Bull. Math. Soc. Sci. Math. Roumanie
Tome 54(102) No. 2, 2011, 185–191

Comparison of Different Bounding Sets of Polynomial Functions Defined in a Given Domain

by
Gun Srijuntongsiri

Abstract

Subdivision and subdivision/iterative hybrid methods for solving a system of polynomial equations require the ability to compute the bounding set of a polynomial function for a given domain. There are different methods to compute the bounding sets and each method produces different bounding sets. Since the sizes of these bounding sets directly affect the running time of subdivision and hybrid methods, selecting the appropriate method for computing bounding sets is important. This article investigates three methods for computing bounding sets and compares them experimentally to see which one produces the smallest sets. The results show that reparametrization followed by constructing convex hulls of the control points of the resulting polynomials in Bernstein basis almost always produces smaller bounding sets than the other two methods. Therefore, using this method to compute bounding sets would result in more efficient subdivision or hybrid methods for solving polynomial systems.

Key Words: Bounding sets; Subdivision; Polynomial equations; Intersection problems.
2010 Mathematics Subject Classification: Primary 68U07; Secondary 65D17.

1 Introduction

Given a polynomial function $f : \mathbb{R}^n \to \mathbb{R}^N$ and a domain D, we are interested in an efficiently-computable convex bounding set V containing $f(x)$ for all $x \in D$. The bounding set V is often used in subdivision methods for finding all zeros of f in a given domain. Therefore, it should be much more efficient to compute V than solving f itself. A common special case of finding all zeros of f in a given domain is locating all intersections between two or more geometric objects [20, 16, 10, 14]. The intersection problems have many applications in computer-aided geometric design, computational geometry, geometric modeling and design, robotics, and

other fields [9, 13, 6, 15]. There are also hybrid methods that combine subdivision and iterative methods proposed for intersection problems [19, 11, 17].

The key idea of subdivision methods is to compute a bounding set that is easy to check if it contains the origin. If it does not, then there are no zeros of f in the domain. If it does, the methods subdivide the domain and check if the bounding set of each subdomains contains the origin recursively. When a subdomain is small enough, it is taken as an approximation of a zero. Hybrid methods operate similarly but instead invoke an iterative method to locate a zero once it is certain that the iterative method converges. We note that hybrid methods are applicable only to some combinations of N and n.

For both pure subdivision and hybrid methods, their efficiencies depend largely on the size of the computed bounding sets. If bounding sets are much larger than the bounded ranges of a function for a given domain, the bounding sets may contain the origin although there is no zero in the domain. Consequently, the methods perform more subdivisions than they should need. Since there are more than one way to define and compute bounding sets, selecting the appropriate definition becomes vital to the efficiency of the subdivision and hybrid methods.

This article considers three methods to compute bounding sets. First is by applying interval arithmetic to the polynomial represented in Bernstein basis. Second is by rewriting the polynomial in monomial basis and applying interval arithmetic to the rewritten expression. Third is by reparametrization followed by computation of a convex hull. The details of these three methods are described in Section 2, 3 and 4. We compare the sizes of the bounding sets computed by the three methods by experimenting on various polynomials to see which method produces the smallest bounding sets.

We now define the problem under consideration more precisely by specifying a representation for the polynomial system. Let $Z_{i,m}(t)$ denote the *Bernstein polynomials*

$$Z_{i,m}(t) = \frac{m!}{i!(m-i)!}(1-t)^{m-i}t^i.$$

Consider a system of N polynomial equations in n variables

$$f(x) \equiv \sum_{i_1=0}^{m_1} \cdots \sum_{i_{n+1}=0}^{m_n} b_{i_1,\cdots,i_n} Z_{i_1,m_1}(x_1) \cdots Z_{i_n,m_n}(x_n) = 0, \tag{1}$$

where $b_{i_1,\cdots,i_n} \in \mathbb{R}^N$ $(i_j = 0, 1, \ldots, m_j)$ denote the coefficients, also known as the *control points*. Note that f is a function that maps $\mathbb{R}^n$ to $\mathbb{R}^N$. For a given hyperrectangle $D = [l_1, h_1] \times [l_2, h_2] \times \cdots \times [l_n, h_n]$, where $l_j \leq h_j$, we seek a bounding set V satisfying

$$V \supseteq \{f(x) : x \in D\}.$$

We choose to represent the polynomial system in the Bernstein basis because it is known to have better numerical stability than the monomial basis [3, 5, 4] and is widely used in practice due to its geometric properties.

As mentioned above, many classes of problems of finding intersections between two parametric geometric objects can reduce to solving (1). For examples, line/surface intersection reduces to the case where $N = n = 2$, curve/curve intersection corresponds with $N = 3, n = 2$, curve/surface intersection corresponds with $N = n = 3$, and surface/surface intersection corresponds with $N = 3, n = 4$.

We proceed with the description of the three methods for computing bounding sets that are considered in this article.

