Exact Computation of a Sum or Difference with Applications to Argument Reduction

Warren E. Ferguson, Jr.

Southern Methodist University, PO Box 750156, 208 Clements Hall, Dallas, TX 75275-0156

Abstract

Results are presented that identify when the computed value of a sum or difference is exact. The accuracy of an argument reduction algorithm is analyzed using these results. This analysis demonstrates that catastrophic cancellation does not occur in this algorithm's computation of the reduced argument.

1: Introduction

The accurate computation of transcendental functions sometimes requires the accurate computation of reduced arguments.

Consider, for example, the argument reduction used by Tang [6] to compute an accurate approximation of $\exp(x)$. A naive version of this argument reduction algorithm begins by splitting the input argument x into two pieces (N, r) where $x = N\frac{\ln(2)}{32} + r$ with N an integer and $|r| \leq \ln(2)/64$. Mathematically,

$$N = \text{RoundToNearestInteger}\left(x \cdot \frac{32}{\ln(2)}\right), \text{ and}$$

$$r = x - N\,\frac{\ln(2)}{32}\,.$$

Once determined, N is split further into two more integers (M, j) where $N = 32M + j$ with $j = N \bmod 32 \geq 0$. The computation of $\exp(x)$ is reduced, then, to

$$\begin{aligned}\exp(x) &= \exp\left(N\frac{\ln(2)}{32} + r\right)\\ &= 2^M\left(2^{j/32} + 2^{j/32}(\exp(r) - 1)\right)\,.\end{aligned}$$

Tang shows how a table look-up of $2^{j/32}$, combined with a polynomial approximation of $\exp(r) - 1$, can lead to an approximation of $\exp(x)$ that is in error by no more than roughly half a unit in the last place. This accuracy, however, is attainable only if r is similarly accurate; note that $1 \leq 2^{j/32} < 2$ and $\exp(r) - 1 \approx r$, so large errors in r produce large errors in $\exp(x)$.

The very nature of this naive argument reduction technique causes cancellation to occur in the difference that yields the reduced argument r. Furthermore, while x is considered exact, the computed value of $N \cdot \ln(2)/32$, when $N \neq 0$, is uncertain simply because $\ln(2)$ is a transcendental number. Consequently, *catastrophic cancellation*[1] can occur in the computation of r and can prevent r from being sufficiently accurate. In this case, cancellation reveals the uncertainty associated with the computed value of $N \cdot \ln(2)/32$.

We say the computed value $x \ominus y$ of $x - y$ is exact when $x \ominus y = x - y$. Section 2 presents results that identify when such computed differences are exact. Exact differences are important because they do not, by themselves, introduce additional errors. Furthermore, exact differences also are used in techniques that increase the accuracy of floating-point computations; see [3,4] for recent results in this area.

Section 3 analyzes Tang's modification of this naive argument reduction algorithm. This analysis verifies that the cancellation that occurs in this modified algorithm is always benign rather than potentially catastrophic.

2: Exact Computation of a Sum or Difference

The phrase *machine number* refers to a normalized base-β floating-point number with p places of precision. Nonzero normalized machine numbers x admit the representation $x = \pm(x_1 \bullet x_2 x_3 \cdots x_p) \cdot \beta^{e(x)}$ where x_i and $e(x)$ are integers with $1 \leq x_1 < \beta$. We call $s(x) \equiv x_1 \bullet x_2 x_3 \cdots x_p$ the significand, and $e(x)$ the exponent, associated with x; note that $1 \leq s(x) < \beta$. We assume $\beta \geq 2$, and we define $e(0) = -\infty$.

[1] Catastrophic cancellation refers to the potentially disastrous loss of accuracy that can occur when nearly equal, but uncertain, numbers are subtracted [1].

1063-6889/95 $4.00 © 1995 IEEE

Whether or not the computed value $x \ominus y$ of the difference $x - y$ of machine numbers x and y will be exact depends, in part, on the relative alignment of the significands $s(x)$, $s(y)$, and $s(x-y)$. Of course, the alignment of these significands is described by their associated exponents $e(x), e(y)$, and $e(x-y)$. In the spirit of Sterbenz's result (Theorem 6 below), where one assumes only the existence of a guard digit, we present the following result.

