Interval operations in rounding to nearest

submitted for publication, February 2007

Siegfried M. Rump [*]   Paul Zimmermann [†]

Abstract

We give a simple and efficient method to simulate interval operations using only rounding to nearest in IEEE 754. The quality in terms of the diameter of the result is significantly improved compared to existing approaches.

Keywords. Floating-point arithmetic, rounding to nearest, predecessor, successor, directed rounding
AMS subject classification (2000). 68-04, 68N30

1 Introduction and notation

Throughout the paper we assume a floating point arithmetic according to the IEEE 754 arithmetic standard [3] with rounding to nearest. Denote the set of (single or double precision) floating-point numbers by $\mathbb{F}$, including $-\infty$ and $+\infty$, and let $\text{fl} : \mathbb{R} \to \mathbb{F}$ denote rounding to nearest according to IEEE 754. This includes especially the rounding "tie to even". Define for single and double precision $\mathbf{u}$, the relative rounding error unit, and η, the smallest positive unnormalized floating point number:

	single precision	double precision
$\mathbf{u}$	2^{-24}	2^{-53}
η	2^{-149}	2^{-1074}

Then $\mathbf{u}$ and η satisfy:

$$\forall \circ \in \{+,-,\times,\div\}\ \forall\, a,b \in \mathbb{F} \setminus \{\pm\infty\} : \quad \text{fl}(a \circ b) = (a \circ b)(1+\lambda) + \mu \tag{1}$$

with $|\lambda| \le \mathbf{u}$ and $|\mu| \le \eta/2$ and at least one of λ, μ is zero, provided $\text{fl}(a \circ b)$ is finite. Note that for addition and subtraction μ is always zero. An important property of the rounding is the monotonicity, that is

$$\forall x, y \in \mathbb{R} \ : \quad x \le y \ \Rightarrow\ \text{fl}(x) \le \text{fl}(y). \tag{2}$$

The floating-point predecessor and successor of a *real* number $x \in \mathbb{R}$ are defined by

$$\text{pred}(x) := \max\{f \in \mathbb{F} : \ f < x\} \quad \text{and} \quad \text{succ}(x) := \min\{f \in \mathbb{F} : \ x < f\},$$

respectively, where, according to IEEE 754, $\pm\infty$ are considered to be floating-point numbers. For example, $\text{succ}(1) = 1 + 2\mathbf{u}$. Using (1) it is not difficult to see that for finite $c \ge 0 \in \mathbb{F}$

$$\min(c(1-2\mathbf{u}), c-\eta) \le \text{pred}(c) \quad \text{and} \quad \text{succ}(c) \le \max(c(1+2\mathbf{u}), c+\eta), \tag{3}$$

[*]Institute for Reliable Computing, Hamburg University of Technology, Schwarzenbergstraße 95, 21071 Hamburg, Germany, and Visiting Professor at Waseda University, Faculty of Science and Engineering, 3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan (`rump@tu-harburg.de`).

[†]Paul Zimmermann, Bâtiment A, INRIA Lorraine, Technopôle de Nancy-Brabois, 615 rue du jardin botanique, BP 101, F-54600 Villers-lès-Nancy, France (`Paul.Zimmermann@loria.fr`).

and similarly for $c < 0$. For $a, b \in \mathbb{F}$ and finite $c := \mathrm{fl}(a \circ b)$ the monotonicity (2) implies

$$a \circ b \in [c_1, c_2] \quad \text{where} \quad c_1 := \mathrm{fl}(\mathrm{fl}(c - 2\mathbf{u}|c|) - \eta) \quad \text{and} \quad c_2 := \mathrm{fl}(\mathrm{fl}(c + 2\mathbf{u}|c|) + \eta). \tag{4}$$

(Note that the above remains true if $a \circ b$ is replaced by any real y, for example $y = \sin x$, as long as c is the correct rounding of y.) This is the usual basis of interval libraries to emulate directed roundings using only rounding to nearest (see, for example, [4]). It is disadvantageous because for $1.5 \cdot 2^k \le |a \circ b| < 2^{k+1}$ the interval $[c_1, c_2]$ is twice as wide as it needed to be, i.e., 4 *ulps* (units in the last place) instead of 2 ulps.

The IEEE 754 standard [3] recommends (but does not require) availability of a function `Nextafter`, where `Nextafter(a,b)` returns the next representable floating-point number of `a` in the direction towards `b`. If `b = a` the result is `a`, if `a` or `b` are `NaN` the result is `NaN`.

The main part of the function `Nextafter` is the computation of the predecessor and successor of a floating-point number. This is obvious if directed roundings, as requested by IEEE 754, are available.

