Computer Aided Geometric Design 3 (1986) 149–154
North-Holland

Efficient evaluation of multivariate polynomials

Larry L. SCHUMAKER

Center for Approximation Theory, Texas A & M University, College Station, TX 77843, U.S.A.

Wolfgang VOLK

SIETEC Siemens – Systemtechnik, D-1000 Berlin 13, Fed. Rep. Germany

Received 16 December 1985
Revised 3 March 1986

Abstract. In CAGD applications, the usual way to store a polynomial defined over a triangle is in the Bernstein–Bézier form, and the accepted way to evaluate it at a given point in the triangle is by the de Casteljau algorithm. In this paper we present an algorithm for evaluating Bernstein–Bézier type polynomials which is significantly faster than de Casteljau. This suggests that a modified form of Bernstein–Bézier may be preferable for CAGD applications (as well as in applications of piecewise polynomials in data fitting and numerical solution of boundary-value problems).

Keywords. Bernstein–Bézier methods, surfaces, polynomial evaluation.

1. Introduction

Suppose that T is a triangle in the plane, and for each point P in T, let (r, s, t) be the Barycentric coordinates of P. Then it is well-known [Boehm et al. '84], that any polynomial of total degree d defined on T can be written in the form

$$p(r, s, t) = \sum_{i=0}^{d} \sum_{j=0}^{i} b_{d-i,i-j,j} B^d_{d-i,i-j,j}(r, s, t), \tag{1.1}$$

where, in general

$$B^d_{i,j,k}(r, s, t) = \frac{d!}{i!j!k!} r^i s^j t^k, \quad \text{all } i+j+k=d, \tag{1.2}$$

are the usual Bernstein-polynomials. The expansion (1.1) is called the *Bernstein–Bézier* (BB) representation of p.

The following *algorithm of de Casteljau* [de Casteljau '59] can be used to evaluate p at a given (r, s, t): Starting with $b^0_{i,j,k} = b_{i,j,k}$ for all $i+j+k=d$,

$$\begin{aligned}&\text{do } k = 1, d\\ &\quad \text{do } i = 0, d-k\\ &\quad\quad \text{do } j = 0, i\\ &\quad\quad\quad b^k_{d-i-k,i-j,j} = r * b^{k-1}_{d-i-k+1,i-j,j} + s * b^{k-1}_{d-i-k,i-j+1,j} + t * b^{k-1}_{d-i-k,i-j,j+1}.\end{aligned} \tag{1.3}$$

Then $p(r, s, t) = b^d_{0,0,0}$. This algorithm requires $d(d+1)(d+2)/2$ multiplications.

0167-8396/86/$3.50 © 1986, Elsevier Science Publishers B.V. (North-Holland)

2. A modified Bernstein–Bézier representation

Given a polynomial as in (1.1), it is clear that it can be rewritten in the form

$$p(r, s, t) = \sum_{i=0}^{d} \sum_{j=0}^{i} c_{d-i,i-j,j} r^{d-i} s^{i-j} t^{j}, \tag{2.1}$$

where

$$c_{d-i,i-j,j} = \frac{d!}{(d-i)!(i-j)!j!} b_{d-i,i-j,j}, \quad j = 0, \ldots, i;\ i = 0, \ldots, d. \tag{2.2}$$

We shall refer to (2.1) as the *Modified Bernstein–Bézier* (MBB) representation of p.

