Improved Algebraic Algorithm On Point Projection For Bézier Curves

Xiao-Diao Chen, Yin Zhou, Zhenyu Shu, Hua Su, Jean-Claude Paul

Improved Algebraic Algorithm On Point Projection For Bézier Curves

Xiao-Diao Chen[1,3], Yin Zhou[1], Zhenyu Shu[1], Hua Su[2], Jean-Claude Paul[2,3]
1. Ningbo Institute of Technology, Zhejiang University, Zhejiang, 315100, P.R. China
2. School of Software, Tsinghua University, Beijing 100084, P.R. China
3. INRIA, France
littlerain-szy@163.com myedge-2000@tom.com chenxd00@mails.tsinghua.edu.cn
huas@mails.tsinghua.edu.cn paul@tsinghua.edu.cn

Abstract
This paper presents an improved algebraic pruning method for point projection for Bézier curves. It first turns the point projection into a root finding problem, and provides a simple but easily overlooked method to avoid finding invalid roots which is obviously irrelative to the closest point. The continued fraction method and its expansion are utilized to strengthen its robustness. Since NURBS curves can be easily turned into Bézier form, the new method also works with NURBS curves. Examples are presented to illustrate the efficiency and robustness of the new method.

Keywords
Point projection; NURBS curve; Continued fraction method; Algebraic pruning method

1 Introduction

The point projection problem is to find the closest point on the curve to a given point, and return the corresponding parameter of the closest point [5; 11; 14; 22; 18; 27]. It is very important in geometric modeling, computer graphics and computer vision [11; 14; 22; 18]. It is also essential for the B-spline curve fitting problem [30].

Given a point $\mathbf{p}$ and a Bézier curve $\mathbf{q}(u)$ of degree p, the point projection problem can be described mathematically as to find u^*, such that

$$\|\mathbf{p}-\mathbf{q}(u^*)\| = \min\{\|\mathbf{p}-\mathbf{q}(u)\| \; | u \in [0,1]\}. \quad (1)$$

If $u^* \notin \{0,1\}$, then Mortensen and Zhou [16; 33] give a necessary condition mathematically as follows,

$$(\mathbf{p}-\mathbf{q}(u^*)) \cdot \mathbf{q}'(u^*) = 0, \quad (2)$$

where $\mathbf{q}'(u)$ denotes the derivative of $\mathbf{q}(u)$. The closest point can be obtained by solving Equation (2). There are many methods on solving all the real roots of a polynomial equation, such as Newton's method [2; 9; 17; 22], Sturm sequence method [1; 23; 31], Bézier clipping method [20; 26], hybrid method [17; 23], and so on [19; 18; 29].

However, not all roots are necessary to be computed. The pruning technique based on the subdivision of the curve is able to reduce the unnecessary computation [9; 12; 13; 14; 22], whose aid is to obtain a good initial value for the Newton-Raphson method. The Newton-Raphson method needs a good initial value for achieving the convergent result [23; 28]; otherwise, it may give a wrong answer or even fail [14; 21; 22]. Each subdivision step of a Bézier curve of degree p in the space $\mathbb{R}^m$ requires $p(p+1)m/2$ additions and $p(p+1)m/2$ multiplications for the non-rational case, or $p(p+1)(m+1)/2$ additions and $p(p+1)(m+1)/2+2p+1$ multiplications for the rational case [4; 6; 20; 21]. To obtain a good enough initial value, it is necessary to subdivide the curve in a very high lever, which leads to too much expensive computation time on subdivision.

The algebraic method of solving the roots of a polynomial equation costs less computation on subdivision, and the geometric method based on the subdivision of the Bézier curves prunes some unnecessary computation. Combining with these two methods, this paper presents an algebraic pruning method for the point projection problem for Bézier curves. The new method spends much fewer computation time on subdivision and obtains higher efficiency. Our method first isolates all the different real roots, then provides a method to prune most of the unnecessary roots, finally combines bisection method with Newton's method to subdivide the object function instead of the curve, until it obtains the exact solutions within a given tolerance.