2 Computing Bounding Sets by Interval Arithmetic on Bernstein Basis Representation

An *interval* $[a, b]$ is a set of real numbers defined as follows [12]:

$$[a, b] = \{x : a \leq x \leq b\}.$$

The *width* of an interval $[a, b]$ is $b - a$. Interval arithmetic operations are defined by

$$[a, b] \circ [c, d] = \{x \circ y : x \in [a, b] \wedge y \in [c, d]\}, \tag{2}$$

where $\circ$ represents an arithmetic operation. Three basic interval arithmetic operations—addition, subtraction, and multiplication—can be rewritten equivalently as

$$\begin{aligned} [a, b] + [c, d] &= [a + c, b + d], \\ [a, b] - [c, d] &= [a - d, b - c], \\ [a, b] \cdot [c, d] &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)], \end{aligned}$$

which give simpler ways to perform interval arithmetic operations than (2). To compute a bounding set V of f for a given domain $D = [l_1, h_1] \times \cdots \times [l_n, h_n]$, replace each occurrence of x_i in (1) with interval $[l_i, h_i]$ and use interval arithmetic operations instead in the evaluation of (1) (usually by the *de Casteljau algorithm* [9, 15]). The correctness of interval arithmetic [8] ensures that the resulting interval is indeed a bounding set of f for D.

3 Computing Bounding Sets by Interval Arithmetic on Monomial Basis Representation

Alternatively, we may rewrite (1) in the monomial basis

$$f(x) = \sum_{i_1=0}^{m_1} \cdots \sum_{i_{n+1}=0}^{m_n} c_{i_1,\cdots,i_n} x_1^{i_1} \cdots x_n^{i_n} = 0, \tag{3}$$

where $c_{i_1,\cdots,i_n} \in \mathbb{R}^N$ $(i_j = 0, 1, \ldots, m_j)$, and perform interval arithmetic on the rewritten expression (3). The appeal for the monomial basis is a direct interval arithmetic evaluation of a polynomial represented in the monomial basis is known

to yield a bounding set with an error linear in the width of the interval of the variables [1]. On the other hand, we do not know of such an error bound for polynomials represented in the Bernstein basis. In fact, it is possible to rewrite a polynomial further to achieve an error quadratic in the width of the interval [1]. We do not investigate the quadratic-error method in this article.

Finally, recall that Bernstein representation of polynomials is known to be numerically more stable than the monomial representation hence the conversion to the monomial basis may not be desirable from numerical viewpoint.

4 Computing Bounding Sets by Reparametrization and Computing Convex Hulls

Another method to compute bounding sets follows the well-known property of the Bernstein representation of a polynomial that any point $f(x)$, where $x \in [0,1]^n$, lies inside the convex hull of the control points of f [2]. For a given hyperrectangle D, let $\hat{f}_D$ be the Bernstein polynomial that reparametrizes with $[0,1]^n$ the function defined by f over D. In other words, $\hat{f}_D(q) \equiv f(\lambda(q))$, where $\lambda(q)$ is a composition of a dilation and translation (uniquely determined) such that $\lambda : [0,1]^n \to D$ is bijective. Then, the convex hull of the control points of $\hat{f}_D$ is the bounding set of f over D.

There are at least two efficient ways to perform the reparametrization. One method is by applying the de Casteljau algorithm twice; once to reparametrize the right endpoints of the domain with 1 and another to reparametrize the left endpoints with 0. Alternatively, the reparametrization can be efficiently performed by a generalization of Horner's rule (An example of this reparametrization method for bivariate Bernstein polynomials is in [18]).

5 Comparison of Bounding Sets Computed by Different Methods

In this section, we show the experimental results comparing the sizes of bounding sets computed by the three methods: (i) by applying interval arithmetic directly to (1), (ii) by rewriting (1) in monomial basis and then applying interval arithmetic to the rewritten expression (3), and (iii) by reparametrization to obtain the control points of $\hat{f}_D$ and then computing the convex hull of these control points. For Method (i), (1) is evaluated by the de Casteljau algorithm. For Method (ii),(3) is evaluated using Horner's rule [7].

We experiment on the case where $N = n = 1$ with polynomials generated by different methods. Table 1 shows the results of the experiments for the cases where the widths of D are between .4 and 1. Table 2 shows the results for the cases where the widths of D are smaller than .1. The polynomials are generated as follows. The "low-deg. rand." polynomials have normally distributed random control points with degrees between 1 and 5, inclusive. The "high-deg. rand." ones have normally distributed random control points with degrees between 10 and 20, inclusive. The "low-period sine" ones are polynomial interpolations of

Table 1: The numbers of test polynomials that each method yields the smallest bounding sets when the widths of D are between .4 and 1. One thousand of each polynomial types are generated and tested.