Theorem 1: Let x and y be machine numbers for which $s(x)$ has $z(x) \geq 0$ trailing zeros, $s(y)$ has $z(y) \geq 0$ trailing zeros, and $e(x-y) \leq \min(\max(e(x), e(y)), e(x)+z(x), e(y)+z(y))$. If subtraction is performed with a guard digit, and underflow does not occur[2], then the computed value $x \ominus y$ of $x - y$ is exact.

Proof: The inequality $e(x-y) \leq \min(\max(e(x), e(y)), e(x)+z(x), e(y)+z(y))$ involves x and y symmetrically. Therefore, by negating and exchanging x and y, if necessary, we can suppose that $0 \leq |y| \leq x$ and $e(x-y) \leq \min(e(x), e(y)+z(y))$. The conclusion follows immediately if either $y = 0$ or $x = y$, so suppose also that neither $y \neq 0$ nor $x \neq y$.

We first establish that $0 \leq e(x) - e(y) \leq z(y) + 1$. The inequality $0 \leq e(x) - e(y)$ follows from the fact that $|y| \leq x$, while the inequality $e(x) - e(y) \leq z(y) + 1$ follows from the following contradiction obtained if one supposes otherwise. Indeed, if it were possible that $e(x) - e(y) \geq z(y) + 2$, then from the inequalities $z(y) \geq 0$ and

$$\beta^{e(x-y)+1} > |x-y| \geq |x| - |y| > (\beta - 1)\beta^{e(x)-1}$$

one would be forced to conclude that

$$e(x-y) > e(x) - 2 \geq e(y) + z(y),$$

and this contradicts the assumptions of the theorem.

Next, consider the case when $0 \leq k \equiv e(x) - e(y) \leq z(y)$. The alignment of y prior to the computation of $x - y$ involves shifting $s(y)$ right k places. The relevant picture of the aligned significands is

$$\begin{array}{rcccccccc} x & = & + & x_1 & \bullet & x_2 & \cdots & x_p \\ -y & = & \pm & d_1 & \bullet & d_2 & \cdots & d_p \\ \hline 0 \leq x-y & = & z_0 & z_1 & \bullet & z_2 & \cdots & z_p \end{array}$$

where d_i denotes the digits of $s(y)$ after the right shift of k places; note that $d_i = 0$ for $i > p$ because $s(y)$ has $z(y)$, where $z(y) \leq k$, trailing zeros. From the inequality $e(x-y) \leq e(x)$ we deduce that $z_0 = 0$, and so we conclude that the computed value $x \ominus y$ of $x - y$ is exact. (This computation can either add or subtract magnitudes.)

[2] The restriction that underflow does not occur is unnecessary when IEEE-754 style gradual underflow is available [2].

Finally, consider the case when $e(x) - e(y) = z(y) + 1$. The alignment of y prior to the computation of $x - y$ involves shifting $s(y)$ right $z(y) + 1$ places. The relevant picture of the aligned significands is

$$\begin{array}{rcccccccc} x & = & + & x_1 & \bullet & x_2 & \cdots & x_p & 0 \\ -y & = & \pm & 0 & \bullet & d_1 & \cdots & d_{p-1} & d_p \\ \hline 0 \leq x-y & = & z_0 & z_1 & \bullet & z_2 & \cdots & z_p & z_{p+1} \end{array}$$

where d_i denotes the digits of $s(y)$ after a right shift of only $z(y)$ places; note that $d_i = 0$ for $i > p$ because $s(y)$ has $z(y)$ trailing zeros. From $e(x-y) \leq e(y) + z(y) < e(x)$ we deduce that $z_0 = z_1 = 0$, and so we conclude that the computed value $x \ominus y$ of $x - y$ is exact if subtraction is performed with a guard digit. (This computation only subtracts magnitudes.) •

Corollary 2: Let x and y be machine numbers for which $e(x-y) \leq \min(e(x), e(y))$. If subtraction is performed with a guard digit, and underflow does not occur[2], then the computed value $x \ominus y$ of $x - y$ is exact.

Proof: This is Theorem 1 specialized to the case where $z(x) = z(y) = 0$. •

In Theorem 1, the restriction that $e(x-y) \leq \max(e(x), e(y))$ is used solely to rule out cases where $z_0 \neq 0$. As the next two results demonstrate, this restriction can be eliminated if one assumes, in addition to the use of a guard digit, either that subtraction is performed with a *carry digit* that handles z_0, or that subtraction is *protofaithful*; see [4] for the definition of *faithful* arithmetic operations. Protofaithful subtraction describes a subtraction for which $x \ominus y = x - y$ whenever $x - y$ is representable as a machine number. Examples of protofaithful subtractions include IEEE arithmetic [2] as well as the arithmetics found on the IBM 370 and DEC VAX computers.