We assume that the Bézier curve is defined by

$$\mathbf{q}(u) = \frac{\sum_{i=0}^{p} B_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{p} B_{i,p}(u) w_i} = \frac{\mathbf{V}(u)}{w(u)}, \;\; 0 \le u \le 1, w_i > 0, \quad (3)$$

where p is the degree of the curve, $\mathbf{P}_i$ are the control points in the space $\mathbb{R}^3$, u is the parameter of the curve, $\{w_i\}$ are the weights, and $\{B_{i,p}(u)\}$ are the pth-degree Bernstein basis functions. We assume that the closest point to the testing point $\mathbf{p}$ is not a end point of the curve.

This paper is organized as follows. Section 2 presents the outline of the new algorithm. Section 3 gives correspond-

ing complexity analysis, and shows several examples. Some conclusions are drawn at the end of this paper.

2 Outline of the algebraic pruning algorithm

Our algorithm has three steps. Firstly, we obtain the object polynomial function from Equation (2) and isolate all of its different real roots. Secondly, by using a simple but easily overlooked classification method, we prune most of the unnecessary roots. Finally, by combining the bisection method with the Newton's method, we subdivide the object function instead of the curve to obtain the exact roots and choose the resulting root.

Isolating the object polynomial equation

Substituting Equation (3) into Equation (2), and multiplying $w(u)^3$, we obtain the object polynomial equation

$$g(u) = (w(u)\mathbf{V}'(u) - w'(u)\mathbf{V}(u)) \cdot (w(u)\mathbf{p} - \mathbf{V}(u)) = 0. \tag{4}$$

If the curve $\mathbf{q}(u)$ is a non-rational Bézier curve, i.e., $w(u) = 1$, then the polynomial function $g(u)$ is of degree at most $2p$. If the curve $\mathbf{q}(u)$ is a rational Bézier curve, $g(u)$ becomes a polynomial of degree at most $3p - 1$. There are many methods for isolating all the real roots of a polynomial equation, such as the Sturm sequence method [1; 31],the nodal mapping method [12], the cone test method [9; 13], the Newton interval method [10], the Descartes method [7; 8; 25]. This paper utilizes the Sturm sequence method to isolate the real roots of $g(u)$. The Sturm sequence $\{S_i(u)\}_{i=0}^k$ for $g(u)$ can be constructed as follows.

$$\begin{aligned} &S_0(u) = g(u), \\ &S_1(u) = g'(u), \\ &S_i(u) = S_{i-2}(u) - R_i(u)S_{i-1}(u), \ \ i \geq 2, \end{aligned}$$

where $R_i(u)$ and $S_i(u)$ are the quotient and the remainder of $S_{i-2}(u)$ divided by $S_{i-1}(u)$, respectively. When the Sturm sequence has been constructed, it only needs n times multiplication and additions for each iterative step, where n is the degree of $g(u)$ [1; 31]. So it is very efficient to isolate the real roots. The precision requirement for computing the Sturm sequence may be quite high [24], and the similar continued fraction method is introduced to strengthen its robustness. Suppose that $g(u) = \sum_{j=0}^{n} b_j u^{n-j}$, $S_i(u) = \sum_{j=0}^{n-i} a_j^{(i)} u^{n-i-j}$, and $R_i(u) = T_i u + M_i$. We have

$$\begin{aligned} &a_j^{(0)} = b_j, \\ &a_j^{(1)} = (n-j)a_j^{(0)}, \\ &T_i = a_0^{(i-2)}/a_0^{(i-1)}, \\ &M_i = (a_1^{(i-2)} - T_i a_1^{(i-1)})/a_0^{(i-1)}, \\ &a_j^{(i)} = a_{j+2}^{(i-2)} - M_i a_{j+1}^{(i-1)} - T_i a_{j+2}^{(i-1)}. \end{aligned}$$

Thus, $\{a_j^{(i)}\}$ can be computed by using the similar continued fraction method. If $\{b_j\}$ are rational real number, then all coefficients $\{a_j^{(i)}\}$ of the functions in the Sturm sequence are described in fraction. When u is a rational real number, the corresponding value of $S_i(u)$ can be described in fraction, which can be computed accurately with long integer arithmetic. And the above method can be very robust.

