Abstract

In computer aided geometric design a polynomial is usually represented in Bernstein form. The de Casteljau algorithm is the most well-known algorithm for evaluating a polynomial in this form. Evaluation via the de Casteljau algorithm has relative forward error proportional to the condition number of evaluation. However, for a particular family of polynomials, a curious phenomenon occurs: the observed error is much smaller than the expected error bound. We examine this family and prove a much stronger error bound than the one that applies to the general case. Then we provide a few examples to demonstrate the difference in rounding.

Keywords: Polynomial evaluation, Floating-point arithmetic, Bernstein polynomial, Round-off error, Condition number

Contents

1 Introduction

In computer aided geometric design, polynomials are usually expressed in Bernstein form. Polynomials in this form are usually evaluated by the de Casteljau algorithm. This algorithm has a round-off error bound which grows only linearly with degree, even though the number of arithmetic operations grows quadratically. The Bernstein basis is optimally suited ([FR87, DP15, MP05]) for polynomial evaluation. Nevertheless the de Casteljau algorithm returns results arbitrarily less accurate than the working precision $\mathbf{u}$ when evaluating $p(s)$ is ill-conditioned. The relative accuracy of the computed evaluation with the de Casteljau algorithm (`DeCasteljau`) satisfies ([MP99]) the following a priori bound:

$$\frac{|p(s) - \texttt{DeCasteljau}(p, s)|}{|p(s)|} \leq \operatorname{cond}(p, s) \times \mathcal{O}(\mathbf{u}). \tag{1.1}$$

For example, consider $u(s) = (1 - 4s)^5, v(s) = (1 - 5s)^5$ and $w(s) = (1 - 6s)^5$. For points of the form

$$s_u = \frac{1}{4} + \frac{6}{16N}, \quad s_v = \frac{1}{5} + \frac{8}{25N}, \quad s_w = \frac{1}{6} + \frac{10}{36N} \tag{1.2}$$

which are near the multiple roots, the condition numbers of evaluation are:

$$\operatorname{cond}(u, s_u) = \operatorname{cond}(v, s_v) = \operatorname{cond}(w, s_w) = |N|^5 + \mathcal{O}\left(N^4\right). \tag{1.3}$$

As we can see in Figure 1.1, one of these is not like the others. Evaluation of both $u(s)$ and $w(s)$ produces relative forward error very close to the a prior bound[1]. However, the observed relative error when evaluating $v(s)$ is significantly lower than expected.

[1] There are actually three different error bounds, but the $|N|^5$ term dominates so much that they are not visually discernible.

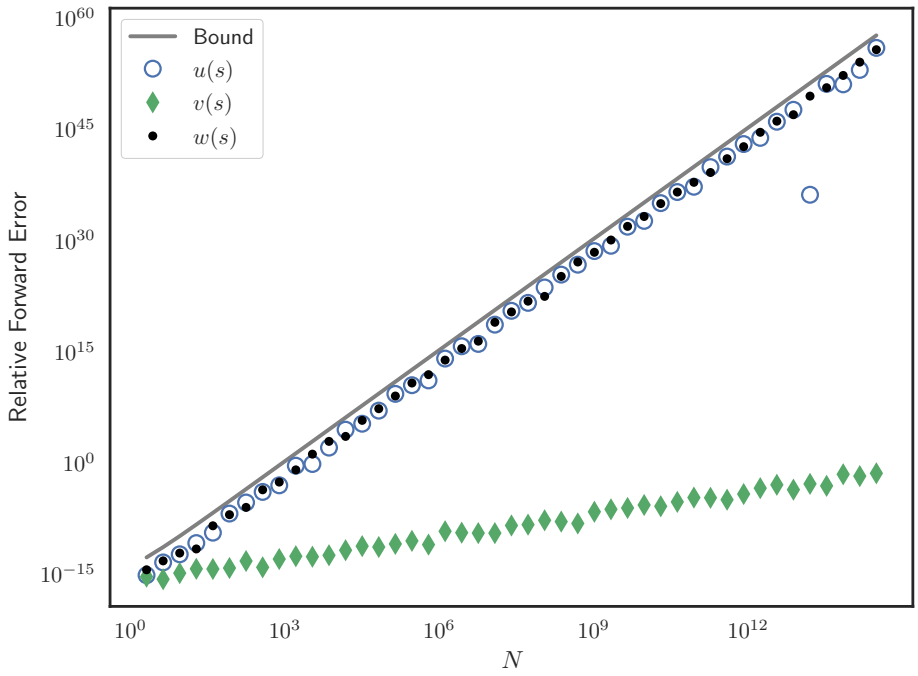

Figure 1.1: Comparing relative forward error to a priori bound for $u(s) = (1-4s)^5, v(s) = (1-5s)^5$ and $w(s) = (1-6s)^5$

