International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 1, January 2015, PP 29-34
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Fast Computation of Cubic Bernstein Polynomials in CNC Interpolation

ZENG Yu

Research Center of GSK CNC Equipment Co., Ltd,
Guangzhou City, China
zengyukey@126.com

WANG Xingbo

Department of Mechatromics
Foshan University, Foshan City, Chian
wxbmail@msn.com

Abstract: *CNC interpolation is essentially a numerical approximation in mathematics. The Bernstein polynomials have played an important role in the area of approximation ever since it was created. Hence the study on their rapid computations is valuable in CNC interpolation. Classical studies on rapid computations always focus on improving the computational efficiency by software algorithms and seldom associate the problem with the working process of the computer's processor. This paper introduces a different thought. The paper presents a method that can rapidly compute the cubic Bernstein polynomials in accordance with the working traits of computers' processors. Consequently, the computational routine can be implemented both by software and by hardware, which greatly enhances the computational efficiency. The principles and process of the method are presented together with its application in shape optimization.*

Keywords: *Bernstein polynomials, CNC interpolation, rapid computation*

1. INTRODUCTION

CNC interpolation in industrial applications is essentially implemented by means of numerical approximation in mathematics (Haslinger & Makinen 2003). This is because a parameterized curve on industrial model is mathematically expressed with three real functions, say in a 3-dimensional space. Hence numerical optimization lays the foundation for the CNC interpolations.

The Bézier curves and surface are fundamental tools in computer graphics (CG), computer aided design(CAD), computer aided manufacture(CAM) and CNC. Theory of computer aided geometric design (CAGD) shows that the Bézier curves and surfaces are all generated from the Bernstein polynomials (Farin 1998). Hence a rapid computation of the Bernstein polynomials directly leads to a rapid computation of the Bézier curves and surfaces. Besides, owing to the important position the Bernstein polynomials play in the area of function approximation (Chen 2000, Gzyl & Palacios 2003, Pallini 2005, Steffens 2006), a study of their rapid computation is also valuable in theory of approximation(Nataray & Arounassalame 2007).

Classical studies on rapid computations always focus on improving computational efficiency by software algorithms and seldom associate the problem with the working process of the computer processor. Recently, we have put forward a new method for high speed CNC interpolation(Wang Xingbo,2011 &2014). The method can be implemented both by software and by hardware. This paper presents the mathematical foundation of the method. Section 2 shows the principle and process of the method and section 3 presents an application in shape optimization.

2. FAST COMPUTATION OF CUBIC BERNSTEIN POLYNOMIALS

A Bernstein polynomial $\beta(t)$ of degree n is defined by

$$\beta = \sum_{i=0}^{n} \alpha_i B_{i,n}(t), 0 \le t \le 1 \tag{1}$$

where $\alpha_i, (i = 0,1,2,...,n)$ are real numbers, and $B_{i,n}(t)$ are the Bernstein basis such that

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, i = 0,2,...,n$$

It is easy to show that the Bernstein basis fit the following properties

$$B_{i,n-1}(t)=\frac{n-i}{n}B_{i,n}(t)+\frac{i+1}{n}B_{i+1,n}(t) \tag{2}$$

$$B'_{i,n}(t)=n(B_{i-1,n-1}(t)-B_{i,n-1}(t)) \tag{3}$$

from which the subdivision property of the Bernstein polynomials is stated as the following lemma.

Lemma Denote the Bernstein polynomial (1) by

$$\beta(t)=<\alpha_0,\alpha_1,...,\alpha_n,t> \tag{4}$$

then it holds

$$\beta(t)=\begin{cases} <\alpha_{0,0},\alpha_{0,1},...,\alpha_{0,n},\dfrac{t}{u}>, 0\le t\le u \\ <\alpha_{0,n},\alpha_{1,n-1},...,\alpha_{n,0},\dfrac{t-u}{1-u}>, u\le t\le 1 \end{cases} \tag{5}$$

Where $\alpha_{i,j}$ is given by

$$\begin{cases} \alpha_{i,0}=\alpha_i, i=0,1,...,n \\ \alpha_{i,j}=(1-t)\alpha_{i,j-1}+t\alpha_{i+1,j-1}, i=0,1,...,n-j \end{cases} \tag{6}$$