Our aim in this section is to show that the MBB representation can be evaluated very efficiently. The key to our evaluation algorithm is the observation that p as in (2.1) can be written in nested form. For example, with $d = 2$ and the association with single subscripted c's shown in Fig. 1, we may write

$$p(r, s, t) = r * r * \left[sr * (sr * c_4 + tr * c_5 + c_2) + tr * (tr * c_6 + c_3) + c_1\right], \tag{2.3}$$

$$p(r, s, t) = s * s * \left[rs * (rs * c_1 + c_2 + ts * c_3) + c_4 + ts * (c_5 + ts * c_6)\right], \tag{2.4}$$

$$p(r, s, t) = t * t * \left[rt * (rt * c_1 + st * c_2 + c_3) + st * (st * c_4 + c_5) + c_6\right], \tag{2.5}$$

where $sr = s/r$, $tr = t/r$, $rs = r/s$, $ts = t/s$, $rt = r/t$, and $st = s/t$. In order to prevent division by zero or by near-zero values, we suggest that (2.3), (2.4), or (2.5) be used depending on which of r, s, or t is biggest. The three associated regions are shown in Fig. 2.

Algorithm 2.1. Let p be a polynomial of total degree d written in the MBB form (2.1). Suppose r, s, t are the Barycentric coordinates of a point in Region 3.

$rt = r/t, \qquad st = s/t$

$A = c_{d,0,0}$

do $i = 1, d$

  $B = c_{d-i,i,0}$

  do $j = 1, i$

    $B = B * st + c_{d-i,i-j,j}$

  $A = A * rt + B$

$p(r, s, t) = A * t^d.$

Discussion. This algorithm requires $(d^2 + 5d)/2$ multiplications and two divisions, counting $d - 1$ multiplications to calculate t^d. In general, t^d could be calculated in a few less multiplications. □

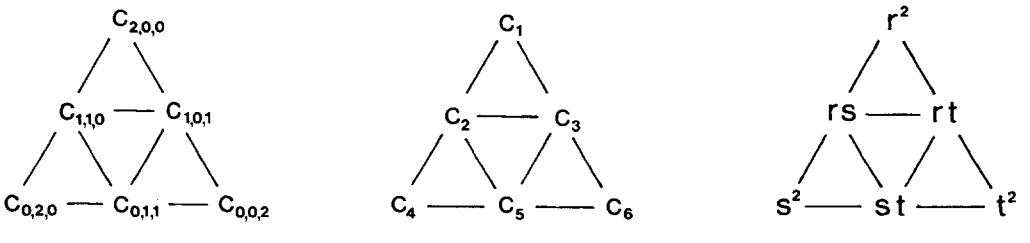

Fig. 1. Coefficients and powers of r, s, and t.

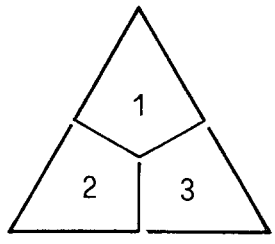

Fig. 2. The regions of choice for Algorithm 2.1.
1: $r \geq s$ and $r \geq t$,
2: $s > r$ and $s \geq t$,
3: $t > r$ and $t > s$.

There are obvious analogs of Algorithm 2.1 for the cases when (r, s, t) is in one of the Regions 1 or 2 defined above. They require the same number of operations as Algorithm 2.1. This use of different algorithms in the three subsets of T guarantees the stability of the method.

We shall compare Algorithm 2.1 with the Casteljau algorithm in Section 4 below, but for the sake of discussion, we first develop a similar algorithm for polynomials written in Taylor form.

3. The Taylor form

Let T be a triangle, and let p be a polynomial of total degree d defined on T. Let $u = x - x_1$ and $v = y - y_1$, where (x_1, y_1) are the orthogonal coordinates of an arbitrary point (e.g. one of the vertices of T). Then we may write p in the Taylor form

$$p(u, v) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} a_{i,j} u^i v^j. \tag{3.1}$$

Writing p in nested form, it is easy to see that $p(u, v)$ can be calculated by the following algorithm.

Algorithm 3.1. (Evaluate p as in (3.1).)