However, frequent change of the rounding mode may be time consuming, or is not supported by the programming language in use. In [2] a corresponding algorithm is given by splitting the floating-point number in two parts, treating the second part as an integer and adding/subtracting 1 regarding possible carry. The algorithm does the job, but is slow. In [1] a corresponding routine is given assuming that a fused multiply and accumulate instruction is available, that is $a \cdot b + c$ with only one rounding, and an unlimited exponent range.

In the following we will describe a simple and efficient routine to compute an interval $[c_1, c_2]$ — with $c_1, c_2 \in \mathbb{F}$ — containing $a \circ b$ for all $a, b \in \mathbb{F}$ and $\circ \in \{+, -, \times, \div\}$ provided $\mathrm{fl}(a \circ b)$ is finite. If $a \circ b$ is not a floating point number, the result is always best possible[1] except a small range near underflow.

2 The result

We use the "unit in the first place" $\mathrm{ufp}(x)$ defined for $x \in \mathbb{R}$ by

$$\mathrm{ufp}(0) := 0 \quad \text{and} \quad \mathrm{ufp}(x) := 2^{\lfloor \log_2 |x| \rfloor} \text{ for } x \neq 0.$$

It denotes the value of the most significant bit in the binary representation of x. Then

$$\forall\, 0 \neq x \in \mathbb{R}: \qquad \mathrm{ufp}(x) \le |x| < 2\mathrm{ufp}(x). \tag{5}$$

This concept proved to be useful in the analysis of new summation and dot product algorithms [5]. The definition is independent of some floating point format. Define

$$\overline{\mathbb{U}} := \{f \in \mathbb{F} \;:\; |f| < \mathbf{u}^{-1}\eta\}. \tag{6}$$

For example, in IEEE 754 double precision, $\overline{\mathbb{U}} = \{f \in \mathbb{F} \;:\; |f| < 2^{-1021}\}$. Note that $\frac{1}{2}\mathbf{u}^{-1}\eta$ is the smallest positive normalized floating-point number. For positive $c \in \mathbb{F}$ such that $\mathrm{succ}(c)$ is finite the following properties are easily verified (see also [5]):

$$\begin{array}{lll}
\text{if } c \in \overline{\mathbb{U}}: & \mathrm{pred}(c) = c - \eta, \quad \mathrm{succ}(c) = c + \eta, & (7) \\
\text{if } \mathbf{u}^{-1}\eta \le c,\ c \neq 2^k: & \mathrm{pred}(c) = c - 2\mathbf{u}\,\mathrm{ufp}(c), \quad \mathrm{succ}(c) = c + 2\mathbf{u}\,\mathrm{ufp}(c), & (8) \\
\text{if } \mathbf{u}^{-1}\eta \le c, \quad c = 2^k: & \mathrm{pred}(c) = c - \mathbf{u}\,\mathrm{ufp}(c), \quad \mathrm{succ}(c) = c + 2\mathbf{u}\,\mathrm{ufp}(c). & (9)
\end{array}$$

Moreover, define for $c \in \mathbb{F} \setminus \{\pm\infty\}$ with $\mathrm{pred}(c)$ and $\mathrm{succ}(c)$ finite:

$$\mathcal{M}^-(c) := \frac{1}{2}\big(\mathrm{pred}(c) + c\big) \quad \text{and} \quad \mathcal{M}^+(c) := \frac{1}{2}\big(c + \mathrm{succ}(c)\big). \tag{10}$$

[1]Since we don't know if $c := \mathrm{fl}(a \circ b)$ is smaller or larger than $a \circ b$, the best possible interval is $[\mathrm{pred}(c), \mathrm{succ}(c)]$, of 2 ulps.

It follows for $c \in \mathbb{F},\ x \in \mathbb{R}$,

$$\begin{array}{rcl} x < \mathcal{M}^-(c) \Rightarrow \mathrm{fl}(x) \le \mathrm{pred}(c) & \text{and} & \mathcal{M}^+(c) < x \Rightarrow \mathrm{succ}(c) \le \mathrm{fl}(x) \\ \mathcal{M}^-(c) < x \Rightarrow c \le \mathrm{fl}(x) & \text{and} & x < \mathcal{M}^+(c) \Rightarrow \mathrm{fl}(x) \le c. \end{array} \tag{11}$$

For given $a, b \in \mathbb{F}$ and $\circ \in \{+, -, \times, \div\}$, consider the following