The cubic Bernstein polynomials are particularly useful in industrial applications. Let A_0,B_0,C_0,D_0 be four real numbers; then the cubic Bernstein polynomial $\beta(t)=<A_0,B_0,C_0,D_0,t>$ is given by

$$\beta(t)=A_0(1-t)^3+3B_0t(1-t)^2+3C_0t^2(1-t)+D_0t^3 \tag{7}$$

By the subdivision property, taking $t=0.5$, which is called a mid-subdivision, yields two cubic Bernstein polynomials, $\beta^l=<A_1^l,B_1^l,C_1^l,D_1^l,t>$ and $\beta^r=<A_1^r,B_1^r,C_1^r,D_1^r,t>$, which fit the following relationship

$$\begin{cases} A_1^l=A_0, B_1^l=\dfrac{1}{2}(A_0+B_0) \\ C_1^l=\dfrac{1}{4}(A_0+2B_0+C_0) \\ D_1^l=\dfrac{1}{8}(A_0+3B_0+3C_0+D_0) \end{cases}, \begin{cases} A_1^r=\dfrac{1}{8}(A_0+3B_0+3C_0+D_0) \\ B_1^r=\dfrac{1}{4}(B_0+2C_0+D_0) \\ C_1^r=\dfrac{1}{2}(C_0+D_0), D_1^r=D_0), \end{cases} \tag{8}$$

For convenience, the eight numbers, $A_1^l,B_1^l,C_1^l,D_1^l,A_1^r,B_1^r,C_1^r,D_1^r$, are denoted by two quaternions, $(A_1^l,B_1^l,C_1^l,D_1^l)$ and $(A_1^r,B_1^r,C_1^r,D_1^r)$, which are called, for convenience, an L-Bernstein Number and a xtitR-Bernstein Number, respectively. Obviously, they can be computed just by left-shift and addition. As is known, shift operation and addition are faster computations; hence a fast computation can be designed for the cubic Bernstein polynomial.

Now performing the mid-subdivision on $(A_1^l,B_1^l,C_1^l,D_1^l)$ and $(A_1^r,B_1^r,C_1^r,D_1^r)$ obtains 4 quaternions

$$(A_2^{l^2},B_2^{l^2},C_2^{l^2},D_2^{l^2}),(A_2^{lr},B_2^{lr},C_2^{lr},D_2^{lr}),(A_2^{rl},B_2^{rl},C_2^{rl},D_2^{rl}),(A_2^{r^2},B_2^{r^2},C_2^{r^2},D_2^{r^2})$$

where $l^2=ll$ means left to left and lr means right to left, and so are the meaning of other symbols.

Obviously, repeating the mid-subdivision process on the quaternions will result in a sequence of the quaternions:

$(A_k^{l^k}, B_k^{l^k}, C_k^{l^k}, D_k^{l^k})$, $(A_k^{l^{k-1}r}, B_k^{l^{k-1}r}, C_k^{l^{k-1}r}, D_k^{l^{k-1}r})$, ……, $(A_k^{r^{k-1}l}, B_k^{r^{k-1}l}, C_k^{r^{k-1}l}, D_k^{r^{k-1}l})$, $(A_k^{r^k}, B_k^{r^k}, C_k^{r^k}, D_k^{r^k})$

And each quaternion in the sequence will give a value of $\beta(t)$ by the first or the fourth component.

Conventionally, the above computation process is designed and implemented by a recursion routine. However we design a new routine for it because, as is known, the recursion routine is less efficient.

Let BCU denote a Basic Calculation Unit, such that it performs the calculation defined in the equation (8). That is, a BCU will outputs two quaternions, $(A_1^l, B_1^l, C_1^l, D_1^l)$ and $(A_1^r, B_1^r, C_1^r, D_1^r)$, and a number $res = D_1^l = A_1^r$ byan input A_0, B_0, C_0, D_0, as illustrated in figure 1.

Now this BCU is used to design a generator of cubic Bernstein polynomials, as illustrated in figure 2.