Theorem 3: Let x and y be machine numbers for which $s(x)$ has $z(x) \geq 0$ trailing zeros, $s(y)$ has $z(y) \geq 0$ trailing zeros, and $e(x-y) \leq \min(e(x) + z(x), e(y) + z(y))$. If subtraction is performed with both guard and carry digits, and neither overflow nor underflow[2] occurs, then the computed value $x \ominus y$ of $x - y$ is exact.

Proof: As in the proof of Theorem 1, we can assume that $0 < |y| \leq x$, and so that $e(y) \leq e(x)$. Furthermore, the proof of Theorem 1 applies when $e(x-y) \leq e(x)$, so consider the only remaining case where $e(x-y) = e(x) + 1$; this equality characterizes the situations where $z_0 \neq 0$.

The inequalities $e(x-y) = e(x) + 1 \leq \min(e(x) + z(x), e(y) + z(y))$ imply that both $z(x) \geq 1$ and $0 \leq k \equiv e(x) - e(y) \leq z(y) - 1$. Consequently, the alignment of y prior to the computation of $x - y$ involves shifting $s(y)$ right k places.

The first picture in the proof of Theorem 1 applies to this case because only an add magnitude computation can

lead to $z_0 \neq 0$. In this picture we know the trailing digits x_p and d_p are zero because $z(x) \geq 1$ and $z(y) \geq k+1$, and so the computed value $x \ominus y$ of $x - y$ is exact. •

Theorem 4: Let x and y be machine numbers for which $s(x)$ has $z(x) \geq 0$ trailing zeros, $s(y)$ has $z(y) \geq 0$ trailing zeros, and $e(x-y) \leq \min(e(x)+z(x), e(y)+z(y))$. If subtraction is *protofaithful*, and neither overflow nor underflow occurs[2], then the computed value $x \ominus y$ of $x - y$ is exact.

Proof: The proof reduces to demonstrating that $x - y$ is representable as a machine number because subtraction is protofaithful[3].

As demonstrated in the proof of Theorem 1, we can assume that both $0 < |y| \leq x$ and $x - y \neq 0$. Note that a machine number w is an integer multiple of $\beta^{e(w)+1-p}$. In particular, w must admit the representation $w = m(w)\beta^{e(w)+1-p}$ where $m(w)$ is an integer satisfying $0 \leq |m(w)| < \beta^p$.

Now x is an integer multiple of $\beta^{e(x)+z(z)+1-p}$ because $s(x)$ has $z(x)$ trailing zeros. Similarly, y is an integer multiple of $\beta^{e(y)+z(y)+1-p}$ because $s(y)$ has $z(y)$ trailing zeros. Therefore $x - y$ is an integer multiple

$$\min(\beta^{e(x)+z(x)+1-p}, \beta^{e(y)+z(y)+1-p}) = \beta^{\min(e(x)+z(x), e(y)+z(y))+1-p}.$$

If m denotes this integer multiple, then

$$x - y = m\beta^{\min(e(x)+z(x), e(y)+z(y))+1-p}.$$

We know, from the definition of the exponent $e(x-y)$, that $|x-y| < \beta^{e(x-y)+1}$. Therefore, we conclude that

$$|m|\beta^{\min(e(x)+z(x), e(y)+z(y))+1-p} < \beta^{e(x-y)+1},$$

or equivalently that

$$|m| < \beta^{e(x-y)+p-\min(e(x)+z(x), e(y)+z(y))}.$$

From the inequality $e(x-y) \leq \min(e(x)+z(x), e(y)+z(y))$ we deduce that $|m| < \beta^p$, and so we conclude that $x - y$ is representable as a machine number. •

The next two results have appeared previously in the literature. In one sense, these results are less general than those above because either they consider only the subtract magnitude case or they ignore trailing zeros that might intentionally be present in $s(x)$ or $s(y)$. Like the results above, Theorem 5 was developed for potential application to the analysis of argument reduction techniques [7]. Theorem 6 is part of the foundation of techniques that can increase the accuracy of floating-point computations [3,4].

