Isolating critical cases for reciprocals using integer factorization

John Harrison

Intel Corporation

ARITH-16

Santiago de Compostela

17th June 2003

Background

result before the final rounding

Suppose we are interested in a floating-point approximation to a mathematical function $f : \mathbb{R} \to \mathbb{R}$.

Many algorithms for such mathematical functions can be considered as applying IEEE rounding to a rational approximation.

In particular, a sequence of standard floating-point operations can be split at a ‘result before the final rounding’.

Instead of rounding $f(x)$ exactly, we are rounding an approximation $f^*(x)$.

division and square root in software

The following paper showed how to implement correctly rounded division and square root in terms of an initial approximation, using fused multiply-add.

> Peter Markstein, *Computation of elementary functions on the IBM RISC System/6000 processor*, IBM Journal of Research and Development vol. 34 (1990), pp. 111–119

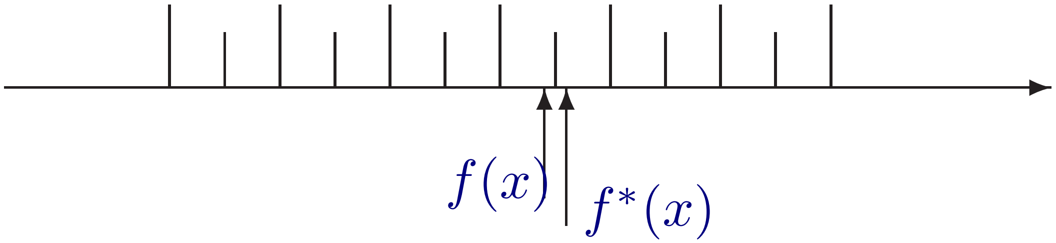
This approach is used for division and square root in the Intel® Itanium® architecture.

Since these operations are supposed to be IEEE-compliant, we must guarantee that they are correctly rounded and set flags correctly.

rounding boundaries

Correct rounding of the result means that $f(x)$ and $f^*(x)$ round the same way.

Essentially, we want to ensure that the following situation is avoided:



where $f(x)$ and $f^*(x)$ are separated by a ‘rounding boundary’:

- A floating-point number (directed rounding modes)
- A midpoint between floating-point numbers (round to nearest)

But the combinatorial details of checking this for all floating-point inputs x are generally painful.

distance to rounding boundary

However, a *sufficient* condition is that $|f^*(x) - f(x)| < |f(x) - b|$ for any rounding boundary b.

This is usually easier to check analytically, by analyzing how small $|f(x) - b|$ can be.

On naive statistical grounds, one would expect the minimum distance to be of order $|f(x)|/2^{2p+d}$ for floating-point precision p and small d.

Results of this kind can be proved analytically for algebraic functions, and have been tested explicitly for many transcendentals:

> Vincent Lefèvre and Jean-Michel Muller, *Worst Cases for Correct Rounding of the Elementary Functions in Double Precision*, INRIA Research Report 4044, 2000

Our aim here is to do the same for reciprocals.

mixed analytical-combinatorial proofs

In practical cases the desired inequality may fail or be difficult to prove analytically for some x.

However, one can then use the approach pioneered in the following paper:

> Marius Cornea-Hasegan, *Proving the IEEE Correctness of Iterative Floating-Point Square Root, Divide and Remainder Algorithms*, Intel Technology Journal 1998-Q2, pp. 1–11

This divides the task into an analytical and combinatorial part:

- Prove analytically that $|f^*(x) - f(x)| \leq \epsilon |f(x)|$
- Find $S_\epsilon = \{x \mid \exists b \in B.\ |f(x) - b| < \epsilon |f(x)|\}$
- Check correct rounding explicitly for each $x \in S_\epsilon$

where B is the set of rounding boundaries.

Difficult cases for reciprocals

scaling the difficult cases

We seek floating-point numbers y and rounding boundaries w such that $w \cdot y \approx 1$.

Write $y = 2^e m$ and $w = 2^e n$ for integers $2^{p-1} \leq m < 2^p$ and $2^p \leq n < 2^{p+1}$, where p is the precision.