As we'll explore in Section 3, $v(s)$ belongs to a family of polynomials that the de Casteljau method can evaluate with a significantly higher level of accuracy than expected. Notice that

$$v(s) = (1-5s)^5 = [(1-s)-4s]^5 = B_{0,5}(s) - 4B_{1,5}(s) + 16B_{2,5}(s) + \cdots; \tag{1.4}$$

in particular, the Bernstein coefficients are powers of 2 (up to sign). The aforementioned family of polynomials contains any of the form

$$b_0\left[(1-s)-2^t s\right]^n = b_0 B_{0,n}(s) - \left(b_0 2^t\right) B_{1,n}(s) + \left(b_0 2^{2t}\right) B_{2,n}(s) + \cdots. \tag{1.5}$$

Polynomials in this family have coefficients that can be represented exactly (i.e. with no round-off).

The paper is organized as follows. Section 2 establishes notation for error analysis with floating point operations and reviews the de Casteljau algorithm. In Section 3, the lowered error bound is proved for polynomials in the special family and numerical experiments compare observed relative error to the newly improved bound. Finally, in Section 4 we comment on the impact that this lowered bound makes on comparisons between the de Casteljau algorithm and the VS algorithm.

2 Basic notation and results

2.1 Floating Point and Forward Error Analysis

We assume all floating point operations obey

$$a \star b = \mathrm{fl}\,(a \circ b) = (a \circ b)(1+\delta_1) = (a \circ b)/(1+\delta_2) \tag{2.1}$$

where $\star \in \{\oplus, \ominus, \otimes, \oslash\}$, $\circ \in \{+, -, \times, \div\}$ and $|\delta_1|, |\delta_2| \leq \mathbf{u}$. The symbol $\mathbf{u}$ is the unit round-off and $\star$ is a floating point operation, e.g. $a \oplus b = \mathrm{fl}\,(a+b)$. (For IEEE-754 floating point double precision, $\mathbf{u} = 2^{-53}$.) We denote the computed result of $\alpha \in \mathbf{R}$ in floating point arithmetic by $\widehat{\alpha}$ or $\mathrm{fl}\,(\alpha)$ and use $\mathbf{F}$ as the set of all floating point numbers (see [Hig02] for more details). Following [Hig02], we will use the following classic properties in error analysis.

1. If $\delta_i \leq \mathbf{u}$, $\rho_i = \pm 1$, then $\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n$,
2. $|\theta_n| \leq \gamma_n \coloneqq n\mathbf{u}/(1-n\mathbf{u})$,
3. $(1+\theta_k)(1+\theta_j) = 1 + \theta_{k+j}$,
4. $\gamma_k + \gamma_j + \gamma_k\gamma_j \leq \gamma_{k+j} \iff (1+\gamma_k)(1+\gamma_j) \leq 1 + \gamma_{k+j}$,
5. $(1+\mathbf{u})^j \leq 1/(1-j\mathbf{u}) \iff (1+\mathbf{u})^j - 1 \leq \gamma_j$.

Theorem 2.1. In the absence of overflow or underflow, for $a, b, -2^t \in \mathbf{F}$

$$\left(-2^t a\right) \otimes b = -2^t \left(a \otimes b\right) \tag{2.2}$$

$$\left(-2^t a\right) \oplus \left(-2^t b\right) = -2^t \left(a \oplus b\right). \tag{2.3}$$

2.2 Bernstein Basis and de Casteljau Algorithm

A polynomial written in the Bernstein basis is of the form

$$p(s) = \sum_{j=0}^{n} b_j B_{j,n}(s) \tag{2.4}$$

where $B_{j,n}(s) = \binom{n}{j}(1-s)^{n-j}s^j$. When $s \in [0,1]$, the Bernstein basis functions are non-negative. We refer to $\widetilde{b}_j \coloneqq \binom{n}{j} b_j$ as the *scaled Bernstein coefficients*. The condition number of evaluation for $p(s)$ is

$$\operatorname{cond}(p, s) = \frac{\widetilde{p}(s)}{|p(s)|} \tag{2.5}$$

where $\widetilde{p}(s) \coloneqq \sum_{j=0}^{n} |b_j| B_{j,n}(s)$.