Algorithm 1 *Bounds for predecessor and successor of finite $c \in \mathbb{F}$ in rounding to nearest*

```
$e = \mathrm{fl}(\mathrm{fl}(\phi|c|) + \eta)$        % $\phi = \mathbf{u}(1+2\mathbf{u}) = \texttt{Nextafter}(\mathbf{u}, +\infty) = \mathrm{succ}(\mathbf{u})$
cinf $= \mathrm{fl}(c - e)$
csup $= \mathrm{fl}(c + e)$
```

Lemma 1 *Let finite $c \in \mathbb{F}$ be given, and let* `cinf` , `csup` $\in \mathbb{F}$ *be the quantities computed by Algorithm 1. Then*

$$\texttt{cinf} \le \mathrm{pred}(c) \qquad \text{and} \qquad \mathrm{succ}(c) \le \texttt{csup}\,. \tag{12}$$

If $|c| \notin [\frac{1}{2}, 2]\mathbf{u}^{-1}\eta$, then both inequalities in (12) are equalities.

Proof. Since $\mathbb{F} = -\mathbb{F}$ and $\mathrm{fl}(-x) = -\mathrm{fl}(x)$ for $x \in \mathbb{R}$, we may assume without loss of generality $c \ge 0$. One verifies the assertions for c being the largest positive floating point number, hence we assume without loss of generality that $\mathrm{succ}(c)$ is finite.

If $c \in \overline{\mathbb{U}}$, then $e \ge \eta$ and (7) imply (12). If $|c| < \frac{1}{2}\mathbf{u}^{-1}\eta$, then $c \le \frac{1}{2}\mathbf{u}^{-1}\eta - \eta$ by (7), so that

$$\phi c \le \frac{1}{2}(1+2\mathbf{u})\eta - \mathbf{u}(1+2\mathbf{u})\eta < \frac{1}{2}\eta$$

implies $\mathrm{fl}(\phi c) = 0$. Hence $e = \eta$, and the lemma is proved for $c \in \overline{\mathbb{U}}$.

Henceforth we may assume $c \notin \overline{\mathbb{U}}$, which means $c \ge \mathbf{u}^{-1}\eta$. Next we prove

$$e > \mathbf{u}\,\mathrm{ufp}(c). \tag{13}$$

Note that $\mathbf{u}\,\mathrm{ufp}(c) \in \mathbb{F}$. By (2), (5), (8) and (9)

$$c' := \mathrm{fl}(\phi c) = \mathrm{fl}(\mathbf{u}(c + 2\mathbf{u}c)) \ge \mathrm{fl}(\mathbf{u}(c + 2\mathbf{u}\,\mathrm{ufp}(c))) = \mathrm{fl}(\mathbf{u}\,\mathrm{succ}(c)). \tag{14}$$

If $\mathbf{u}\,\mathrm{succ}(c)$ is not subnormal, i.e., $\mathbf{u}\,\mathrm{succ}(c) \ge \frac{1}{2}\mathbf{u}^{-1}\eta$, then $\mathbf{u}\,\mathrm{succ}(c) \in \mathbb{F}$ and

$$e = \mathrm{fl}(c' + \eta) \ge c' \ge \mathrm{fl}(\mathbf{u}\,\mathrm{succ}(c)) = \mathbf{u}\,\mathrm{succ}(c) > \mathbf{u}\,\mathrm{ufp}(c),$$

which proves (13). If $\mathbf{u}\,\mathrm{succ}(c)$ is subnormal, i.e, $\mathbf{u}\,\mathrm{succ}(c) < \frac{1}{2}\mathbf{u}^{-1}\eta$, then (14), (2) and (5) imply

$$c' \ge \mathrm{fl}(\mathbf{u}\,\mathrm{succ}(c)) \ge \mathrm{fl}(\mathbf{u}\,\mathrm{ufp}(c)) = \mathbf{u}\,\mathrm{ufp}(c). \tag{15}$$

On the other hand

$$\phi c < \mathbf{u}(1+2\mathbf{u})\mathrm{succ}(c) < \alpha := (1+2\mathbf{u}) \cdot \frac{1}{2}\mathbf{u}^{-1}\eta \in F,$$

implies $c' = \mathrm{fl}(\phi c) \le \alpha < \mathbf{u}^{-1}\eta$, such that (7) and (15) yield

$$e = \mathrm{fl}(c' + \eta) = c' + \eta > \mathbf{u}\,\mathrm{ufp}(c).$$

This proves (13). By (8) and (9) we know

$$c - \mathbf{u}\,\mathrm{ufp}(c) \le \mathcal{M}^-(c) \quad \text{and} \quad \mathcal{M}^+(c) = c + \mathbf{u}\,\mathrm{ufp}(c),$$