It is shown that three BCUs are used. The first one is used for the initial subdivision, which turns the quaternion (A_0, B_0, C_0, D_0) into $(A_1^l, B_1^l, C_1^l, D_1^l)$ and $(A_1^r, B_1^r, C_1^r, D_1^r)$. The later two quaternions are immediately used as theinputs of the other two BCUs, which are called, for convenience, L-BCU and R-BCU respectively, and perform afterwards recycling computations. The L-memory and R-memory are two FIFO(First in First Out) storagestructures to store the output of the L-BCU and R-BCU respectively. The L-result and R-result are another two storage structures to store the final results. This generator can automatically generate all the values of a Bernstein polynomial. The mechanism is as follows:

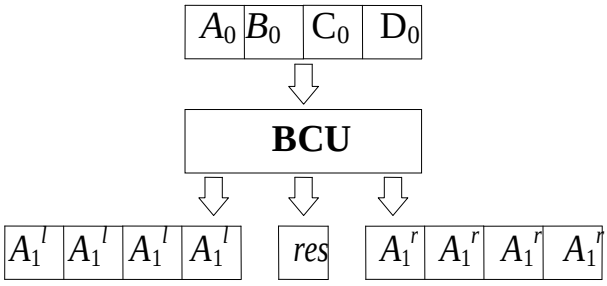

Fig1. *A BCU and its inputs and outputs*

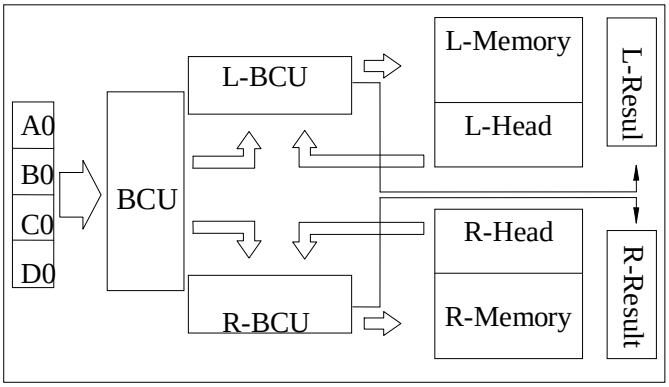

Fig2. *A generator of cubic Bernstein polynomials*

Seen from the figure 2, three outputs, an L-Bernstein Number L, a R-Bernstein Number R and a result number res, denoted also by *R*, are obtained after the initial computation; and L and R are put to be inputs of the L-BCU and the R-BCU respectively. After the L-BCU and the R-BCU finish their first computations, 4 outputs are obtained, LL and LR from L-BCU, RL, RR from R-BCU. The two result numbers $LL = L^2$ and RL are stored into L-result and R-result, respectively. To continue the computation, LL and RL are used as the inputs of the L-BCU and the R-BCU for their next computations while LL and RR are respectively stored into L-memory and R-memory. When the L-BCU and the R-BCU finish their second computations, the two outputs, LLL and LLR, are stored into the L-memory, and other two ones, RLL and RLR, are stored into the R-

memory while the two numbers, L^3 and RL^2, are stored into L-result and R-result, respectively. The stored LR, RR are then fetched to be the inputs of the L-BCU and the R-BCU to perform their third computations.

Keeping this process leads to a continuous computation in the L-BCU and the R-BCU and a data-stored sequence in the four memory structures as shown in figure 3. Note that the data stored in the L-result and the R-result form a storage structure of a full binary tree, and are easy to fetch.