Then we want $m \cdot n \approx 2^{2p}$. Thus, we've reduced the problem to an essentially number-theoretic one.

For such difficult cases, the *relative* distance of y from a midpoint is about $|m \cdot n - 2^{2p}|/2^{2p}$.

finding difficult cases with factorization

We use a straightforward algorithm to find the difficult cases. For each $\delta \in \mathbb{Z}$ in the range of interest, we:

- Find the prime factorization of $2^{2p} + \delta = p_1^{\alpha_1} \cdot \cdots \cdot p_k^{\alpha_k}$
- Consider all possible ways of distributing the primes among two integers m and n in the appropriate range.

In general, we refer to a factorization $r = m \cdot n$ with $m < A$ and $n < B$ as an (A, B)-balanced factorization of r.

a toy example (1)

Consider the case $p = 6$ and $\delta \in \{\pm 1, \pm 2, \pm 3\}$. In each case we find the prime factorization of $2^{2p} + \delta$:

$$\begin{aligned}
2^{12} + 1 &= 17 \cdot 241 \\
2^{12} - 1 &= 3^2 \cdot 5 \cdot 7 \cdot 13 \\
2^{12} + 2 &= 2 \cdot 3 \cdot 683 \\
2^{12} - 2 &= 2 \cdot 23 \cdot 89 \\
2^{12} + 3 &= 4099 \\
2^{12} - 3 &= 4093
\end{aligned}$$

a toy example (2)

For $2^{12}+1$, $2^{12}+2$, $2^{12}\pm 3$, there no possible distribution obeying the range restrictions.

For $2^{12}-2$ there is exactly one such distribution, $m \cdot b = 89 \cdot (2 \cdot 23) = 89 \cdot 46$.

For $2^{12}-1$, there are four possible distributions:

$$\begin{aligned}
m \cdot b &= (3^2 \cdot 13) \cdot (5 \cdot 7) = 117 \cdot 35 \\
m \cdot b &= (3 \cdot 5 \cdot 7) \cdot (3 \cdot 13) = 105 \cdot 39 \\
m \cdot b &= (7 \cdot 13) \cdot (3^2 \cdot 5) = 91 \cdot 45 \\
m \cdot b &= (5 \cdot 13) \cdot (3^2 \cdot 7) = 65 \cdot 63
\end{aligned}$$

Hence $S_{2^{-10}} = \{32, 35, 39, 45, 46, 63\}$ (including the exact case).

implementation

We implement a naive recursive algorithm that examines the prime factors in decreasing order.

- For each $p_i^{\alpha_i}$, try the $\alpha_i + 1$ ways of distributing that prime power: $(0, \alpha_i)$, $(1, \alpha_i - 1)$, . . . $(\alpha_i, 0)$.
- In each case, recursively try all ways of distributing the remaining primes.
- As soon as a range violation is encountered, terminate that path.

The implementation is written in Objective CAML (http://www.ocaml.org) using the factorization function from PARI/GP (http://www.parigp-home.de).

Is this practical?

efficiency in practice

Is such a straightforward algorithm really acceptably efficient?

Surprisingly, yes!

- PARI/GP's factorization software usually deals quickly with numbers up to about 300 bits
- The average number of (A, B)-balanced factorizations of a number of size r is about $ln(AB/r)$, which is only $ln(2) \approx 0.693$ in the range we examine
- Even the total number of divisors $d(n) = \Pi_{i=1}^{i=k}(1 + \alpha_i)^n$ is reasonable until we get to quad precision, and this provides an upper bound on the time taken for the distribution process.

For more on these assertions, see the paper.

numbers with balanced factorization

It's theoretically interesting to ask what proportion of positive integers $\leq AB$ have an (A, B)-balanced factorization . . . .

The result $ln(AB/r)$ above counted duplicates.

As r becomes larger, the average number of prime factors increases and hence the degree of duplication increases, slowly but without limit.

Hence the proportion of numbers $\leq AB$ with an (A, B)-balanced factorization tends to zero as A and B tend to infinity.