Algorithm 2.1 *de Casteljau algorithm for polynomial evaluation.*

function `result` = `DeCasteljau`(b, s)
  $n =$ `length`$(b) - 1$
  $\widehat{r} = 1 \ominus s$

  for $j = 0, \ldots, n$ **do**
    $\widehat{b}_j^{(n)} = b_j$
  end for

  for $k = n-1, \ldots, 0$ **do**
    for $j = 0, \ldots, k$ **do**
      $\widehat{b}_j^{(k)} = \left(\widehat{r} \otimes \widehat{b}_j^{(k+1)}\right) \oplus \left(s \otimes \widehat{b}_{j+1}^{(k+1)}\right)$
    end for
  end for

  `result` $= \widehat{b}_0^{(0)}$
end function

Theorem 2.2. The de Casteljau algorithm (Algorithm 2.1) satisfies

$$|p(s) - \texttt{DeCasteljau}(p, s)| \leq \gamma_{3n}\widetilde{p}(s). \tag{2.6}$$

Proof. See Appendix A. ■

3 Improved Bound

We seek to analyze our family of polynomials of the form:

$$p(s) = b_0 \left[(1-s) - 2^t s\right]^n. \tag{3.1}$$

When written in the Bernstein basis, $p(s)$ has coefficients that satisfy

$$b_{j+1} = -2^t b_j. \tag{3.2}$$

In a finite precision binary arithmetic, (3.2) will hold exactly (i.e. with no round-off) until overflow or underflow makes it impossible to represent b_j in the given arithmetic (the mantissa will always be the same but the sign and exponent will change). This useful property remains true for the intermediate terms computed by the de Casteljau method:

$$b_{j+1}^{(k)} = (1-s)b_{j+1}^{(k+1)} + s b_{j+2}^{(k+1)} = -2^t \left[(1-s)b_j^{(k+1)} + s b_{j+1}^{(k+1)}\right] = -2^t b_j^{(k)}. \tag{3.3}$$

Remarkably, this also holds true for the **computed** values: $\widehat{b}_{j+1}^{(k)} = -2^t \widehat{b}_j^{(k)}$. Following Theorem 2.1 we have

$$\begin{aligned}
\widehat{b}_{j+1}^{(k)} &= \left[\widehat{r} \otimes \widehat{b}_{j+1}^{(k+1)}\right] \oplus \left[s \otimes \widehat{b}_{j+2}^{(k+1)}\right] && (3.4)\\
&= \left[-2^t \left(\widehat{r} \otimes \widehat{b}_j^{(k+1)}\right)\right] \oplus \left[-2^t \left(s \otimes \widehat{b}_{j+1}^{(k+1)}\right)\right] && (3.5)\\
&= -2^t \left(\left[\widehat{r} \otimes \widehat{b}_j^{(k+1)}\right] \oplus \left[s \otimes \widehat{b}_{j+1}^{(k+1)}\right]\right) && (3.6)\\
&= -2^t \widehat{b}_j^{(k)}. && (3.7)
\end{aligned}$$

Thus, for such $p(s)$, we only need compute $\widehat{b}_0^{(k)}$:

$$\widehat{b}_0^{(k)} = \left[\widehat{r} \otimes \widehat{b}_0^{(k+1)}\right] \oplus \left[s \otimes \widehat{b}_1^{(k+1)}\right] = \left[\widehat{r} \otimes \widehat{b}_0^{(k+1)}\right] \oplus \left[\left(-2^t\right) \left(s \otimes \widehat{b}_0^{(k+1)}\right)\right]. \tag{3.8}$$

