Alternative Split Functions and Dekker's Product

Stef GRAILLAT Vincent LEFÈVRE Jean-Michel MULLER

Sorbonne Université Inria CNRS

CNRS, LIP6 ENS de Lyon, Univ. Claude Bernard Lyon 1, LIP

(ARITH-27 paper, 2020, with additional remarks)

RAIM 2021, 2021-05-27

The context

- A binary floating-point system, precision p, no FMA.
- Assume no underflow, no overflow.
- Definition: A real number x fits in t bits if there exist integers M and q such that $x = M \cdot 2^q$ with $|M| < 2^t$.
- Property: The exact product of two FP numbers a and b fits in $2p$ bits.
  Thus one can write: $r_h + r_\ell = ab$ (several choices).
  $\rightarrow$ Algorithm using rounding to nearest (RN): Dekker's product, with $r_h = \mathrm{RN}(ab)$ and $|r_\ell| \leqslant \frac{1}{2}\,\mathrm{ulp}(ab)$.
  Applications: accurate algorithms for dot product, polynomial evaluation...
- This algorithm uses a basic block to split each FP input into two FP numbers that fit in $\lfloor p/2 \rfloor$ bits, so that one can compute products that fit in p bits and are therefore exact.
  Note: if p is odd, $\lfloor p/2 \rfloor + \lfloor p/2 \rfloor = p - 1$, but in base 2, one more bit of information with the sign of r_ℓ.
  $\rightarrow$ Algorithm using rounding to nearest: Veltkamp's splitting.

Generalization in directed rounding modes

The main goal: generalize Dekker's product using a directed rounding mode, either toward $-\infty$ (downward: RD) or toward $+\infty$ (upward: RU).

Reason: changing the rounding mode may be expensive or impossible.
Note: changing the rounding mode is currently not supported by GCC (GCC 10).

Dekker's product: compute the rounded product (to nearest) $r_h = \mathrm{RN}(ab)$, then the error term $r_\ell = ab - r_h$ exactly.
In RD (or RU)[1]: if $r_h = \mathrm{RD}(ab)$, then $|ab - r_h| < \mathrm{ulp}(ab)$, so that $ab - r_h$ fits in p bits, i.e. the error term is still exactly representable.

For the algorithm, we first need to be able to split a FP number...

[1] Here and later, input values are assumed to be $\neq 0$ for the explanations/proofs, but the algorithms will remain valid for 0, with null results.

Veltkamp's splitting

General algorithm with $2 \leqslant s \leqslant p-2$, depending on the position of the split. Here, $s = \lceil p/2 \rceil$. "RN" omitted for exact operations.

SplitRN (Veltkamp's splitting)

$c \leftarrow \mathrm{RN}((2^s+1) \cdot a)$
$a_h \leftarrow \mathrm{RN}(a-c)+c$
$a_\ell \leftarrow a - a_h$
return (a_h, a_ℓ)

How the algorithm works: mathematically, $(a-c)+c = a$, but here, with a rounding: c carefully chosen so that $a_h = \mathrm{RN}_{p-s}(a)$.

$a = a_h + a_\ell$, where a_h is a multiple of $2^s\,\mathrm{ulp}(a)$ and a_ℓ is a multiple of $\mathrm{ulp}(a)$. Due to the use of RN, a_h is a rounded to nearest in precision $p - s = \lfloor p/2 \rfloor$. Consequence: $|a_\ell|$ is minimized.

Therefore, $|a_\ell|$ fits in $s-1$ bits, thus in $\lfloor p/2 \rfloor$ bits.

Splitting in directed rounding modes: SplitRD

This will not work so well, but one can write $a_\ell = A_\ell \cdot \mathrm{ulp}(a)$ and the condition on a_ℓ can be loosened to $A_\ell^2 < 2^p$, which will be sufficient for Dekker's product in directed rounding modes: the exact partial products will fit in p bits.

Roughly speaking, we want to compute a splitting $a = a_h + a_\ell$ in an efficient way, where a_h is a "good" $\lfloor p/2 \rfloor$-bit approximation to a, in the sense that $|a_\ell|$ cannot be too large.

Assume rounding toward $-\infty$ (RD), with the restriction $a \geqslant 0$.