Theorem 5 (*Ziv* [7]): Let x and y be like-signed machine numbers that satisfy $0 < |y| \leq |x| \leq |y| + \beta^{e(y)+1}$. If subtraction is performed with a guard digit, and underflow does not occur[2], then the computed value $x \ominus y$ of $x - y$ is exact.

Proof: Without loss of generality, assume that both x and y are positive.

If $y \leq x < y + \beta^{e(y)+1}$, then $0 \leq x - y < \beta^{e(y)+1}$ and so $e(x-y) \leq e(y) = \min(e(x), e(y))$. Corollary 2 applies and shows that the computed value $x \ominus y$ of $x - y$ is exact.

If $x = y + \beta^{e(y)+1} = (1 + s(y)/\beta)\beta^{e(y)+1}$, then x is a machine number only if the trailing digit of $s(y)$ is zero. Theorem 1, with $z(x) = 0$ and $z(y) = 1$, then applies and shows that the computed value $x \ominus y$ of $x - y$ is exact. •

Theorem 6 (*Sterbenz* [5], *see also* [1]): Let x and y be like-signed machine numbers for which $|y|/2 \leq |x| \leq 2|y|$. If subtraction is performed with a guard digit, and underflow does not occur[2], then the computed value $x \ominus y$ of $x - y$ is exact.

Proof: Without loss of generality, assume that both x and y are positive. The inequalities $y/2 \leq x \leq 2y$ and $x/2 \leq y \leq 2x$ are equivalent. Therefore, by exchanging x and y, if necessary, we also can assume that $y \leq x$. Clearly $y \leq x \leq 2y < y + \beta^{e(y)+1}$, and so the result follows from Theorem 5. (The proof of Theorem 5 also demonstrates that this result follows from Corollary 2.) •

3: Application

We now return to the analysis of Tang's argument reduction algorithm for $\exp(x)$. For convenience, the following analysis considers rounding errors associated with the use of IEEE-754 single precision arithmetic (SP) [2]. This arithmetic is characterized by $\beta = 2$ and $p = 24$, and its numbers span a dynamic range[4] of $\pm\left[2^{-341}, 2^{320}\{1 - 2^{-24}\}\right]$. Therefore, valid arguments x for $\exp(x)$ are numbers in the range $\left[\ln(2^{-341}), \ln(2^{320}\{1-2^{-24}\})\right] \simeq [-341, 320) \cdot \ln(2)$.

Tang's modification of the naive argument reduction algorithm replaces the computation of the reduced argument r by the following more accurate segmented computation of $r = r_1 + r_2$.

```
N = RoundToNearestInteger(x · InvL);
j = N mod 32;
m = N - j;
If |N| < 2^9 Then r1 = x - N · L1
             Else r1 = (x - m · L1) - j · L1;
r2 = - N · L2;
M  = m/32;.
```

[3] The style of this proof is derived from conversations with Douglas Priest, see also the proof of Lemma 1 on page 12 of [4].

[4] This is the range when bias adjustment is utilized.

[5] For brevity, binary numbers are displayed in hexadecimal format.

The following picture[5] illustrates Tang's definitions of $InvL$, L_1, and L_2:

$$\begin{aligned}\frac{32}{\ln(2)} &= (\overbrace{2E.2A8EC}^{InvL}A5\ldots)_{16}\\ &= (0.\overbrace{B8AA3B}^{InvL}29\ldots)_{16}\cdot 2^6, \text{ and}\\ \frac{\ln(2)}{32} &= (0.\underbrace{058B90}_{L_1}\underbrace{BFBE8E}_{L_2}7BC\ldots)_{16}\\ &= (0.\underbrace{B172}_{L_1}\underbrace{17F7D1C}_{L_2}C\ldots)_{16}\cdot 2^{-5}\end{aligned}$$

Note, in particular, that the significand of L_1 has $24-15=9$ trailing zeros. For convenience, we also define

$$L \equiv \frac{1}{InvL} = \frac{1}{(0.B8AA3B)_{16}2^6} = (0.058B90C\ldots)_{16}.$$

Tang states that, as a result of cancellation, the computation yielding r_1 is exact. The results of the previous section will now be used to justify this statement.