Theorem 3.1. For a polynomial of the form

$$p(s) = b_0 \left[(1-s) - 2^t s\right]^n \tag{3.9}$$

the relative accuracy of the computed evaluation with the de Casteljau algorithm satisfies the following a priori bound:

$$\frac{|p(s) - \texttt{DeCasteljau}(p, s)|}{|p(s)|} \leq (1 + |\phi|\,\gamma_3)^n - 1 \tag{3.10}$$

where

$$\phi \coloneqq \frac{(1-s) + 2^t s}{(1-s) - 2^t s} = \frac{1 + (2^t - 1)\,s}{1 - (2^t + 1)\,s}. \tag{3.11}$$

Proof. Let $r = 1 - s$ and $\widehat{r} = 1 \ominus s$. For $k < n$,

$$\widehat{b}_0^{(k)} = \left[\widehat{r} \otimes \widehat{b}_0^{(k+1)}\right] \oplus \left[\left(-2^t\right) \left(s \otimes \widehat{b}_0^{(k+1)}\right)\right] = \widehat{b}_0^{(k+1)} \left[r(1+\theta_3) - 2^t s(1+\theta_2)\right]. \tag{3.12}$$

Note that this can be written as $\widehat{b}_0^{(k+1)} \left[(r - 2^t s) + E\right]$ where the round-off term satisfies $|E| \leq (r + 2^t s)\,\gamma_3$ for $s \in [0, 1]$. Hence we write $\widehat{b}_0^{(k)} = \widehat{b}_0^{(k+1)} \left[(r - 2^t s) + (r + 2^t s)\,\theta_3\right]$. Since $\widehat{b}_0^{(n)} = b_0$, we have

$$\widehat{b}_0^{(0)} = b_0 \left[\left(r - 2^t s\right) + \left(r + 2^t s\right)\theta_3\right]^n. \tag{3.13}$$

Dividing this by $b_0^{(0)} = b_0 \left(r - 2^t s\right)^n$ we have

$$\widehat{b}_0^{(0)} = b_0^{(0)} \left(1 + \phi \cdot \theta_3\right)^n. \tag{3.14}$$

Hence our relative error can be bound by

$$\left|(1 + \phi \cdot \theta_3)^n - 1\right| \leq (1 + |\phi|\,\gamma_3)^n - 1 \tag{3.15}$$

as desired. ■

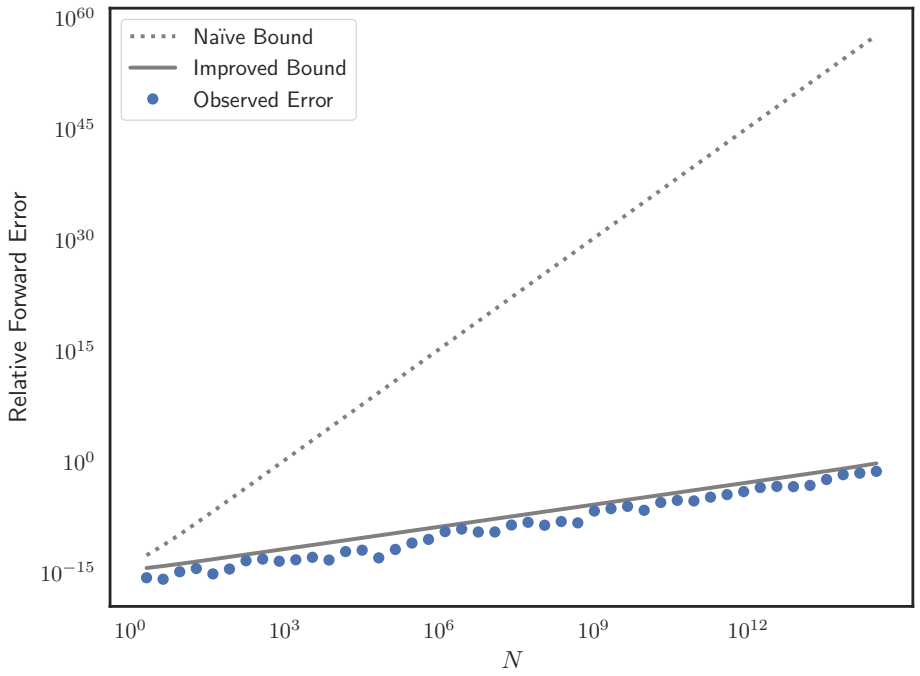

Figure 3.1: Comparing naïve relative error bound to improved bound for $p(s) = (1 - 5s)^5$ evaluated at $s = 1/5 + 8/(25N)$

Since

$$\widetilde{p}(s) = \sum_{j=0}^{n} \left| b_0 \left(-2^t\right)^j \right| B_{j,n}(s) = |b_0| \left[(1 - s) + 2^t s\right]^n \tag{3.16}$$

we see that $\operatorname{cond}(p, s) = \widetilde{p}(s)/|p(s)| = |\phi|^n$. So we can compare our improved bound $(1 + |\phi| \gamma_3)^n - 1$ to the naïve bound $\gamma_{3n} |\phi|^n$.