Pruning technique based on a classification method

The square distance between point $\mathbf{p}$ and $\mathbf{q}(u)$ is

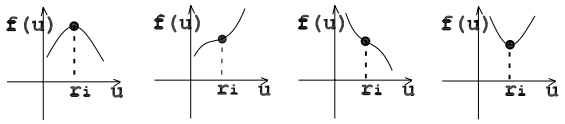
$$f(u) = \|\mathbf{p} - \mathbf{q}(u)\|^2.$$

The derivative of $f(u)$ is

$$f'(u) = \frac{g(u)}{w(u)^3}.$$

Since $w_i > 0$, we have $w(u) > 0, \forall u \in [0,1]$. Thus, $f'(u)$ has the same sign as $g(u)$ in the interval $[0,1]$. Let $(a_i, b_i) \in [0,1]$ be the ith interval obtained in the first step, which contains only one root of $g(u)$. Suppose $g(a_i) \neq 0$, $g(b_i) \neq 0$, and $r_i \in (a_i, b_i)$ is the root of $g(u)$. For each interval (a_i, b_i), according to the signs of $g(a_i)$ and $g(b_i)$, $f(u)$ can be classified into four cases, which are $(+,-)$, $(+,+)$, $(-,-)$ and $(-,+)$, as shown in Fig. 1(a),(b),(c) and (d). In the cases 1(a),(b) and (c), $f(u)$ doesn't reach the local minimum value at r_i. Thus, in these three cases, we needn't compute the root r_i. So we only need to consider the case when $g(a_i) < 0$ and $g(b_i) > 0$, as shown in the case 1(d).

By this simple but easily overlooked classification, we are able to prune most of the invalid intervals obtained in the first step.



(a) (b) (c) (d)

Figure 1: Four cases of $f(u)$ in a sub-interval.

Computing the remaining roots of $g(u)$

For each remaining interval (a_i, b_i), it requires to compute the corresponding root r_i of $g(u)$. The Newton method may converge rapidly, and the bisection method seems stable for calculating the roots. This paper combines the bisection method with Newton's method to compute the real roots, which is a tradeoff but reasonable way [32]. We only need to deal with the case when $g(a_i) < 0$ and $g(b_i) > 0$.

Algorithm 1: Bisection algorithm of computing a root of $g(u)$ in an interval.
Input: a tolerance τ, $g(u)$, an interval (a_i, b_i) such that $g(a_i) < 0$ and $g(b_i) > 0$, and there is a unique root of $g(u)$ in (a_i, b_i).
Output: The unique root r_i of $g(u)$.

1. If $b_i - a_i < \tau$, let $r_i = (a_i + b_i)/2$, go to Step (5);
2. Let $m_i = (a_i + b_i)/2$;
   Compute the value of $g(m_i)$;

3. If $g(m_i) = 0$, then $r_i = m_i$, go to Step (5);

4. Set $a_i = m_i$ when $g(m_i) < 0$, or set $b_i = m_i$ when $g(m_i) > 0$;
   Go to Step (1);

5. **End** of the algorithm.

When all the $\{r_i\}$ have been computed, we choose the value r^* such that $f(r^*) = \min\{f(u)|u \in \{r_i\} \bigcup \{0, 1\}\}$, and return the corresponding point $\mathbf{q}(r^*)$ as the closest point. The new algorithm for the point projection problem is generalized as follows.

Algorithm 2: Algorithm of point projection for a Bézier curve.
Input: A Bézier curve and a given point **p**.
Output: The closest point and the corresponding parameter u^*.

1. Compute the function $g(u)$ and $f(u)$;
   Isolate the different real roots of $g(u)$ into several intervals $\{(a_i, b_i)\}$ by using the Descartes method;

2. Prune the invalid intervals with a simple classification method;

3. For each remaining interval, combine the bisection method with Newton's method to calculate the unique root;

4. Choose the very root u^* such that $f(u^*) = \min\{f(u)|u \in \{r_i\} \bigcup \{0, 1\}\}$, and return the corresponding point $\mathbf{q}(u^*)$ as the closest point.

5. **End** of the algorithm.

3 Complexity analysis and examples

The pruning method based on the subdivision of the curve utilizes the geometric property of the curve, and we call it a geometric pruning method. Our method is based on the subdivision of the algebraic function $g(u)$, and we call it an algebraic pruning method.