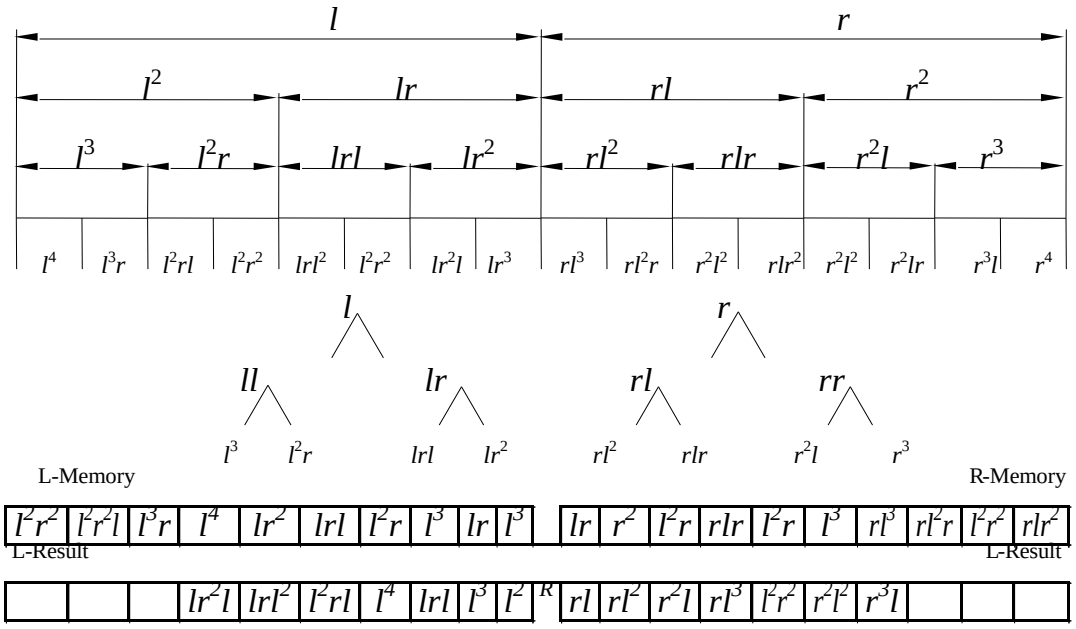

Fig3. *Data storage in memory structures*

3. APPLICATION IN SHAPE OPTIMIZATIONS

A cubic Bézier curve is essentially constructed from the cubic Bernstein polynomials. Hence it can be fast commutated via the above process. For a given sequence of N control points $\{\boldsymbol{P}_i\}_{i=1}^N$ together with another boundary points $\boldsymbol{P}_0$, $\boldsymbol{P}_{N+1}$, the i-th cubic Bézier interpolating segment that passes through $\boldsymbol{P}_i, \boldsymbol{P}_{i+1}$ is defined by

$$r_i(t) = P_i(1-t)^3 + 3(P_i + \eta P_{i-1}P_{i+1})t(1-t)^2 + 3(P_{i+1} - \mu P_i P_{i+2})t^2(1-t) + P_{i+1}t^3 \tag{9}$$

where $0 < \eta, \mu < 1$.

In fact, by the eq. (9), it yields

$$r_i(0) = P_i, r_i(1) = P_{i+1} \tag{10}$$

$$r_i(0) = \eta P_{i-1}P_{i+1}, r_i(1) = \mu P_i P_{i+2} \tag{11}$$

Then if we take $\eta = \mu$, the piecewise curve

$$\Gamma = \bigcup_{i=1}^{N-1} r_i \tag{12}$$

is a C1-continuous curve that pass through all the control points $\{\boldsymbol{P}_i\}_{i=1}^N$. It is of course a GC^1-continuous when $\eta \neq \mu$. Consequently, when i goes through 1,2,…,N, the curve Γ passes through all the control points $\{\boldsymbol{P}_i\}_{i=1}^N$ and r_i has the same tangent vector at the end-point $r_i(1)$ with that of r_{i+1} at the start-point $r_{i+1}(0)$.

Normally, the curve (9) can be calculated by the De Casteljau's algorthm. However, one will see that, it can be calculated by the generator designed in previous section. In fact, the equation (9) can be written by

$$r_i(t) = P_i BZ_0(t) + P_{i+1} BZ_1(t) - \mathbf{BZ}(t) \tag{13}$$

where

$$BZ_0(t) = (1-t)^3 + 3t(1-t)^2 + 3\mu t^2(1-t) + t^3 \tag{14}$$

$$BZ_1(t) = (1-t)^3 + 3\eta t(1-t)^2 + 3t^2(1-t) + t^3 \tag{15}$$

$$\mathbf{BZ}(t) = 3\eta P_{i-1} t(1-t)^2 + 3\mu P_{i+2} t(1-t)^2 + P_i(1-t)^3 + P_{i+1} t^3 \tag{16}$$

Since $BZ_0(t)$ and $BZ_1(t)$ are two cubic Bernstein polynomials and the $\mathbf{BZ}(t)$ is composed of three cubic Bernstein polynomials, it is sure that the curve (13) can be fast calculated with the generator of cubic Bernstein polynomials.