$p = a_{0,d}$
do $i = 1, d$
  $A = a_{i,d-i}$
  do $j = 1, i$
    $A = A * u + a_{i-j,d-i}$
$p = p * v + A.$

Discussion. This algorithm requires $(d^2 + 3d)/2$ multiplications. □

4. Comparison of the evaluation methods

We refer to Algorithm 2.1 for the evaluation of a MBB polynomial as the *VS algorithm*. If one is given a BB polynomial, then as is clear from (2.2), the coefficients of its MBB form can be computed from the original coefficients by multiplication by appropriate factors. As these factors are easily precomputed (and three of them have value 1), this conversion can be accomplished in $(d^2 + 3d - 4)/2$ multiplications.

We now compare the various possibilities. The combination of Algorithm 2.1 with the conversion from BB to MBB form will be referred to as the *VSC Algorithm*. It can be used to

Table 1
A comparison of evaluation algorithms

d	2	3	4	5	6	7	8	9
dCas	12	30	60	105	168	256	360	495
VSC	12	21	32	45	60	77	96	117
VS	9	14	20	27	35	44	54	65
Tay	5	9	14	20	27	35	44	54

evaluate polynomials in the Bernstein–Bézier form. We now have the following counts on the number of multiplications and divisions (MD's) for each of our evaluation algorithms:

$(d^3 + 3d^2 + 2d)/2$ MD's for de Casteljau,
$(2d^2 + 8d)/2$ MD's for VSC,
$(d^2 + 5d + 4)/2$ MD's for VS,
$(d^2 + 3d)/2$ MD's for Taylor.

It is clear that de Casteljau grows cubically with d, while the others are all of quadratic growth. Table 1 shows the effect for $2 \leq d \leq 9$.

Table 1 shows that evaluation of polynomials in MBB form is considerably more efficient than evaluation of polynomials in BB form using de Casteljau. Given a polynomial in BB form, it is also clear that (except for $d = 2$), the VSC algorithm is also significantly faster than de Casteljau. In this connection, we should mention that in practice one often has to evaluate the same polynomial several times, and then only one conversion of the BB form is required after which VS can be applied at as many points as desired. Finally, the Taylor form is seen to be the most efficient of all, although it has only a marginal advantage over VS. Since the cost of conversion from either the BB or MBB form to the Taylor form is substantial, the Taylor evaluation Algorithm 3.1 is not an efficient way to evaluate polynomials in BB or MBB form.

We may now form the following conclusion. For applications where a Bernstein–Bézier type expansion is desired (for example in CAGD where this form of representing the polynomial provides a convenient handle for controlling smoothness between patches), we recommend the use of the MBB form together with the VS algorithm.

5. Polynomials on tetrahedra

There has been recent interest in the use of piecewise polynomials defined on tetrahedral tesselations. In this section we briefly describe how the above results can be generalized to this case. Let T now be a tetrahedron in R^3, and for each point P in T, let (r, s, t, u) be its corresponding Barycentric coordinate vector. Then given any polynomial of total degree d defined on T, it can be written in the modified Bernstein–Bézier form

$$p(r, s, t, u) = \sum_{i=0}^{d} \sum_{j=0}^{i} \sum_{k=0}^{j} c_{d-i,i-j,j-k,k} r^{d-i} s^{i-j} t^{j-k} u^k. \tag{5.1}$$

There are a total of $(d+1)(d+2)(d+3)/6$ coefficients. Fig. 3 illustrates the Bézier net of coefficients in the case of $d = 2$.

We now discuss the problem of evaluating (5.1) efficiently. As in the two-dimensional case, we use different algorithms depending on where the point (r, s, t, u) lies in the tetrahedron T.

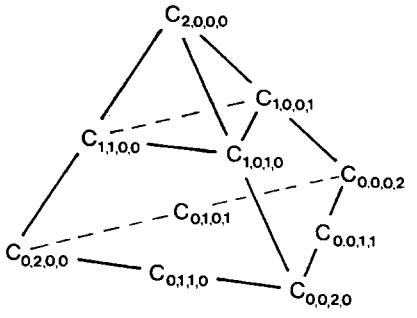

Fig. 3. The Bézier net of coefficients of (5.1).