First, we deduce that $|N| \le 341\cdot\ln(2)\cdot 32/\ln(2) = 341\cdot 32 = (2AA0)_{16}$ because $|x| \le 341\cdot\ln(2)$ and $InvL < 32/\ln(2)$. Next, by the definition of the round-to-nearest-integer function, $x=(N+\eta)L$ where $|\eta|\le 1/2$. From this bound we determine, for example, that

$$\begin{aligned}\left|x - N\frac{\ln(2)}{32}\right| &= \left|N\left(L-\frac{\ln(2)}{32}\right)+\eta L\right|\\ &\le (2AA0)_{16}\left|L-\frac{\ln(2)}{32}\right| + \frac{L}{2}\\ &\sim (0.02C5FD\ldots)_{16}\end{aligned}$$

Compare this bound with the bound $\ln(2)/64 \sim (0.02C5C8\ldots)_{16}$ achievable with exact argument reduction. Finally, the analysis of the computation of r_1 breaks naturally into two cases.

Case 1: $|N| < 2^9$.

Recall SP has 24-bit significands. Therefore, NL_1 will be computed exactly because N is a 9-bit integer and the significand of L_1 has 9 trailing zeros. Consider now the computation of $x - NL_1$. This computation is exact when $N=0$, so we assume that $|N|\ge 1$. We find

$$\begin{aligned}|x-NL_1| &= |N(L-L_1)+\eta L|\\ &\le (2^9-1)(L-L_1)+\frac{L}{2} \sim (0.02C74\ldots)_{16},\\ |x| &= |N+\eta|L \ge L/2 \sim (0.02C5C\ldots)_{16}, \text{ and}\\ |NL_1| &\ge L_1 = (0.058B9)_{16}.\end{aligned}$$

Therefore

$$e(x-NL_1) \le -7 \le \min(e(x), e(NL_1)),$$

and so Corollary 2 tells us that $r_1 = x - NL_1$ will be computed exactly. •

Case 2: $2^9 \le |N| \le (2AA0)_{16} < 2^{14}$.

Part 1: Recall $N = m + j$ where $j = N \bmod 32$ and $m = N - j$. Therefore $0 \le j \le 31$ and $0 < |m| \le |N| + |j| \le 341\cdot 32 + 31 = (2ABF)_{16} < 2^{14}$, and so mL_1 will be computed exactly because m is a 14-bit integer (whose last 5 bits are zero) while the significand of L_1 has 9 trailing zeros.

Consider the computation of $x - mL_1$. We find

$$\begin{aligned}|x-mL_1| &= |m(L-L_1)+(j+\eta)L|\\ &\le (2ABF)_{16}(L-L_1)+\left(31\frac{1}{2}\right)\cdot L\\ &\sim (0.AECC\ldots)_{16},\\ |x| &= |N+\eta|L\\ &\ge (2^9-1/2)L \sim (B.145\ldots)_{16}, \text{ and}\\ |mL_1| &\ge 2^9 L_1 = (B.172)_{16}.\end{aligned}$$

Therefore

$$e(x-mL_1) \le -1 < \min(e(x), e(mL_1)),$$

and so Corollary 2 tells us $x - mL_1$ will be computed exactly.

Part 2: Consider the computation $(x - mL_1) - jL_1$. This computation is exact when $j = 0$, so assume that $1 \le j \le 31$. Note that jL_1 will be computed exactly because j is a 5 bit integer while the significand of L_1 has 9 trailing zeros. We find

$$\begin{aligned}|r_1| &= |N(L-L_1)+\eta L|\\ &\le (2AA0)_{16}(L-L_1)+\frac{L}{2} \sim (0.02E5E\ldots)_{16},\\ |x-mL_1| &= |jL_1+(x-NL_1)|\\ &\ge L_1 - |x-NL_1| \sim (0.02A5A\ldots)_{16}, \text{ and}\\ |jL_1| &\ge L_1 = (0.058B9)_{16}.\end{aligned}$$

Therefore

$$e((x-mL_1)-jL_1) \le -7 \le \min(e(x-mL_1), e(jL_1)),$$

and so Corollary 2 tells us $r_1 = (x - mL_1) - jL_1$ will be computed exactly. •

Theorem 6 could be used to establish some, but not all, of these results; see the appendix for details. For example, consider the case when $x = -(E9.946B)_{16}$. In this case $N = -(2A1F)_{16}$, $m = -(2A20)_{16}$, $j = 1$, $x - mL_1 = (0.02A7)_{16}$, and $jL_1/2 = (0.2C5C8)_{16}$. For this argument x we find that Theorem 6 cannot be used to establish that $r_1 = (x - mL_1) - jL_1$ will be computed exactly.