We provide a simple complexity analysis between these two methods. For the convenience of comparison, we assume that both the geometric and algebraic pruning method will use the Newton-Raphson method when the remaining parameter interval is within a given tolerance after pruning process. Based on such assumption, the total number of pruning step can be taken as the same, and one needs to comparison the computation time for each pruning step. For the geometric pruning method, each pruning step consists of curve subdivision and geometric judging computation, here we only consider the computation time on curve subdivision. For the algebraic pruning method, there are two stages, one is to isolate the real roots, and the other is to prune the remaining intervals with the bisection method. In the bisection method, it is to evaluate the value of $g(u)$ in Equation (4). With Horner's method [15], the computation time for the bisection method is equal to the degree of $g(u)$. Table 1 shows the computation time comparisons between the geometric pruning method (**GPM**) and Sturm sequence method (**SSM**).

Table 1: Multiplication times for each pruning step

GPM	**SSM**	
	Isolation	Bisection
$r_1 p(p+1) + r_2$	$2\ d^*$	d^*

where
$\begin{cases} r_1 = 3/2, r_2 = 0, d^* = 2p - 1, \text{for non-rational case,} \\ r_1 = 2, r_2 = 2p + 1, d^* = 3p - 1, \text{ for rational case.} \end{cases}$

For each pruning step, the algebraic method is much faster that the geometric method. But general algebraic method will compute all the real roots, while the geometric method can prune part of invalid roots. Finally, sometimes the total computation time of the geometric method may be less. The new method is able to prune most of invalid roots and speeds up the algebraic method very much, which leads to less computation time than the geometric method, shown as examples 1 and 2. Here we select the methods in [14; 27] as the geometric pruning method (**GPM**). We also comparison the general algebraic method (**GAM**) with our improved algebraic method(**IAM**). All examples are implemented under Windows PC 1.7G with 512M memory.

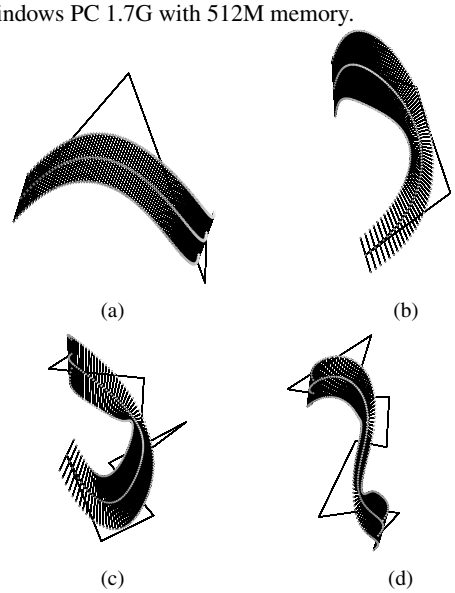


Figure 2: Point projection for B*é*zier curves of degree 3 (a), 6(b), 8(c) and 10(d).

Example 1. Fig. 2 shows projection examples for Bézier curves of degree 3,6,8 and 10, respectively. In these cases, we select 200 points for each Bézier curve. The corresponding results are shown in Table 2. In Table 2, T_g, T_a and T_i denote the corresponding computation time of **GPM**, **GAM**,

and **IAM** with the unit millisecond. As shown in Table 2, **IAM** are more rapid than **GPM** and **GAM**.

Table 2: Resulting computation time for the point projection problem of Bézier curves

Time	Fig 2(a)	Fig 2(b)	Fig 2(c)	Fig 2(d)
T_g	0.017ms	0.051ms	0.061ms	0.071ms
T_a	0.013ms	0.036ms	0.051ms	0.072ms
T_i	0.007ms	0.013ms	0.018ms	0.025ms

In Table 3, N_k denotes the number of points whose corresponding distance error are within the tolerance 10^{-k}, $k = 5, 6, \cdots, 9$. As shown in the Table 3, the numbers N_9 and N_8 in the improved method **IAM** are larger than those numbers in **GPM** and **GAM**.