so (13) and (11) yield

$$c - e < \mathcal{M}^-(c) \quad \text{and} \quad \mathcal{M}^+(c) < c + e \tag{16}$$

and prove (12). It remains to prove that the inequalities in (12) are sharp for $c > 2\mathbf{u}^{-1}\eta$. In this case

$$c \le \text{pred}(2\text{ufp}(c)) = 2(1 - \mathbf{u})\text{ufp}(c)$$

follows by (5) and (9) and also for $2\text{ufp}(c)$ in the overflow range, so that

$$\phi c = \mathbf{u}(1 + 2\mathbf{u})c < 2\mathbf{u}(1 + \mathbf{u})\text{ufp}(c) =: (1 + \mathbf{u})C. \tag{17}$$

Since $\text{ufp}(c) \ge 2\mathbf{u}^{-1}\eta$ it follows that $C = 2\mathbf{u}\,\text{ufp}(c) \in \mathbb{F}$. If $C \ge \frac{1}{2}\mathbf{u}^{-1}\eta$, then

$$\phi c < (1 + \mathbf{u})C = \mathcal{M}^+(C),$$

and (11) yields $c' = \text{fl}(\phi c) \le C$. If $C < \frac{1}{2}\mathbf{u}^{-1}\eta$, then $C \le \frac{1}{4}\mathbf{u}^{-1}\eta$ and (17) and $C \in \mathbb{F}$ give

$$c' = \text{fl}(\phi c) \le \text{fl}(C + \frac{1}{4}\eta) = C,$$

so that $\text{ufp}(c) \ge 2\mathbf{u}^{-1}\eta$ proves

$$e = \text{fl}(c' + \eta) \le \text{fl}(C + \frac{1}{2}\mathbf{u}\,\text{ufp}(c)) = \text{fl}(\frac{5}{2}\mathbf{u}\,\text{ufp}(c)) = \frac{5}{2}\mathbf{u}\,\text{ufp}(c). \tag{18}$$

Hence (8) and (9) imply

$$c + e \le \text{succ}(c) + \frac{1}{2}\mathbf{u}\,\text{ufp}(c) < \mathcal{M}^+(\text{succ}(c)),$$

so that `csup` $= \text{fl}(c + e) = \text{succ}(c)$ by (10). This proves the right inequality in (12). A similar argument applies when $\text{pred}(\text{pred}(c)) \ge \text{ufp}(c)$ and shows `cinf` $= \text{fl}(c - e) = \text{pred}(c)$. It remains to prove the left inequality in (12) for $\text{ufp}(c) \in \{\text{pred}(c), c\}$. If $\text{pred}(c) = \text{ufp}(c)$, then (18) gives

$$c - e \ge c - \frac{5}{2}\mathbf{u}\,\text{ufp}(c) = \text{pred}(c) - \frac{1}{2}\mathbf{u}\,\text{ufp}(c),$$

and rounding tie to even implies $\text{fl}(c - e) = \text{pred}(c)$. Finally, if $c = \text{ufp}(c)$, then c is a power of 2 and $c > 2\mathbf{u}^{-1}\eta$ yields $c \ge 4\mathbf{u}^{-1}\eta$. In that case either $c \ge \frac{1}{2}\mathbf{u}^{-2}\eta$ and then $\phi c \in \mathbb{F}$; or $c \le \frac{1}{4}\mathbf{u}^{-2}\eta$ and then $\mathbf{u}c \le \phi c \le \mathbf{u}c + \frac{1}{2}\eta$, which shows with rounding tie to even that $\text{fl}(\phi c) = \mathbf{u}c$. In both cases $c' = \text{fl}(\phi c) \le \phi c \le \frac{9}{8}\mathbf{u}c$, and $c' + \eta \le \frac{9}{8}\mathbf{u}c + \frac{1}{4}\mathbf{u}c$ implies

$$e = \text{fl}(c' + \eta) \le \text{fl}(\frac{11}{8}\mathbf{u}c) = \frac{11}{8}\mathbf{u}c < \frac{3}{2}\mathbf{u}c.$$

Hence (8) and (9) imply $\text{pred}(c) = (1 - \mathbf{u})c$ and $\mathcal{M}^-(\text{pred}(c)) = (1 - \frac{3}{2}\mathbf{u})c$, and (11) finishes the proof. $\square$

Given $a, b \in \mathbb{F}$, rigorous and mostly sharp bounds for $a \circ b, \circ \in \{+, -, \times, \div\}$ can be computed by applying Algorithm 1 to $c := \text{fl}(a \circ b)$. This holds for the square root as well. Although addition and subtraction cause no error if the result is in the underflow range, the extra term η cannot be omitted in the computation of e because it is needed for c slightly outside the underflow range.