Splitting in directed rounding modes: SplitRD [2]

The first question: How do we build a_h? [Veltkamp: $\mathrm{RN}(a - c) + c$]

$\rightarrow$ With an operation of the form $\mathrm{RD}(a^* - c) + c$, with $a^* \approx a$ using a simple operation, e.g. $a^* = \mathrm{RD}(k \cdot a)$ where k is a constant close to 1.

- Concerning c: Its only goal is to determine at which exponent to split.
  $\rightarrow$ Very similar to Veltkamp's c. The rounding mode (here, RD vs RN) has no influence, except for the binade boundary for $a^* - c$.
- Concerning a^*: We wish to emulate RN (used in Veltkamp's splitting) with RD. We have $\mathrm{RN}(x) = \mathrm{RD}(x + \mathrm{ulp}(x)/2)$, except at some boundary points (not an issue). So we need $a^* \approx a + \mathrm{ulp}(a - c)/2$.
  - Simple operation $\rightarrow$ not an equality: we are sometimes a bit wrong. But the condition on a_ℓ (thus on a_h) has been loosen.
  - With c of the order of $2^s \cdot a$, $\mathrm{ulp}(a - c)$ is of the order of $2^{-\lfloor p/2 \rfloor} \cdot a$, thus k would be around $1 + (1/2) \cdot 2^{-\lfloor p/2 \rfloor}$.
  - The optimal value of k (minimizing the bound on A_ℓ) can be found later, by analyzing the algorithm with k in some interval, and confirmed by testing.
  - Note: for $a < 0$, one would get $k < 1$ (not tried).

Splitting in directed rounding modes: SplitRD [3]

SplitRD, assuming $a \geqslant 0$.

uses $s = \lceil p/2 \rceil$ **and** $k = \mathrm{RN}\left(1 + \frac{2}{3} \cdot 2^{-\lfloor p/2 \rfloor}\right)$.
$a^* \leftarrow \mathrm{RD}(k \cdot a)$
$c \leftarrow \mathrm{RD}((2^s + 1) \cdot a^*)$
$a_h \leftarrow \mathrm{RD}(a^* - c) + c$
$a_\ell \leftarrow a - a_h$
return (a_h, a_ℓ)

Assuming $p \geqslant 2$ and $a \geqslant 0$:

- $a_h + a_\ell = a$;
- a_h is a multiple of $2^s\,\mathrm{ulp}(a)$ and fits in $p - s = \lfloor p/2 \rfloor$ bits;
- a_ℓ is of the form $A_\ell \cdot \mathrm{ulp}(a)$, where A_ℓ is an integer satisfying

$$|A_\ell| \leqslant \frac{4}{3} \cdot 2^{\lceil p/2 \rceil - 1} + \frac{5}{2} \quad \text{and} \quad A_\ell^2 < 2^p.$$

Note: Testing shows that a^* can be replaced by a in the computation of c, but not proved yet.

Splitting in directed rounding modes: SplitRD [4]

The proof: 2 pages. Interesting facts:

- The proof needs $p \geqslant 6$ for p even and $p \geqslant 11$ for p odd.
  $\rightarrow$ Exhaustive tests to check the properties for the smaller values of p.
- The proof starts by assuming $k = 1 + \lambda \cdot 2^{s-p} = 1 + \lambda \cdot 2^{-\lfloor p/2 \rfloor}$ with $0 < \lambda \leqslant 1$. Note: in SplitRD, $\lambda \approx 2/3$.
- One gets $-\lambda \cdot 2^{-\lfloor p/2 \rfloor} \cdot a \leqslant a_\ell \leqslant -\lambda \cdot 2^{-\lfloor p/2 \rfloor} \cdot a + (2^s + 2) \cdot \mathrm{ulp}(a)$, giving bounds that do not depend on a:

$$-2\lambda \cdot 2^{-\lfloor p/2 \rfloor} \leqslant \frac{a_\ell}{2^{e_a}} \leqslant -\lambda \cdot 2^{-\lfloor p/2 \rfloor} + (2^s + 2) \cdot 2^{1-p}.$$

- Best choice of λ obtained when the bounds are equal in absolute value, assuming that the bounds are tight. This gives

$$\lambda = \frac{2 + 2^s}{3} \cdot 2^{1-s} \approx \frac{2}{3}.$$

Splitting in directed rounding modes: SplitRU

For rounding toward $+\infty$ (RU), Algorithm SplitRU derived from SplitRD, thanks to the relation $\mathrm{RU}(x) = -\mathrm{RD}(-x)$, and using a negative constant $k' = -k$.