Polynomial type	Num. of poly. that interval arithmetic in Bernstein basis method yields smallest V	Num. of poly. that interval arithmetic in monomial basis method yields smallest V	Num. of poly. that reparam. method yields smallest V
Low-deg. rand.	8	54	938
High-deg. rand.	0	0	1000
Low-period sine	0	0	1000
High-period sine	0	0	1000
Low-var. rand.	11	111	878

$\sin(ax + b)$ at five to sixteen evenly-spaced points between -1 and 1, inclusive, where a and b are normally distributed random numbers. The "high-period sine" ones are generated in the same way as the "low-period sine" ones but with the function $\sin(20ax + b)$ instead. The "low-var. rand." ones have degrees between 1 and 3, inclusive, with control points normally distributed with very small variance. The experimental results show that method (iii) consistently produces the smallest bounding sets among the three methods for all five types of polynomials and for both small and large D's. Another observation is that Method (iii) performs even better than the other twos when the polynomials have many "wiggles" such as those arising from interpolation of the sine function.

6 Conclusion and Future Directions

We compare three methods for computing bounding sets of polynomial functions over a given domain. It is shown experimentally that reparametrization with $[0, 1]^n$ followed by constructing convex hulls of the resulting control points usually yields smaller bounding sets than the other two methods. We conclude the article by posing the following related questions that are not considered here:

- **Formal proofs.** Can it be shown analytically that the convex hull methods always produce the smallest bounding sets among the three methods?
- **Better methods to compute bounding sets.** Are there other methods to compute bounding sets that are not considered here but are more efficient and/or produce smaller bounding sets? For example, Alefeld and Rokne show that it is possible to rewrite polynomials in such a way that interval arithmetic yields quadratic error [1]. Does this method produce smaller bounding sets than the other methods?

Table 2: The numbers of test polynomials that each method yields the smallest bounding sets when the widths of D are smaller than .1. One thousand of each polynomial types are generated and tested.

Polynomial type	Num. of poly. that interval arithmetic in Bernstein basis method yields smallest V	Num. of poly. that interval arithmetic in monomial basis method yields smallest V	Num. of poly. that reparam. method yields smallest V
Low-deg. rand.	19	73	908
High-deg. rand.	0	0	1000
Low-period sine	0	0	1000
High-period sine	0	0	1000
Low-var. rand.	32	121	847

References

[1] G. Alefeld and J. G. Rokne. On the evaluation of rational functions in interval arithmetic. *SIAM Journal on Numerical Analysis*, 18(5):862–870, 1981.

[2] G. Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, 5 edition, 2002.

[3] R. T. Farouki. On the stability of transformations between power and Bernstein polynomial forms. *Computer Aided Geometric Design*, 8(1):29–36, 1991.

[4] R. T. Farouki and T. N. T. Goodman. On the optimal stability of the bernstein basis. *Mathematics of Computation*, 65(216):1553–1566, October 1996.

[5] R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Berstein form. *Computer Aided Geometric Design*, 4(3):191–216, 1987.

[6] J. Gallier. *Curves and Surfaces in Geometric Modeling: Theory and Algorithms*. Morgan Kaufmann, 1999.

[7] M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, 2005.

[8] T. Hickey, Q. Ju, and M. H. Van Emden. Interval arithmetic: From principles to implementation. *J. ACM*, 48(5):1038–1068, 2001.

[9] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. A. K. Peters, Wellesley, MA, 1993. Translated by L. L. Schumaker.

[10] E. G. Houghton, R. F. Emnett, J. D. Factor, and C. L. Sabharwal. Implementation of a divide-and-conquer method for intersection of parametric surfaces. *Computer Aided Geometric Design*, 2(1–3):173–183, 1985.

[11] P. Koparkar. Surface intersection by switching from recursive subdivision to iterative refinement. *The Visual Computer*, 8:47–63, 1991.

[12] R. E. Moore. *Interval Analysis*. Prentice Hall, 1966.

[13] M. E. Mortenson. *Geometric Modeling*. John Wiley and Sons, New York, 1985.

[14] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, 1993.

[15] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag Berlin Heidelberg, Germany, 2002.

[16] Steven M. Rubin and Turner Whitted. A 3-dimensional representation for fast rendering of complex scenes. *SIGGRAPH Comput. Graph.*, 14(3):110–116, 1980.

[17] G. Srijuntongsiri and S. A. Vavasis. A condition number analysis of a line-surface intersection algorithm. *SIAM Journal on Scientific Computing*, 30(2):1064–1081, 2007.

[18] G. Srijuntongsiri and S. A. Vavasis. Properties of polynomial bases used in a line-surface intersection algorithm. In *Parallel Processing and Applied Mathematics*, 2009.

[19] D. L. Toth. On ray tracing parametric surfaces. *SIGGRAPH Comput. Graph.*, 19(3):171–179, 1985.

[20] T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

Received: 12.07.2010,
Accepted: 02.01.2011.

Sirindhorn International Institute of Technology,
Thammasat University, Thailand
E-mail: gun@siit.tu.ac.th