Contents

1 IEEE Stuff

Algorithm 1: *EFT of the sum of two floating point numbers.*

function $[x, y] = \texttt{TwoSum}(a, b)$
$x = a \oplus b$
$z = x \ominus a$
$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

Algorithm 2: *Splitting of a floating point number into two parts.*

function $[x, y] = \texttt{Split}(a)$
$z = a \otimes (2^r + 1)$
$x = z \ominus (z \ominus a)$
$y = a \ominus x$

Algorithm 3: *EFT of the product of two floating point numbers.*

function $[x, y] = \texttt{TwoProd}(a, b)$
$x = a \otimes b$
$[a_h, a_l] = \texttt{Split}(a)$
$[b_h, b_l] = \texttt{Split}(b)$
$y = a_l \otimes b_l \ominus (((x \ominus a_h \otimes b_h) \ominus a_l \otimes b_h) \ominus a_h \otimes b_l)$

Algorithm 4: *EFT of the sum of two floating point numbers with a FMA.*

function $[x, y] = \texttt{TwoProdFMA}(a, b)$
$x = a \otimes b$
$y = \texttt{FMA}(a, b, -x)$

2 de Casteljau's Method

Consider de Casteljau's method to evaluate a degree n polynomial in Bernstein-Bézier form with control points p_j:

$$\begin{aligned} b_j^{(0)} &= p_j \\ b_j^{(k)} &= (1 - s) b_j^{(k-1)} + s b_{j+1}^{(k-1)} \\ b(s) &= b_0^{(n)}. \end{aligned}$$

2.1 Condition Number

For a polynomial $p(x)$ in the power basis, we have ([LGL06]):

$$\operatorname{cond}(p(x)) = \frac{\widetilde{p}(|x|)}{|p(x)|} = \frac{\sum_j |a_j| \, |x|^j}{|p(x)|}.$$

In particular, this means that if $x \geq 0$ and each $a_j \geq 0$ we must necessarily have $\operatorname{cond}(p(x)) = 1$. To see an example in use, consider $p(x) = (x-1)^n$ and input values of the form $x = 1 + \delta$ (with $|\delta| \ll 1$). Since $a_j = \binom{n}{j}(-1)^{n-j}$ we have $\widetilde{p}(x) = (x+1)^n$ hence

$$\operatorname{cond}(p(1+\delta)) = \frac{(2+\delta)^n}{|\delta|^n} = \left|1 + \frac{2}{\delta}\right|^n.$$

As $\delta \to 0$, this value approaches ∞ (as expected).

For a polynomial $p(s)$ in Bernstein form, we have ([JLCS10]):

$$\operatorname{cond}(p(s)) = \frac{\widetilde{p}(s)}{|p(s)|} = \frac{\sum_j |p_j|\, |b_{j,n}(s)|}{|p(s)|}.$$

The Bernstein form is suited for $s \in [0,1]$, which means $b_{j,n}(s) \geq 0$ typically. If $s \in [0,1]$ and each $p_j \geq 0$ we must necessarily have $\operatorname{cond}(p(s)) = 1$. To see an example in use, consider

$$p(s) = (1-2s)^n = [(1-s) - s]^n = \sum_j \binom{n}{j} (1-s)^{n-j}(-s)^j = \sum_j (-1)^j b_{j,n}(s)$$

and input values of the form $x = \frac{1}{2} + \delta$ (with $|\delta| \ll \frac{1}{2}$). Since $p_j = (-1)^j$ we have $\widetilde{p}(s) = [(1-s)+s]^n = 1$

$$\operatorname{cond}\left(p\left(\frac{1}{2} + \delta\right)\right) = \frac{1}{|2\delta|^n}.$$

As $\delta \to 0$, this value approaches ∞ (as expected).

2.2 Selection of Test Cases

From [DP15] (end of Section 3):

> We can observe that, in this case, the algorithm with a good behavior everywhere is the de Casteljau algorithm

In the same paper (when referring to [Bez13] at the beginning of Section 2):

> assuming that all control points are positive. This assumption avoided ill-conditioned polynomials. In this section, we shall show that this is a natural assumption in Computer Aided Geometric Design (from now on, C.A.G.D.) and that it permits to assure high relative precision for the evaluation through a large family of representations in C.A.G.D.

From the same author, in [MP05] (towards the end of Section 5, at the bottom of page 109):

> Let us observe that in this case, the de Casteljau algorithm presents better stability properties for the evaluation near the roots. In fact, the de Casteljau algorithm has good behaviour even when using simple precision, although the running error bound is not so accurate in points close to the roots.

2.3 K-Fold Error Filtering

After implementing for $K = 2, 3, \ldots, 12$ and instrumenting all relevant floating point operations, the K-fold Horner requires

$$(5 \cdot 2^K - 8)n + \left((K+8)2^K - 12K - 6\right) = \mathcal{O}\left((n+K)2^K\right)$$

flops to evaluate a degree n polynomial (this only applies when $n \geq K - 1$). As a comparison, the non-compensated form of Horner requires $2n$ flops. Of these, $\left(2^{K-1} - 1\right) n - 2^{K-1}(K-3) - 2$ are FMA (fused-multiply-add) instructions.

After implementing for $K = 2, 3, 4, 5$ and instrumenting all relevant floating point operations, the K-fold de Casteljau requires

$$(15K^2 - 34K + 26)T_n + K + 5 = \mathcal{O}\left(n^2 K^2\right)$$

flops to evaluate a degree n polynomial. (Here T_n is the nth triangular number.) As a comparison, the non-compensated form of de Casteljau requires $3T_n + 1$ flops. Of these, $(3K - 4)T_n$ are FMA instructions. On hardware that doesn't support FMA, every FMA will be exchanged for 10 $\ominus$'s and 6 $\otimes$'s so the count will increase by $(10 + 6 - 1)(3K - 4)T_n$.

3 Bogus Section for Refs

Here they are, for now

- Compensated Horner ($K = 2$) ([LGL06])
- Compensated de Casteljau ([JLCS10])
- Newton with compensated Horner ([Gra08])
- K-fold Sum ([ORO05])
- K-fold Horner ([GLL09])

References

[Bez13] Licio Hernanes Bezerra. Efficient computation of Bézier curves from their Bernstein–Fourier representation. *Applied Mathematics and Computation*, 220:235–238, sep 2013.

[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.

[GLL09] Stef Graillat, Philippe Langlois, and Nicolas Louvet. Algorithms for accurate, validated and fast polynomial evaluation. *Japan Journal of Industrial and Applied Mathematics*, 26(2-3):191–214, oct 2009.

[Gra08] Stef Graillat. Accurate simple zeros of polynomials in floating point arithmetic. *Computers & Mathematics with Applications*, 56(4):1114–1120, aug 2008.

[JLCS10] Hao Jiang, Shengguo Li, Lizhi Cheng, and Fang Su. Accurate evaluation of a polynomial and its derivative in Bernstein form. *Computers & Mathematics with Applications*, 60(3):744–755, aug 2010.

[LGL06] Philippe Langlois, Stef Graillat, and Nicolas Louvet. Compensated Horner scheme. In Bruno Buchberger, Shin'ichi Oishi, Michael Plum, and Sigfried M. Rump, editors, *Algebraic and Numerical Algorithms and Computer-assisted Proofs*, number 05391 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[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.

[ORO05] Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, jan 2005.