WHEN A LOGARITHM IS A MISSPELLED ALGORITHM

Mioara Joldes

Résumé. — The high-quality floating-point implementation of elementary functions is a complex process. A variety of techniques ranging from numerical polynomial approximation algorithms to rigorous computation for bounding the approximation error are used. Firstly, we present a brief overview of this process. Then we show how rigorous computation methods based on multiprecision interval arithmetic and Taylor models are used for certified computation of tight bounds for supremum norms of approximation errors.

1. General research context

In many floating-point software systems there is a need for computing approximated values of mathematical functions such as exp, log, sin, arccos, or some compositions of them. These elementary functions are usually implemented in mathematical libraries called `libm`. Such libraries are available on most systems and many numerical programs depend on them. The word game, "A logarithm is just a misspelled algorithm", finds a proper signification in the general context of this work : the implementation and evaluation of an elementary function (like log) on a computer. Usually, this is reduced to a simple finite sequence of floating-point additions and multiplications. These operations are preferred because they are highly optimized in current hardware. A brief overview of the process of implementing elementary functions on a computer (in a floating-point environment) and of additional tools required for this process is presented in what follows. Starting early in the history of computing architectures, the floating-point format is used for approximate computation with real numbers. Before 1985, floating-point computations were very heterogeneous depending on the computer platform on which they were performed, which led to many numerical errors. The IEEE-754 standard adopted in 1985, defines binary floating-point formats : $\mathcal{F}_t = \left\{2^E \cdot m | E \in \mathbb{Z}, m \in \mathbb{Z}, 2^{t-1} \leq |m| \leq 2^t - 1\right\} \cup \{0\}$

of precision t. For example, with the required bounds on E, $\mathcal{F}_{24}$ is the single precision format. It also defines exact semantics for the basic operations ($+$, âĹŠ, $\times$, $\div$ and $\sqrt{}$). All these operations have to produce correctly rounded results, as if the operations were carried out in infinite precision and these intermediate results were then rounded. This contributed to a certain level of portability and provability of floating-point algorithms across standard-compliant platforms. However, most of real applications use also elementary functions. But until recently, there was no such requirement for these functions. The main impediment for this was the table maker's dilemma (TMD) [2], named in reference to the early builders of logarithm tables. This problem can be stated as follows : consider an elementary function f and a floating-point number x. Since floating-point numbers are rational numbers, in many cases, the image $y = f(x)$ is not a rational number, and therefore, cannot be represented exactly as a floating-point number. The correctly rounded result will be the floating-point number that is closest to this mathematical value. Using a finite precision environment (on a computer), only an approximation $\hat{y}$ to the real number y can be computed. If the accuracy used for computation is not enough, it is impossible to decide the correct rounding of $\hat{y}$. A technique published by Ziv [1] is to improve the accuracy of the approximation until the correctly rounded value can be decided. A first practical improvement over Ziv's approach derives from the availability of tight bounds on the worst-case accuracy required to compute many elementary functions, computed by Lefevre and Muller [2]. This improvement allowed for the possibility of writing a libm where the functions are correctly rounded and this is obtained at known and modest additional costs. This is one of the main purposes of the Arenaire team that develops the CRlibm project [3]. In the new standard IEEE754-2008, correct rounding for elementary functions is recommended. The participation of leading microprocessor manufacturers like Intel or AMD for this standard revision proves that the purpose of CRlibm was achieved : the requirement of correct rounding for elementary functions is compatible with industrial requirements and can be done for a modest additional cost compared to a classical libm. However, beside the TMD, the development and implementation of correctly rounded elementary functions is a complex process. A general scheme for this would include :

– Use the above mentioned methods of Muller and Lefevre [2] to obtain the necessary precision t in the worst-cases.

– Argument reduction for the function f to be considered : this involves the reduction of the problem to evaluating a function g over a small interval $[a, b]$. For this different ad-hoc methods are used on a case by case basis for each function.

– Find a polynomial approximation p_1 for g such that the maximum relative error between p_1 and g is small enough to allow for correct rounding in the general case. Find a polynomial approximation p_2 for g such that the maximum relative error between p_2 and g is small enough (less then 2^{-t}) to allow for correct rounding in the

worst-case. Obtaining good polynomial approximations and the certification of the approximation errors is detailed below.
– Write the code for evaluating p_1 and p_2 with the required accuracy. In this step round-off errors have to be taken into account for each multiplication and addition such that the total error stays below the required threshold.

2. Rigorous and tight bounding of approximation errors

As mentioned above, one key step in the implementation of elementary functions is polynomial approximation. This is preferred since polynomials can be evaluated completely based only on multiplications and additions. But since in general, polynomial operations are easier to use or implement, there are many other applications where it is useful to have a polynomial approximation to a function. In the same time, the approximation error between the function and the polynomial is very important since one must know that the approximation is sufficiently good for the respective application.

In general, since we want to minimize the number of operations needed for evaluation, we are interested in finding polynomial approximations for which, given a degree n, the maximum error between the function and the polynomial is minimum. The "minimax approximation" has been broadly developed in the literature and its application to elementary function implementation is discussed in detail in S. Chevillard's thesis [4]. Usually this approximant is computed numerically, using a version of Remez algorithm [4], so an a posteriori error bound is needed.