Meanwhile, it can see that, the calculation with eq.(13) is of high precision. In fact, the eq.(13) shows different computational traits from the eq.(9). By the eq.(9), it is seen that,in order to perform the De Casteljau's algorithm, it should first to compute the coefficients $\boldsymbol{P}_i + \eta \boldsymbol{P}_{i-1} \boldsymbol{P}_{i+1}$ and $\boldsymbol{P}_{i+1} - \mu \boldsymbol{P}_i \boldsymbol{P}_{i+2}$. This includes three times of subtraction, two times of multiplication and two times of addition. During these computations, especially the subtraction, computational errors will occur. These errors will obviously affect the the final precision of computation. In addition, the three times of subtraction is also a factor that decreases the computational efficiency. On the other hand, by eq.(13), there are no such extra errors and the computational efficiency is improved since only one subtraction is performed. Note that, the three items on the right side of the eq.(13) are independent each other, thus they can of course be computed by parallel mode. All these show that, calculation by the new method is of a good attempt.

ACKNOWLEDGEMENTS

The author's research work is supported by the national Ministry of science and technology under project 2013GA780052, Department of Guangdong Science and Technology under projects 2012 B010600018 and 2012B011300068, Foshan Bureau of Science and Technology under projects 20 13AG10007, 2012HC100131 and 2011AA100021, and Chancheng government under projects 20 11GY006, 2011B1023, 2011A1025, 2011A1030, 2012B1011, 2013B1018 and 2013A1021. The a uthors sincerely present thanks to them all.

REFERENCES

[1] G Farin. Curves and Surfaces for Computer Aided Geometric Design. Academic Press, (1998).

[2] J Haslinger, R A E Makinen. Introduction to Shape Optimization: Theory, Approximation, and Computation. Society for Industrial Mathematics, (2003).

[3] K G Steffens. The History of Approximation Theory: From Euler to Bernstein. Birkhauser Boston, (2006).

[4] Y Y Chen. Some Thoughts on the Polynomial Approximations to a Given Function. Chinese Journal of Physics, 38(5), 926-938, (2000).

[5] H Gzyl, J L Palacios. On the Approximation Properties of Bernstein Polynomials via Probabilistic Tools. Boletín de la Asociacion Matem′atica Venezolana, 10(1), 5-12, (2000).

[6] P S V Nataray, M Arounassalame. A New Subdivision Algorithm for the Bernstein Polynomial Approach to Global Optimization. International Journal of Automation and Computing, 4(4), 342-352, (2007).

[7] A Pallini.Bernstein-type Approximation of Smooth Functions. Statistica, (2),159-190.(2005)

[8] Wang Xingbo, Zeng Yu. Study of Data Structure in Development of High-Speeded CNC Interpolators , ICMTMA2011 , pp.393-396

[9] WANG Xingbo. "A Rapid CNC Interpolator with Capacity of Parallel Computation", Chinese Patent, ZL201010593567.6, January , 2014

AUTHORS' BIOGRAPHY

Zeng Yu, was born in Jiangxi, China. He got his Master degree from National University of Defense Technology in 2007, and the has worked as an technical engineer in Research Center of GSK CNC Equipment Co., Ltd, being in charge of the development of various products of the company, which is one of the major producer of CNC equipments in China.

WANG Xingbo was born in Hubei, China. He got his Master and Doctor's degree at National University of Defense Technology of China and had been a staff in charge of researching and developing CAD/CAM/NC technologies in the university. Since 2010, he has been a professor in Foshan University, still in charge of researching and developing CAD/CAM/NC technologies. Wang has published over 70 papers and obtained more than 15 patents in mechanical engineering