For example, we'll use $b_0 = 1, n = 5, t = 2$, i.e. $p(s) = (1 - 5s)^5$ for a numerical experiment with. We compare the naïve and improved bounds to the observed relative forward error in Figure 3.1. The figure shows the evaluation of $p(s)$ at the points $\left\{\frac{1}{5} + \frac{8}{25 \cdot 2.1^e}\right\}_{e=1}^{45}$, which cause the condition number of evaluation to grow exponentially. The form $s = 1/5 + 8/(25N)$ is chosen because it simplifies ϕ:

$$\phi = \frac{1 + 3s}{1 - 5s} = -N - \frac{3}{5}. \tag{3.17}$$

As can be seen in the figure, the observed errors closely match the improved bound and are significantly smaller than the naïve bound. What's more, the actual error is still useful (i.e. less than $\mathcal{O}(1)$) when the condition number becomes very large.

4 Implications

In [DP15], the de Casteljau algorithm is compared to the VS algorithm ([SV86]) along with a few other methods. The VS algorithm relies on two transformations of $p(s) = \sum_{j=0}^{n} \widetilde{b}_j (1-s)^{n-j} s^j$ using $\sigma_1 = (1-s)/s$ and $\sigma_2 = s/(1-s)$:

$$p(s) = s^n \left[\widetilde{b}_0 \sigma_1^n + \cdots + \widetilde{b}_n\right] = (1 - s)^n \left[\widetilde{b}_0 + \cdots + \widetilde{b}_n \sigma_2^n\right]. \tag{4.1}$$

See Appendix B for more details on the VS algorithm, in particular Algorithm B.1 which describes the method and Theorem B.1 which provides an error bound.

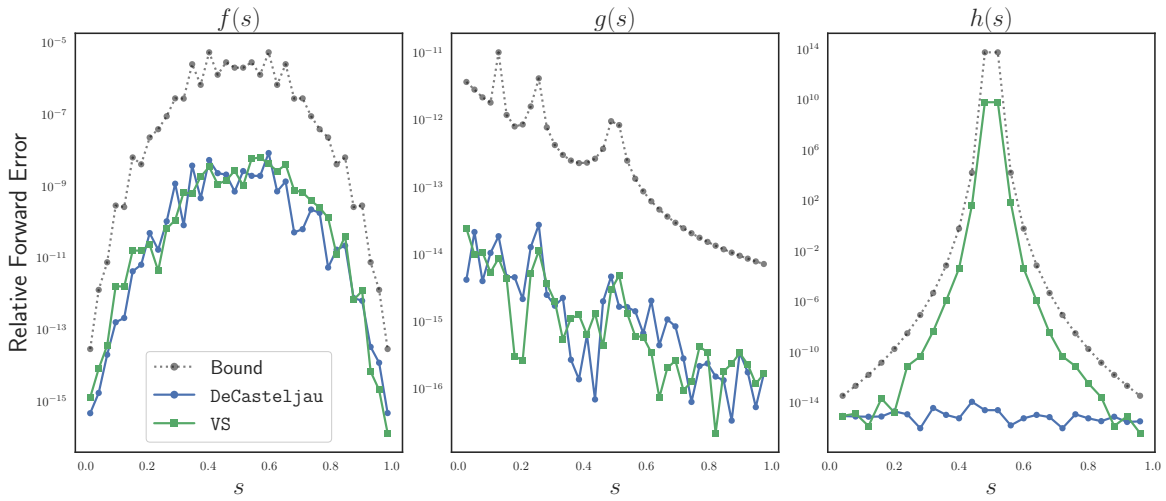

Figure 4.1: Comparing the de Casteljau and VS algorithms.