We define four subsets of $\dot{T}$ by

Region 1: $r \geq s,\ r \geq t,\ r \geq u,$
Region 2: $s > r,\ s \geq t,\ s \geq u,$
Region 3: $t > r,\ t > s,\ t \geq u,$
Region 4: $u > r,\ u > s,\ u > t.$

Algorithm 5.1. Let p be as in (5.1), and let (r, s, t, u) be a point in Region 4 of the tetrahedron T.

$ru = r/u, \quad su = s/u, \quad tu = t/u$
$A = c_{d,0,0,0}$
do $i = 1, d$
  $B = c_{d-i,i,0,0}$
  do $j = 1, i$
    $C = c_{d-i,i-j,j,0}$
    do $k = 1, j$
      $C = C * tu + c_{d-i,i-j,j-k,k}$
    $B = B * su + C$
  $A = A * ru + B$
$p(r, s, t, u) = A * u^d.$

Discussion. This operation requires a total of $(d^3 + 6d^2 + 17d)/6$ multiplications (counting $d - 1$ for the calculation of u^d). It also requires 3 divisions. (The number u^d could be calculated in a few less multiplications, in general.) □

6. Remarks

(1) It is possible to construct similar algorithms based on nesting for the evaluation of MBB polynomials which do not make use of quotients, but we have not been able to make these alternatives as efficient as the VS algorithm presented here. In any case, we believe there is no reason to avoid the use of quotients as long as one makes the choice of which quotients to use according to their relative size.

(2) The de Casteljau algorithm operates with convex combinations. This is of course extremely stable when all the coefficients are nonnegative, which, in general, they will not be. This convexity property does insure that all intermediate quantities which are computed remain bounded by the norm $\|b\|$ of the original coefficients. Since all quotients which appear in the VS algorithm are less than 1 (as are r, s, t), the intermediate quantities which appear in VS are bounded by $nc\|c\|$, where $nc = (d + 1)(d + 2)/2$ is the number of coefficients.

(3) We have not been able to find much about the complexity of evaluating multivariate polynomials. Algorithm 3.1 for evaluating the Taylor form requires precisely $nc - 1$ multiplications (and a like number of additions). This is one less than the number of coefficients, which is exactly the situation for the well-known univariate Horner scheme. Since the VS algorithm requires only a few more operations (to calculate 2 quotients and a power of the form r^d, we believe it is close to optimal).

(4) Our focus in this paper has been on the evaluation of a multivariate polynomial at a single point. When the same polynomial has to be evaluated at many points lying on a straight line (such as would be needed for raster display), additional economies can be realized [cf. Schumaker '86] for the details of raster evaluation).

(5) The de Casteljau Algorithm (1.3) not only produces the value of a BB polynomial p at a point (r, s, t), but as an intermediate calculation also produces the gradient vector $(D_r p, D_s p)$ at the same point [cf. Boehm et al. '84]. This is of particular interest in some rendering algorithms where either the tangent plane or a normal vector to the surface is required. If p is written in MBB form the value of the gradient is not automatically produced in the course of the VS algorithm. On the other hand, it is cheap and easy to get the coefficients of both $D_r p$ and $D_s p$ from those of p. Since these are both polynomials of degree $d - 1$, the VS algorithm requires $d^2 + 3d$ MD's. Thus the value of p and the gradient can all be gotten at the cost of $(3d^2 + 11d + 4)/2$ MD's. For $d \geq 4$ this is cheaper than de Casteljau.

References

Boehm, W., Farin, G. and Kahmann, J. (1984), A survey of curve and surface methods in CAGD, Computer Aided Geometric Design 1, 1–60.

de Casteljau, F., Courbes et surfaces à pôles, André Citröen Automobiles, Paris, 1963.

Schumaker, L.L., Numerical aspects of spaces of piecewise polynomials on triangulations, in: J. Mason, ed., Proceedings of Shrivenham Conference on Approximation Theory, 1986.