range	formula (4)	Algorithm 1
$[1/2, 1]$	2 / 2 / 2.999696 / 4	2 / 2 / 2.000000 / 2
$[2^{-1020}, 2^{-1019}]$	2 / 4 / 3.250584 / 4	2 / 2 / 2.000000 / 2
$[2^{-1021}, 2^{-1020}]$	4 / 4 / 4.000000 / 4	2 / 2 / 2.999696 / 4
$[2^{-1022}, 2^{-1021}]$	4 / 4 / 4.999696 / 6	4 / 4 / 4.000000 / 4
$[2^{-1023}, 2^{-1022}]$	4 / 4 / 4.000000 / 4	2 / 2 / 2.000000 / 2
$[2^{-1024}, 2^{-1023}]$	2 / 2 / 2.000000 / 2	2 / 2 / 2.000000 / 2

Figure 1: Minimal, median, average and maximal difference in ulps between bounds of enclosing interval with formula (4) and Algorithm 1, for IEEE 754 double precision. For each range between consecutive powers of two, $1,000,000$ random values of c were used. The excluded range $[\frac{1}{2}, 2]\mathbf{u}^{-1}\eta$ from Lemma 1 corresponds to $[2^{-1022}, 2^{-1020}]$.

3 Conclusion

When using rounding to nearest only, the advantage of Algorithm 1 over formula (4) — for computing rigorous bounds of $a \circ b$ — is that the width is often halved (see Fig. 1).

However, this applies only if $a, b \in \mathbb{F}$. In applications often interval operations $A \circ B$ for thick intervals A, B are executed. The wider A and B are, the less is the gain of Algorithm 1 compared to (4). In practical applications the gain is negligible unless point intervals play a significant role.

Moreover, the algorithm is valid only for finite floating point result. If infinite bounds are allowed, a necessary case distinction — as would be necessary before applying (4) — may slow down the Algorithm.

As a corollary of Lemma 1, we can compute $\mathrm{succ}(c)$ as follows:

Algorithm 2 `Nextabove`(c).

```
$e \leftarrow |c|$
*if* $e \geq 2\mathbf{u}^{-1}\eta$ *then*
    $e \leftarrow \mathrm{fl}(\phi e)$
    $e \leftarrow \mathrm{fl}(e + \eta)$
    *Return* $\mathrm{fl}(c + e)$
*elseif* $e \geq \mathbf{u}^{-1}\eta$ *then*
    *Return* $\mathrm{fl}(c + 2\eta)$
*else*
    *Return* $\mathrm{fl}(c + \eta)$.
```

An algorithm to compute $\mathrm{pred}(c)$ follows similarly. We have implemented this algorithm in the C language, and compared it to the native `nextafter(c,MAXDOUBLE)` implementation. We obtained the following timings in seconds for 10 million calls on a 800Mhz Pentium M under Linux with gcc 4.0.2 (using inline code):

range	nextafter	Algorithm 2
$[1/2, 1]$	1.072	0.192
$[2^{-1020}, 2^{-1019}]$	1.080	0.192
$[2^{-1021}, 2^{-1020}]$	1.076	3.256
$[2^{-1022}, 2^{-1021}]$	1.088	0.156
$[2^{-1023}, 2^{-1022}]$	18.369	0.160
$[2^{-1024}, 2^{-1023}]$	18.389	0.160

References

[1] S. Boldo and J.-M. Muller. Some Functions Computable with a Fused-mac. Technical Report 5320, Institut National de Recherche en Informatique et en Automatique (INRIA), 2004.

[2] W.J. Cody, Jr. and J.T. Coonen. Algorithm 722: Functions to support the IEEE standard for binary floating-point arithmetic. *ACM Transactions on Mathematical Software*, 19(4):443–451, 1993.

[3] American National Standards Institute, Institute of Electrical, and Electronic Engineers. IEEE standard for binary floating-point arithmetic. *ANSI/IEEE Standard, Std. 754-1985*, 1985.

[4] R.B. Kearfott, M. Dawande, K. Du, and C. Hu. Algorithm 737: INTLIB: A Portable Fortran 77 Interval Standard-Function Library. *ACM Transactions on Mathematical Software*, 20(4):447–459, 1994.

[5] S.M. Rump, T. Ogita, and S. Oishi. Accurate Floating-Point Summation. Technical Report 05.1, Faculty of Information and Communication Science, Hamburg University of Technology, 2005.