[5] For brevity, binary numbers are displayed in hexadecimal format.

Of course, the goal of Tang's algorithm is to determine an accurate value of the reduced argument. We can assess this accuracy by bounding the values of r_1, r_2 and the error in the pre-rounded segmented value $r_1 + r_2$ of the reduced argument $x - N \cdot \ln(2)/32$.

First, the above analysis led to the following bound on r_1:

$$|r_1| \leq (0.02E5E\ldots)_{16}.$$

Next, we can bound r_2 as follows:

$$|r_2| \sim |N|L_2 \leq (2AA0)_{16}L_2 \sim (0.001FED\ldots)_{16}$$

Finally, the error in the pre-rounded segmented value $r_1 + r_2$ of the reduced argument $x - N\frac{\ln(2)}{32}$ is just the error $\xi \equiv (-NL_2) - \mathrm{fl}(-NL_2)$ associated with $r_2 = \mathrm{fl}(-NL_2)$, the stored SP value[6] of $-NL_2$. From the bound on $|N|L_2$, and assuming round-to-nearest rounding, we deduce $|\xi| \leq (1.FED\ldots)_{16}16^{-9}$. Consequently

$$\begin{aligned}\left[x - N\frac{\ln(2)}{32}\right] - [r_1 + r_2] &= \left[x - N\frac{\ln(2)}{32}\right] - [(x - NL_1) + (-NL_2 - \xi)] \\ &= N\left[L_1 + L_2 - \frac{\ln(2)}{32}\right] + \xi\end{aligned}$$

and so

$$\begin{aligned}&\left|\left[x - N\frac{\ln(2)}{32}\right] - [r_1 + r_2]\right| \\ &\quad \leq (2AA0)_{16}|L_1 + L_2 - \ln(2)/32| + (1.FED\ldots)_{16}2^{-36} \\ &\quad \leq (3.48A2\ldots)_{16}2^{-36}\end{aligned}$$

This bound should be compared with the bound on the error in pre-rounded value of the reduced argument computed by the first naive algorithm. To derive this bound let $\widetilde{L} = \mathrm{RoundToSP}\left(\frac{\ln(2)}{32}\right) = (0.058B90C)_{16}$ and suppose $\mathrm{fl}\left(N\widetilde{L}\right) = N\widetilde{L}(1+\epsilon)$ where $|\epsilon| \leq 2^{-24}$. We then note

$$\begin{aligned}&\left|\left[x - N\frac{\ln(2)}{32}\right] - \left[x - \mathrm{fl}\left(N\widetilde{L}\right)\right]\right| \\ &\qquad = \left|N\left[\widetilde{L}(1+\epsilon) - \frac{\ln(2)}{32}\right]\right| \\ &\qquad \leq (2AA0)_{16}\left\{\left[\widetilde{L} - \frac{\ln(2)}{32}\right] + |\epsilon|\widetilde{L}\right\} \\ &\qquad \leq (A.F44\ldots)_{16}2^{-24}\end{aligned}$$

[6] Suppose numbers are stored in base β with p-digit mantissas. If $\mathrm{fl}(x)$ denotes the stored value of x, then for round-to-nearest arithmetic: $|\mathrm{fl}(x) - x| \leq |x| \cdot \frac{1}{2}\beta^{1-p}$.

4: Comments

The author would like to thank Nick Higham, Douglas Priest, and several anonymous referees for comments that helped clarify issues related to these results. Nick Higham also deserves thanks for reminding the author of reference [5].

References

[1] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, Vol. 23, No. 1, pp. 26-45, 1991. See also "Floating-Point and Computer Systems," *CSL-89-9*, [P89-00119], Xerox Corporation, Palo Alto Research Center, 333 Coyote Hill Rd., Palo Alto, CA, 94304, August 1989.

[2] Institute of Electrical and Electronic Engineers, "IEEE Standard for Binary Floating-Point Arithmetic," *ANSI/IEEE Standard 754-1985*, Institute of Electrical and Electronic Engineers, New York, 1985.

[3] D. M. Priest, "Algorithms for Arbitrary Precision Floating Point Arithmetic," in *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*, June 26-28, 1991, edited by P. Kornerup and D. Matula, pp. 132-143.