Example 2. Fig. 3 shows point inversion examples for Bézier curves of degree 4,5,7 and 9, respectively. For each case, we select 201 points at parameter $\{i/200\}_{i=0}^{200}$ on the corresponding Bézier curve. The corresponding results are shown in Table 4. In Table 4, T_m denotes the corresponding computation time with the unit millisecond, and N_k denotes the number of points whose corresponding error are within the tolerance 10^{-k}. As shown in Table 4, **IAM** is more rapid than **GPM** method. For all the examples shown in Fig. 4, all the resulting points by **IAM** are within the tolerance 10^{-8}, and there are more points within the tolerance 10^{-6}, 10^{-7}, 10^{-8} or 10^{-9} in **IAM** than **GPM**.

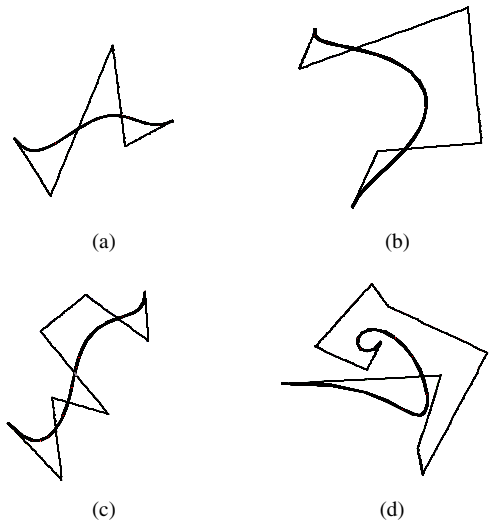


Figure 3: Point inversion for Bézier curves of degree 4 (a), 5(b), 7(c) and 9(d).

4 Conclusion

The algebraic method turns the point projection problem for Bézier curves into a root-finding problem of a polynomial equation. Usually, it will compute all the real roots. This paper provides an improved method, which reduce most of the computation for finding the invalid real roots. It also introduces a similar continued fraction method to improve the robustness of the new method. The new method can pruning most of the invalid roots as the geometric methods do, but the computation time for each pruning step is much less than the geometric methods, which are based on the subdivision of the curve itself. Examples are shown to illustrate the efficiency and the robustness of the new method.

Acknowledgement

The research was supported by the Chinese 973 Program (2004CB719403,2004CB318000), the National Science Foundation of China (60403047,60533070) and Ningbo Science Foundation(2007A610046) .

References

[1] Acton F.S., Witkins W. Numerical methods that work(2nd printing). Washington : *Mathematical Association of America*, 1990.

[2] Abramchuk V.S.,Lyashko S.I. Solution of equations with one variable. *Journal of Mathematical Sciences* 2002, 109(4):1669–1679.

[3] Aliashvili T. Counting real roots of polynomial endomorphisms. *Journal of Mathematical Sciences* 2003,118(5):5325–5346.

[4] Boehm, W. Inserting new knots into B-spline curves. *Computer-Aided Design* 1980, 12(2):199–201.

[5] Chen X.D., Su H., Yong J.H., Paul J.C., Sun J.G. A counterexample on point inversion and projection for NURBS curve. *Computer Aided Geometric Design* 2007, 24:302.

[6] Cohen E., Lyche T., Riesebfeld R. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing* 1980, 14(2):87–111.

[7] Collins G.E., Akritas A.G. Polynomial real root isolation using Descarte's rule of signs. *Proceedings of the third ACM symposium on Symbolic and algebraic computation*, 1976, pp. 272–275.

[8] Collins G.E. Polynomial minimum root separation. *Journal of Symbolic Computation* 2001, 32(5):467–473.

[9] Elber G.,Kim M.S. Geometric constraint solver using multivariate rational spline functions. *Proceedings of the sixth ACM symposium on Solid modeling and applications*, 2001, pp.1-10.

[10] Hansen E. R., Greenberg R. I. Interval Newton method. *Applied Mathematics and Computation* 1983, 12(2-3):89–98.