Obtaining a tight bound for the approximation error reduces to computing a tight bound for the supremum norm of the error function over the considered interval. This error function is given by $\varepsilon(x) = p(x)/f(x) - 1$ or $\varepsilon(x) = p(x) - f(x)$ depending on whether the relative or absolute error is considered. In other words, we are looking for a sufficiently tight interval $\mathbf{r}$, such that $\|\varepsilon\|_\infty \in \mathbf{r}$. Here, $\|\varepsilon\|_\infty$ denotes the supremum norm, defined by $\|\varepsilon\|_\infty = \sup_{x\in[a,\,b]}\{|\varepsilon(x)|\}$. The presented problem can be seen as a univariate rigorous global optimization problem, but it seems to present many issues unsuspected at a first sight [5]. In consequence, we present two rigorous computing techniques that we used for solving this problem : interval arithmetic and Taylor models. In what concerns reliability in computation with finite precision numbers and validation of the results obtained, one well established technique is interval arithmetic. In the framework of interval arithmetic, we define an interval $\boldsymbol{x}$ as a pair $\boldsymbol{x} = [\underline{x}, \overline{x}]$ consisting of two numbers $\underline{x}$ and $\overline{x}$ with $\underline{x} \leq \overline{x}$. A real number $x \in \mathbb{R}$ is *contained* in an interval $\boldsymbol{x}$, i.e., $x \in \boldsymbol{x}$, iff $\underline{x} \leq x \leq \overline{x}$. The elementary arithmetic operations, as well as the elementary functions can be straightforwardly extended to handle intervals. If the endpoints of an interval are not representable on a given computer, outward rounding, possibly in multiprecision arithmetic, is performed. The MPFI library [6] provides such a multiprecision interval arithmetic :

when performing an operation, the user chooses the precision used for representing the bounds of the result. The precision may be arbitrarily high.

One fundamental use of interval arithmetic is bounding the image of a function over an interval. However, it is very well known that interval calculations generally overestimate the image of a function and this phenomenon is in general proportional to the width of the interval. We are therefore interested in using thin intervals for obtaining a reasonably tight bounding of the function image.

While this method can be successfully used in general, when trying to solve our problem, one is faced with what is known in the literature of interval based methods under the name of "dependency phenomenon". Roughly speaking, since f and p are highly correlated, branch and bound methods based on using intervals of smaller width in order to obtain less overestimation, end up with an unreasonably high number of small intervals. Note that the difference between f and p is also highly cancellating, but this can be avoided using multiprecision interval arithmetic.

In order to reduce the dependency, Taylor models are used. They are a basic tool for replacing functions with a polynomial and an interval remainder bound, on which basic arithmetic operations or bounding methods are easier. When speaking about Taylor models, one usually considers a couple (p, R), where : p is the Taylor polynomial, and R is an interval bound for the Taylor remainder.

Thus, we have : $\forall x \in I, f(x) - p(x) \in R$. Arithmetic operations can be easily defined on such couples. For example, the sum of two Taylor models is obtained by adding the polynomials and the interval remainders respectively. For all others operations and compositions with elementary we refer to [7]. However, the available software implementations for Taylor models are scarce. The best known is COSY [7]. Although highly optimised and used, COSY has two major drawbacks for our specific problem. First, it does not provide multiple precision arithmetic, and thus fails to solve the cancellation problem mentioned. Second, it does not deal with the problem of functions with false singularities (that appear frequently in our case when considering relative errors), thus failing to provide a finite bound for the remainder for such functions.

3. Results and Conclusions

Multiprecision interval arithmetic and Taylor models are used in order to compute rigorous and tight bounds for approximation errors. The techniques presented above were implemented and used for example for one function needed in the code of `CRlibm`.

Références

[1] A. Ziv, *Fast evaluation of elementary mathematical functions with correctly rounded last bit*, ACM Transactions on Mathematical Software, 17(3) :410-423, 1991.
[2] V. Lefevre, J.-M. Muller, *Worst cases for correct rounding of the elementary functions in double precision*, 15th IEEE Symposium on Computer Arithmetic, Colorado, June 2001.
[3] CRLibm, *a library of correctly rounded elementary functions in double-precision*, http ://lipforge.ens-lyon.fr/www/crlibm/.
[4] S. Chevillard, *Évaluation efficace de fonctions numériques. Outils et exemples*, École Normale Supérieure de Lyon, Lyon, France, 2009.
[5] S. Chevillard, M. Joldes, C. Lauter *Certified and fast computation of supremum norms of approximation errors*, 19th IEEE SYMPOSIUM on Computer Arithmetic, Los Alamitos, CA, Portland, OR,169–176, 2009
[6] MPFI, *Multiple Precision Floating-Point Interval Library*, http ://gforge.inria.fr/projects/mpfi/.
[7] K. Makino and M. Berz, *Taylor Models and Other Validated Functional Inclusion Methods*, International Journal of Pure and Applied Mathematics, Vol. 4, 379–456, 2003.

Mioara Joldes

Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon,
46, Allée d'Italie 69364 Lyon Cedex 07.

E-mail : `mioara.joldes@ens-lyon.fr`