A paper by Erdös from 1960 gives a precise asymptotic expression for the case $A = B$, namely $ln(A)^{-\alpha+o(1)}$ where $\alpha = 1 - ln(e \cdot ln(2))/ln(2) \approx 0.0860$.

Thanks to Vassil Dimitrov for pointing this out!

An application

standard division algorithm

The usual IPF algorithm for double-extended $(p = 64)$ division is:

1.	$y_0 = \texttt{frcpa}(b)$	
2.	$d = 1 - by_0$	$q_0 = ay_0$
3.	$d_2 = dd$	$d_3 = dd + d$
4.	$y_1 = y_0 + y_0 d_3$	$d_5 = d_2 d_2 + d$
5.	$y_2 = y_0 + y_1 d_5$	$r_0 = a - bq_0$
6.	$e = 1 - by_2$	$q_1 = q_0 + r_0 y_2$
7.	$y_3 = y_2 + ey_2$	$r = a - bq_1$
8.	$q = q_1 + ry_3$	

All operations but the last are done in round-to-nearest.

optimized division algorithm

If we know that $a = 1$, we might substitute the following reciprocation algorithm:

1. $y_0 = \texttt{frcpa}(b)$
2. $d = 1 - by_0$
3. $d_2 = dd$ $\qquad d_3 = dd + d$
4. $y_1 = y_0 + y_0 d_3$ $\qquad d_5 = d_2 d_2 + d$
5. $y_2 = y_0 + y_1 d_5$
6. $e = 1 - by_2$
7. $y = y_2 + ey_2$

This has lower latency and uses fewer instructions. But will it yield correct rounding and flag settings?

error analysis

We can easily prove analytically that the relative error before the last rounding is bounded by $2^{-123.37} < 25/2^{2p}$.

Consequently we need to find $S_{24/2^{2p}}$, i.e. distribute factors of $2^{128} + \delta$ for $\delta \in \{-24, -23, \ldots, 23, 24\}$.

For round-to-nearest, there are 134 difficult cases close to a rounding boundary:

```
0xFFFFFFFFFFFFFFFF 0xFFFFFFFFFFFFFFFD 0xFE421D63446A3B34
0xFBFC17DFE0BEFF04 0xFB940B119826E598 0xFB0089D7241D10FC
0xFA0BF7D05FBE82FC 0xF912590F016D6D04 0xF774DD7F912E1F54
0xF7444DFBF7B20EAC 0xF39EB657E24734AC 0xF36EE790DE069D54
....
0x83AB6A090756D410 0x83AB6A06F8A92BF0 0x83A7B5D13DAE81B4
0x8365F2672F9341B4 0x8331C0CFE9341614 0x82A5F5692FAB4154
0x8140A05028140A04 0x8042251A9D6EF7FC
```

For directed rounding modes there are 220.

results

The algorithm works for all the round-to-nearest difficult cases and hence rounds correctly in round-to-nearest.

It works in directed rounding modes except for inputs with the following significands:

```
0x8C82DA588ADC6416 0x84FDF027EF813F7B 0x827B9B8059090AB2
0x8080402010080401 0x8000080000400001 0x8000000000000001
0x8000000000000000
```

This also implies that even in round-to-nearest, the inexact flag setting may be wrong for the case of exact powers of 2.

Reciprocal square root

extension to reciprocal square root

Essentially the same approach can be used to find hard cases for the reciprocal square root.

This time we need to find cases where $|m^2 b - 2^q|$ is small for $q \in \{3p, 3p+1\}$.

Potentially interesting since the best theoretical bounds for the relative distance from a midpoint are of order 2^{-3p} whereas one would expect something closer to 2^{-2p}.

Using this technique, we have proved some stronger bounds for $p = 64$.

However, for quad precision, the factorizations at last become very difficult. Possibly alternative factorization algorithms would do much better.

Conclusions

a brief summary

- Another addition to the existing results on difficult cases, with a distinctive approach using prime factorization
- Quite straightforward algorithm though dependent on good factoring software
- Theoretical analysis indicates it is feasible for some interesting problems
- Has been used for at least one real application
- A similar approach improves known bounds for reciprocal square root