In [DP15], the authors use[2]

$$f(s) = \prod_{j=1}^{20}(s - j/20), \quad g(s) = \prod_{j=1}^{20}(s - 2/2^j), \quad h(s) = (s - 1/2)^{20} \tag{4.2}$$

to compare the relative error of various methods. Figure 4.1 reproduces the numerical experiments they performed by evaluating $f(s)$ at $\left\{\frac{2j-1}{72}\right\}_{j=1}^{36}$, $g(s)$ at $\left\{\frac{j}{39}\right\}_{j=1}^{38}$ and $h(s)$ at $\left\{\frac{4j}{100}\right\}_{j=1}^{24}$. In addition, we have included a dotted line marking the a prior error bound for each point of evaluation. Viewing the errors for $f(s)$ and $g(s)$, there is no qualitative difference between the de Casteljau algorithm and the VS algorithm, so the authors use $h(s)$ as the tiebreaker to conclude

> the algorithm with a good behavior everywhere is the de Casteljau algorithm.

However,

$$h(s) = (s - 1/2)^{20} = 2^{-20}(2s - 1)^{20} = 2^{-20}\left[(1 - s) - s\right]^{20}, \tag{4.3}$$

i.e. it is a member of our special family of polynomials with $b_0 = 2^{-20}, n = 20, t = 0$. This gives the de Casteljau algorithm an unfair advantage over the VS algorithm in a test case that is not representative of general polynomials.

Given the quadratic growth in the number of arithmetic operations performed in the de Casteljau algorithm, an alternative with linear growth (i.e. the VS method) should not be discounted if it produces similar results. The special family of polynomials explored here explains why the de Casteljau algorithm performs so much better than the VS method and shows that such polynomials are not indicative of the accuracy of the two methods relative to one another.

5 Acknowledgements

The author would like to thank W. Kahan the publication ([Kah72]) that motivated the title of this work and is closely related to the topic at hand.

References

[DP09] Jorge Delgado and J. M. Peña. Running relative error for the evaluation of polynomials. *SIAM Journal on Scientific Computing*, 31(5):3905–3921, Jan 2009.

[2]The coefficients of $f(s)$ and $g(s)$ cannot be represented exactly in IEEE-754 floating point double precision.

[DP15] Jorge Delgado and J.M. Peña. Accurate evaluation of Bézier curves and surfaces and the Bernstein-Fourier algorithm. *Applied Mathematics and Computation*, 271:113–122, Nov 2015.

[FR87] R.T. Farouki and V.T. Rajan. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design*, 4(3):191–216, Nov 1987.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Jan 2002.

[Kah72] William Kahan. Conserving confluence curbs ill-condition. Technical report, UC Berkeley Department of Computer Science, Aug 1972.

[MP99] E. Mainar and J.M. Peña. Error analysis of corner cutting algorithms. *Numerical Algorithms*, 22(1):41–52, 1999.

[MP05] E. Mainar and J. M. Peña. Running Error Analysis of Evaluation Algorithms for Bivariate Polynomials in Barycentric Bernstein Form. *Computing*, 77(1):97–111, Dec 2005.

[SV86] Larry L. Schumaker and Wolfgang Volk. Efficient evaluation of multivariate polynomials. *Computer Aided Geometric Design*, 3(2):149–154, Aug 1986.

A Proof Details

Proof of Theorem 2.2. When using the de Casteljau method, we have $b_j^{(n)} = \widehat{b}_j^{(n)} = b_j$ and for $k = n-1, \ldots, 0$ and $j = 0, \ldots, k$:

$$b_j^{(k)} = (1-s)b_j^{(k+1)} + sb_{j+1}^{(k+1)} \tag{A.1}$$

$$\widehat{b}_j^{(k)} = \left[(1 \ominus s) \otimes \widehat{b}_j^{(k+1)}\right] \oplus \left[s \otimes \widehat{b}_{j+1}^{(k+1)}\right]. \tag{A.2}$$