Table 3: Point numbers within different tolerances

Num	Fig 2(a)			Fig 2(b)			Fig 2(c)			Fig 2(d)		
	T_g	T_a	T_i	T_g	T_a	T_i	T_g	T_a	T_i	T_g	T_a	T_i
N_9	67	120	120	50	135	136	50	87	104	105	51	113
N_8	111	200	200	93	199	200	104	150	200	149	150	200
N_7	164	200	200	140	200	200	167	194	200	190	184	200
N_6	198	200	200	200	200	200	200	196	200	196	188	200
N_5	200	200	200	200	200	200	200	198	200	200	196	200

Table 4: Resulting computation time for the point inversion problem of Bézier curves

	Fig 4 (a)		Fig 4(b)		Fig 4(c)		Fig 4(d)	
	GPM	**IAM**	**GPM**	**IAM**	**GPM**	**IAM**	**GPM**	**IAM**
$T_m(ms)$	0.017	0.011	0.026	0.015	0.028	0.020	0.060	0.043
N_9	52	88	128	132	96	112	92	98
N_8	88	201	172	201	120	201	130	201
N_7	138	201	178	201	160	201	158	201
N_6	201	201	201	201	201	201	201	201

[11] Hu S.M., Wallner J. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design* 2005, 22(3):251–260.

[12] Johnson D.E., Cohen E. A framework for efficient minimum distance computations. *Proceedings of IEEE Intemational Conference on Robotics & Automation* 1998, pp. 3678–3684.

[13] Johnson D.E., Cohen E. Distance extrema for spline models using tangent cones. *Proceedings of the 2005 conference on Graphics Interface* 2005, pp.169–175.

[14] Ma Y.L., Hew W.T. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Computer Aided Geometric Design* 2003, 20(2):79–99.

[15] Mishina A.P., Proskuryakov I.V. Higher algebra: linear algebra, polynomials, general algebra (1st English edition). *Oxford, Pergamon Press Ltd.,* 1965.

[16] Mortenson M.E. *Geometric modeling*. New York : Wiley, 1985.

[17] Patrikalakis N., Maekawa T. Shape interrogation for computer aided design and manufacturing. Springer, 2001.

[18] Pegna J., Wolter F.E. Surface curve design by orthogonal projection of space curves onto free-form surfaces. *Journal of Mechanical Design, ASME Transactions* 1996, 118(1):45–52.

[19] Petkovic M.,Petkovic L.,Ilic S. The guaranteed convergence of Laguerre-like method. *Computers and Mathematics with Applications* 2003, 46(2):239–251.

[20] Phien H.N., Dejdumrong N. Efficient algorithms for Bézier curves. *Computer Aided Geometric Design* 2000, 17(3):247–250.

[21] Piegl L., Tiller W. *The NURBS book*. New York: Springer, 1995.

[22] Piegl L., Tiller W. Parametrization for surface fitting in reverse engineering. *Computer-Aided Design* 2001, 33(8):593–603.

[23] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical recipes in C: the art of scientific computing (Second Edition). New York : *Cambridge University Press*, 1992.

[24] Reif J.H. An O(n $\log^3$n) algorithm for the real root problem. *Foundations of Computer Science* 1993, pp.626–635.

[25] Rouillier F., Zimmermann P. Efficient isolation of polynomial's real roots. *Journal of Computational and Applied Mathematics* 2004, 162(1):33–50.

[26] SederbergT.W., Nishita T. Curve intersection using Bezier clipping. *Computer-Aided Design* 1990, 22(9):538-549.

[27] Ilijas Selimovic. Improved algorithms for the projection of points on NURBS curves and surfaces. *Computer Aided Geometric Design* 2006, 23:439-445.

[28] Wait R. The Numerical Solution of Algebraic Equations. Wiley, Chichester, 1979.

[29] Wu X., Fu D. New High-order convergence iteration methods without employing derivatives for solving nonlinear equations. *Computers and Mathematics with Applications*, 2001, 41(3):489–495.

[30] Yang H.P., Wang W.P. , Sun J.G. . Control point adjustment for b-spline curve approximation. *Computer-Aided Design* 2004, 36(7):639–652.

[31] Yang L., Zhang J.Z., Hou X.L. Non-linear equation system and automated theory proving (in chinese). Shanghai : *Shanghai press of science, technique and edution*, 1996.

[32] Ye Y. Combining binary search and Newton's method to compute real roots for a class of real functions. *Journal of Complexity* 1994, 10(3):271–280.

[33] Zhou, J.M., Sherbrooke, E.C., Patrikalakis, N. Computation of stationary points of distance functions. *Engineering with Computers*, 1993, 9:231–246.