICAN SOCIETY OF MECHANICAL ENGINEERS JOURNAL OF MANUFACTURING SCIENCE AND ENGINEERING, (2002) 124(2), p.305-312.

[2] Lijun L, Weitao B, Wei S, etal. "Research on a new linear interpolation algorithm of NURBS curve"[C]//Mechanical and Automation Engineering (MAEE), 2013 International Conference on. IEEE, (2013), p.80-84.

[3] X.Zhiming, C.Jincheng, F.Zhengjin, etal. “Performance Evaluation of a Real-Time Interpolation Algorithm for NURBS Curves” [J].International Journal of Advanced Manufacturing Technology, (2002), p.270-276.

[4] Erkorkmaz K, Altintas Y, etal. “High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation” [J]. International Journal of machine tools and manufacture, (2001)41(9) p.1323-1345.

[5] Farouki R T, Shah S, etal. “Real-time CNC interpolators for Pythagorean-hodograph curves” [J]. Computer Aided Geometric Design, (1996) 13(7) p.583-600.

[6] WU Y, SHANG Y, etal. “Study on Control Algorithm for Micro-line Interpolation End-point” [J]. Modular Machine Tool & Automatic Manufacturing Technique, (2006) 2: 000.

[7] Valentino J, Goldenberg J. “Introduction to computer numerical control (CNC)” [M]. Englewood Cliffs: Prentice Hall, (2003).

[8] Koren Y. “Computer Control of Manufacturing Systems” [M].New York: McGraw-Hill, (1983).

[9] Li W, Fan N, Yang C, etal. “The Study for Algorithm of Linear Interpolation Based on Bipolar Coordinates” [J]. Intelligent Computation Technology & Automation International Conference on, (2010)3 p.977-979.

[10] Wei Y, Baosheng Y, Xinqiao Q, etal. “Optimization of point-by-point comparison linear interpolation algorithm” [J]. Journal of Wuhan University of Science and Technology, (2012) 3: 017.

[11] Dumitru D, Strajescu E, etal. “Study of the CNC lathe control parameter influence upon the contouring accuracy at the linear interpolation” [J]. University" Politehnica" of Bucharest Scientific Bulletin, Series D: Mechanical Engineering, (2010)72(2), p.123-134.