SplitRU, assuming $a \geqslant 0$.

uses $s = \lceil p/2 \rceil$ **and** $k' = -\mathrm{RN}\left(1 + \frac{2}{3} \cdot 2^{-\lfloor p/2 \rfloor}\right)$.
$a^* \leftarrow \mathrm{RU}(k' \cdot a)$
$c \leftarrow \mathrm{RU}((2^s + 1) \cdot a^*)$
$a_h \leftarrow -(\mathrm{RU}(a^* - c) + c)$
$a_\ell \leftarrow a - a_h$
return (a_h, a_ℓ)

Compared to SplitRD, an additional operation (negation to get a_h).

Remark: It may probably be possible to avoid it, either by computing $c = \mathrm{RU}(-(2^s + 1) \cdot a^*)$ or by choosing k' a bit less than 1.

But this would need to modify the proof.

Exact multiplication using SplitRD

Adaptation of Dekker's multiplication algorithm.

Dekker's product in rounding toward $-\infty$, assuming $p \geqslant 11$, $a \geqslant 0$ and $b \geqslant 0$.

uses $s = \lceil p/2 \rceil$.
$(a_h, a_\ell) \leftarrow \mathrm{SplitRD}(a)$
$(b_h, b_\ell) \leftarrow \mathrm{SplitRD}(b)$
$r_h \leftarrow \mathrm{RD}(a \cdot b)$
$t_1 \leftarrow \mathrm{RD}(-r_h + \mathrm{RD}(a_h \cdot b_h))$ (exact)
$t_2 \leftarrow \mathrm{RD}(t_1 + \mathrm{RD}(a_h \cdot b_\ell))$ (exact)
$t_3 \leftarrow \mathrm{RD}(t_2 + \mathrm{RD}(a_\ell \cdot b_h))$ (exact)
$r_\ell \leftarrow \mathrm{RD}(t_3 + \mathrm{RD}(a_\ell \cdot b_\ell))$ (exact)
return (r_h, r_ℓ)

The FP numbers r_h and r_ℓ satisfy $r_h + r_\ell = ab$.

Remark: The condition $p \geqslant 11$ initially came from SplitRD, which is actually valid for $p \geqslant 2$. Then the proof just requires $p \geqslant 5$ (instead of $p \geqslant 3$ for RN).

Implementation and timings

SplitRN (Veltkamp), SplitRD and SplitRU implemented in C, using `double`.

- Compilers (in 2020) under Linux x86_64 (Debian/unstable): GCC 10 preversion, GCC 9.2.1, Clang 9.
- Compiler options: `-frounding-math -std=c11 -O3 -march=native`.
- Assembly code analyzed. Function: some "move" instructions (one more with GCC 9.2.1) to follow the ABI, except when inlined (preferred).
- Test on an array of inputs (should fit in the cache), run several times.
- On an Intel Xeon CPU E5-2609 v3, with the GCC 10 preversion: without vectorization, +10% time for SplitRD and +17% time for SplitRU; with vectorization (default), all 3 algorithms take the same time.
- On other machines: $(+4\%, +7\%)$ on POWER9, $(+15\%, +15\%)$ on AArch64, up to $(+17\%, +34\%)$ on AMD Opteron.
- But timings are very dependent on the context.
- SplitRN + `fesetround(FE_TONEAREST)`: loss of a factor 5!
  $\rightarrow$ SplitRD/RU can be very interesting when current rounding mode $\neq$ RN.

Conclusion

- We have proposed alternatives to Veltkamp's splitting and Dekker's product for rounding toward $-\infty$ and rounding toward $+\infty$.
- Possible interest on architectures on which changing the rounding mode is expensive or impossible.
- According to exhaustive tests in small precisions, the best values of k for RD are RN $\left(1+\frac{2}{3}\cdot 2^{-\lfloor p/2\rfloor}\right)$ and RD $\left(1+\frac{2}{3}\cdot 2^{-\lfloor p/2\rfloor}\right)$, which are different for some values of p. Both yield a maximum value of $|A_\ell|$ equal to $\left\lfloor \frac{4}{3}\cdot 2^{\lceil p/2\rceil-1}\right\rfloor$ (note the absence of the additional term $5/2$).
  Possible future work: prove these properties in any precision.