Some arguments concerning correct rounding of the elementary functions

Jean-Michel Muller, Paul Zimmermann

Some arguments concerning correct rounding of the elementary functions

Jean-Michel Muller and Paul Zimmermann

August 17, 2006

1 why standardizing correct rounding ?

- to improve portability: for instance, when strict portability is needed, as was recently the case for the LHC@Home project. This project distributes a very large computation on a wide, heterogeneous network of computers, and requires strict floating-point determinism when checking the consistency of this distribution, due to the chaotic nature of the phenomenon being simulated. Default libraries on different systems would sometimes return slightly different results.

- with the directed roundings, for implementing interval arithmetic in a trustable yet accurate way: in round-to-nearest mode, correct rounding provides an accuracy improvement over usual `libm`s of only a fraction of a unit in the last place *(ulp)*, since the values difficult to round were close to the middle of two consecutive floating-point numbers. This may be felt of little practical significance. However, the three other rounding modes are needed to guarantee intervals in interval arithmetic. Without correct rounding in these directed rounding modes, interval arithmetic may loose up to two *ulp*s of precision in each computation. Actually, current interval elementary function libraries are even less accurate than that, because they sacrifice accuracy to a very strict proof;

- because – at least for some functions – **it can be done without a significant loss in performance.** See `http://lipforge.ens-lyon.fr/frs/download.php/58/crlibm-0.11beta1.pdf` as well as our forthcoming paper on the log function: `http://perso.ens-lyon.fr/jean-michel.muller/log.pdf`. Hence, anyway, **several libraries that provide correct rounding will be available in the near future**: it would be a pity to miss the opportunity of specifying at least what these functions should return in special cases.

- because we feel that a standard must show the goal and not merely acknowledge the existing.

2 Current status of CRLIBM

The functions that are provided, with correct rounding, in the current version[1] of CRLIBM so far are:

- exp, log, log2, log10, cosh, sinh, arcsin of all double precision numbers;
- sin in $[-\pi/2, +\pi/2]$, cos in $[-\pi/2, +\pi/2]$, tan in $[-1/8, +1/8]$ (this will quickly improve since we have the hardest to round cases between 0 and $\pi/2$), arctan in $[-\tan(1/8), +\tan(1/8)]$;

Will soon be provided (the worst cases for correct rounding have been computed, and the alpha versions of the programs are written and are being checked): $\exp(x) - 1$ and $\log(1 + x)$ for all double precision numbers. For example, the hardest to round case with exponent -4 of $\exp(x) - 1$ is

$$x = (1.1110100100100011110000011000100011101010011110011011)_2 \times 2^{-4}$$

with

$$e^x - 1 = (1.0000001111000101101001000010000010000010101111100111100\underbrace{0000\cdots000}_{56 \text{ zeros}}1111...)_2 \times 2^{-3}$$

Also, we are getting worst cases for the decimal64 format, as well as (this was not a big challenge) for the binary32 and decimal32 formats, and we now have hardest to round cases for the trigs in larger domains.

What is special about CRLIBM is the fact that we provide proofs of the functions we implement. To be able to generate more functions, more quickly, we are partially automating the process:

- using our gappa tool[2], we compute error bounds and generate formal proofs for these bounds. Examples can be found in the paper by Dinechin et al:
  `http://perso.ens-lyon.fr/guillaume.melquiond/doc/06-mcms-article.pdf`;
- we are developing methodologies for getting very good approximations under constraints[3]

A new technique suggested by Hanrot, Lefèvre and Zimmermann might soon allow us to find the hardest to round cases for the trig functions in the full range, and, therefore, to derive programs for these functions.

[1] see: `http://lipforge.ens-lyon.fr/frs/download.php/58/crlibm-0.11beta1.pdf`

[2] see: `http://lipforge.ens-lyon.fr/www/gappa/`

[3] see: `http://www.ens-lyon.fr/LIP/Pub/Rapports/DEA/DEA2006/DEA2006-03.ps.gz`, unfortunately in French for the moment.