This means that

$$\widehat{b}_j^{(k)} = (1-s)\widehat{b}_j^{(k+1)}(1+\theta_3) + s\widehat{b}_{j+1}^{(k+1)}(1+\theta_2) \tag{A.3}$$

so that

$$\widehat{b}_0^{(0)} = (1-s)\widehat{b}_0^{(1)}(1+\theta_3) + s\widehat{b}_1^{(1)}(1+\theta_2) \tag{A.4}$$

$$= \widehat{b}_0^{(2)}B_{0,2}(s)(1+\theta_6) + \widehat{b}_1^{(2)}B_{1,2}(s)(1+\theta_5) + \widehat{b}_2^{(2)}B_{2,2}(s)(1+\theta_4) \tag{A.5}$$

$$= \sum_{j=0}^{n} \widehat{b}_j^{(n)}B_{j,n}(s)(1+\theta_{3n-j}) \tag{A.6}$$

$$= b_0^{(0)} + \sum_{j=0}^{n} b_j B_{j,n}(s)\theta_{3n-j}. \tag{A.7}$$

Hence we have

$$|p(s) - \texttt{DeCasteljau}(p, s)| \leq \gamma_{3n}\widetilde{p}(s) \tag{A.8}$$

as desired. Note that this differs from the bound given in [MP99], Corollary 3.2 because the authors don't consider the round-off when computing $\widehat{r}$. ■

B VS Algorithm

In [SV86], a modified form of Horner's method is described for evaluating a polynomial in Bernstein form. Following (4.1), the algorithm applies Horner's method to the scaled Bernstein coefficients with an input related to s:

Algorithm B.1 *VS algorithm for polynomial evaluation.*

function $\texttt{result} = \texttt{VS}(b, s)$
  $n = \texttt{length}(b) - 1$
  $\widehat{r} = 1 \ominus s$
  if $s \geq 1/2$ **then**
    $\widehat{\sigma} = \widehat{r} \oslash s$
    $m = s$
    $[c_0, \ldots, c_n] = [b_n, \ldots, b_0]$
  else
    $\widehat{\sigma} = s \oslash \widehat{r}$
    $m = \widehat{r}$
    $[c_0, \ldots, c_n] = [b_0, \ldots, b_n]$
  end if

  $\widehat{p}_n = c_n$
  for $k = n - 1, \ldots, 0$ **do**
    $\widehat{p}_k = [\widehat{\sigma} \otimes \widehat{p}_{k+1}] + \left[\binom{n}{k} \otimes c_k\right]$
  end for

  $\widehat{m}_1 = \widehat{m}$
  for $k = 2, \ldots, n$ **do**
    $\widehat{m}_k = \widehat{m}_{k-1} \otimes \widehat{m}$
  end for

  $\texttt{result} = \widehat{m}_n \otimes \widehat{p}_0$
end function

This has the benefit of using a linear number of floating point operations, as compared to the de Casteljau method, which uses a quadratic number of floating point operations.

Theorem B.1. The value computed by the VS algorithm (Algorithm B.1) satisifies[3]

$$|p(s) - \texttt{VS}(p, s)| \leq \begin{cases} \gamma_{6n}\widetilde{p}(s) & \text{when } s \in [0, 1/2) \\ \gamma_{5n}\widetilde{p}(s) & \text{when } s \in [1/2, 1]\,. \end{cases} \tag{B.1}$$

Proof. Since the algorithm has two branches depending on $s \geq 1/2$, we have to make a few distinctions throughout. However, most arguments apply to both branches of the algorithm. The following analysis **assumes** that there is no round-off introduced by the computation of binomial coefficients. For $n \leq 56$, $\binom{n}{k}$ can be represented exactly in IEEE-754 floating point double precision for all k but $\binom{57}{25}$ is the "first" binomial coefficient that must be rounded.

In either case, computing $\widehat{\sigma} = (1 \ominus s) \oslash s$ or $\widehat{\sigma} = s \oslash (1 \ominus s)$ requires two floating point operations, so $\widehat{\sigma} = \sigma\,(1 + \theta_2)$. This round-off factor contributes to $2n$ of the $5n$ (or $6n$) in the coefficient of $\widetilde{p}(s)$.