[4] D. M. Priest, *On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations*, draft dated 9 November 1992. This Ph.D. dissertation was submitted to U. C. Berkeley and can be retrieved by anonymous ftp to ftp.icsi.berkeley.edu as pub/theory/priest-thesis.ps.Z

[5] P. H. Sterbenz, *Floating-Point Computation*. Prentice-Hall, Englewood Cliffs, NJ, 1974, pg. 138. In particular, see Theorem 4.3.1 and the Corollary that follows.

[6] P. T. P. Tang, "Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic," *ACM Transactions on Mathematical Software*, Vol. 15, No. 2, pp. 144-157, June 1989.

[7] Abraham Ziv, "Fast evaluation of elementary mathematical functions with correctly rounded last bit," *ACM Transactions on Mathematical Software*, Vol. 17, No. 3, pp. 410-423, September 1991.

Appendix

We begin by noting

$$\begin{aligned}\frac{\frac{1}{2}L_1 - L}{L} &= -(0.80001166BA)_{16} \\ \frac{2L_1 - L}{L} &= (0.FFFFBA6518)_{16}, \text{ and} \\ \frac{L - L_1}{L} &= (0.000022CD74)_{16}.\end{aligned}$$

Case 1: $|N| < 2^9$.

Without loss of generality, suppose $N \geq 0$. There is nothing to prove when $N = 0$, so suppose further that $N \geq 1$. We know NL_1 will be computed exactly, and so

Theorem 6 tells us $x - NL_1$ will be computed exactly if we can establish

$$\frac{1}{2}NL_1 \le x \le 2NL_1.$$

Substitute $x = (N+\eta)L$, divide by L, and subtract N to obtain the equivalent inequalities

$$\frac{\frac{1}{2}L_1 - L}{L}N \le \eta \le \frac{2L_1 - L}{L}N\,.$$

These inequalities are valid for all $N \ge 1$ because $|\eta| \le \frac{1}{2}$. Note the crucial use of the fact that L is greater than L_1.

If L_1 were smaller than L we would not have been able to satisfy the left-hand inequality when $N = 1$. This means Theorem 6 would apply for all but $N = 1$; a case that would have to be considered separately. This difficulty is avoided when Corollary 2 is applied. •

Case 2: $2^9 \le |N| \le 341 \cdot 32 = (2AA0)_{16}$ and $2^9 \le |m| \le (2ABF)_{16}$.

Part 1: Without loss of generality, suppose $x > 0$. We know $2^9 \le N \le (2AA0)_{16}$ and $2^9 \le m \le (2ABF)_{16}$. We also know mL_1 will be computed exactly, and so Theorem 6 will tell us $x - mL_1$ will be computed exactly if we can establish

$$\frac{1}{2}mL_1 \le x \le 2mL_1.$$

Substitute $x = (m+j+\eta)L$, divide by L, and subtract m to obtain the equivalent inequalities

$$\frac{\frac{1}{2}L_1 - L}{L}m \le j+\eta \le \frac{2L_1 - L}{L}m\,.$$

These inequalities are valid for all $m \ge 2^9$ because $0 \le j \le 31$ and $|\eta| \le \frac{1}{2}$.

Part 2: There is nothing to prove when $j = 0$, so assume $1 \le j \le 31$. We know jL_1 will be computed exactly and we note $x - mL_1 = m(L - L_1) + (j+\eta)L \ge L/2 > 0$. Therefore, Theorem 6 will tell us $(x - mL_1) - jL_1$ will be computed exactly if we can establish

$$\frac{jL_1}{2} \le x - mL_1 \le 2jL_1.$$

Substitute $x = (m+j+\eta)L$, divide by L, and subtract j to obtain the equivalent inequalities

$$\frac{\frac{1}{2}L_1 - L}{L}j \le \frac{L - L_1}{L}m + \eta \sim (2.2\ldots)_{16}16^{-5}m + \eta \le \frac{2L_1 - L}{L}j\,.$$

The left-hand inequality can fail for large negative m when $\eta \sim -\frac{1}{2}$ and $j = 1$. This arises when, for example, $x = -(E9.946B)_{16}$. For this x we find $N = -2A1F$, $m = -2A20$, $j = 1$, $\eta \sim -(0.7FD3\ldots)_{16}$, and $\frac{L-L_1}{L}m + \eta \sim -(0.858D\ldots)_{16}$. •