When $s < 1/2$, we apply Horner's method to the scaled Bernstein coefficients, however when $s \geq 1/2$ we reverse the order before performing Horner's method. As a result, we refer to c_k and $\widetilde{c}_k$ instead of b_k and $\widetilde{b}_k$. In the $s < 1/2$ case, $c_k = b_k$ and in the other $c_k = b_{n-k}$. When computing $p(s)$, we start with $\widehat{p}_n = c_n$ and then for $k = n - 1, \ldots, 0$:

$$\widehat{p}_k = [\widehat{\sigma} \otimes \widehat{p}_{k+1}] \oplus \left[\binom{n}{k} \otimes c_k\right]. \tag{B.2}$$

This means that

$$\widehat{p}_k = \sigma\widehat{p}_{k+1}(1 + \theta_4) + \widetilde{c}_k(1 + \theta_2) \tag{B.3}$$

so that[4]:

$$\widehat{p}_0 = \sigma\widehat{p}_1(1 + \theta_4) + \widetilde{c}_0(1 + \theta_2) \tag{B.4}$$

[3] The coefficients γ_{5n} and γ_{6n} differ from γ_{4n} in [DP09] (Theorem 4.2) because the authors don't account for the multiplication by $\binom{n}{k}$ in computing the scaled Bernstein coefficients or the round-off in $1 \ominus s$ when computing $(1 - s)^n$.

[4] We could ignore round-off from the multiplication by $\binom{n}{0}$, but we don't.

$$= \sigma^2 \widehat{p}_2(1+\theta_8) + \sigma \widetilde{c}_1(1+\theta_6) + \widetilde{c}_0(1+\theta_2) \tag{B.5}$$

$$\vdots$$

$$= \sigma^n \widehat{p}_n(1+\theta_{4n}) + \sum_{j=0}^{n-1} \widetilde{c}_j \sigma^j (1+\theta_{4j+2}). \tag{B.6}$$

Defining $\widehat{m}_0 = 1, \widehat{m}_1 = m$ and $\widehat{m}_{k+1} = \widehat{m}_k \otimes m$ we'll have $\widehat{m}_n = m^n(1+\theta_{n-1})$. Hence, in the final step of the VS algorithm we have

$$\widehat{m}_n \otimes \widehat{p}_0 = m^n \widehat{p}_0(1+\theta_n) = m^n \left[\sigma^n \widetilde{c}_n(1+\theta_{5n}) + \sum_{j=0}^{n-1} \widetilde{c}_j \sigma^j (1+\theta_{4j+2+n}) \right]. \tag{B.7}$$

When $s < 1/2$, $m = \widehat{r} = (1-s)(1+\theta_1)$, hence $m^n = (1-s)^n(1+\theta_n)$, $\sigma = s/(1-s)$ and

$$\widehat{m}_n \otimes \widehat{p}_0 = (1-s)^n \left[\sigma^n \widetilde{b}_n(1+\theta_{6n}) + \sum_{j=0}^{n-1} \widetilde{b}_j \sigma^j (1+\theta_{4j+2+2n}) \right] \tag{B.8}$$

$$= p(s) + (1-s)^n \left[\sigma^n \widetilde{b}_n \theta_{6n} + \sum_{j=0}^{n-1} \widetilde{b}_j \sigma^j \theta_{4j+2+2n} \right] \tag{B.9}$$

$$= p(s) + b_n B_{n,n}(s) \theta_{6n} + \sum_{j=0}^{n-1} b_j B_{n-j,n}(s) \theta_{4j+2+2n} \tag{B.10}$$

hence

$$|p(s) - \texttt{VS}(p,s)| \leq \gamma_{6n} \widetilde{p}(s). \tag{B.11}$$

When $s \geq 1/2$, $\sigma = (1-s)/s$. Since $m = s$, no extra round-off accumulates in m^n, but the order of the coefficients is reversed:

$$\widehat{m}_n \otimes \widehat{p}_0 = s^n \left[\sigma^n \widetilde{b}_0(1+\theta_{5n}) + \sum_{j=0}^{n-1} \widetilde{b}_{n-j} \sigma^j (1+\theta_{4j+2+n}) \right] \tag{B.12}$$

$$= p(s) + s^n \left[\sigma^n \widetilde{b}_0 \theta_{5n} + \sum_{j=0}^{n-1} \widetilde{b}_{n-j} \sigma^j \theta_{4j+2+n} \right] \tag{B.13}$$

$$= p(s) + b_0 B_{0,n}(s) \theta_{5n} + \sum_{j=0}^{n-1} b_{n-j} B_{n-j,n}(s) \theta_{4j+2+n} \tag{B.14}$$

hence

$$|p(s) - \texttt{VS}(p,s)| \leq \gamma_{5n} \widetilde{p}(s) \tag{B.15}$$

as desired. ■