Validated Special Functions Software

Annie Cuyt, Franky Backeljauw, Stefan Becuwe, and Joris Van Deun

Department of Mathematics and Computer Science
University of Antwerp
Middelheimlaan 1, B-2020 Antwerp, Belgium
{annie.cuyt}@ua.ac.be

Abstract. Because of the importance of special functions, several books and a large collection of papers have been devoted to the numerical computation of these functions, the most well-known being the NBS handbook by Abramowitz and Stegun. But up to this date, symbolic and numeric environments offer no routines for the validated evaluation of special functions. We point out how a provable correct function evaluation can be returned efficiently.

1 Introduction

Functions that are termed special have explicitly known and simple representations as infinite/asymptotic series and/or continued fractions. Together the convergence domains of these representations often cover the full area of interest for users of these functions. Hence they lend themselves easily for a variable precision implementation. While the use of series to approximate a function in numeric software is well established, that of continued fractions is far from traditional. In [1] we describe how a combination of both techniques leads to validated software. The accumulation of round-off errors is tracked and bounded above while the accumulation of truncation errors is subject to divide-and-conquer.

We assume to have at our disposal a scalable precision, IEEE 754-854 compliant, floating-point implementation of the basic operations, square root and remainder, comparisons, base and type conversions, at least in the rounding mode round-to-nearest. Such an implementation is characterized by four parameters: the internal base β, the precision p and the exponent range $[L, U]$. Here we aim at least at implementations for $\beta = 2$ with precisions $p \geq 53$, and at implementations for use with $\beta = 2^i$ or $\beta = 10^i$ where $i > 1$. The IEEE 754-854 standard was revised in 2008. For our toolkit to be widely applicable, we do not expect the available floating-point implementation to support more advanced features such as exactly rounded mixed precision fused operations (including the assignment operation). We do however assume that the base conversions (between decimal and base β) are exactly rounded.

The goal is to compute a special mathematical quantity such as $\exp(-x^2)$ or $\sqrt{\pi}$ or $\Gamma(1/x)$. We refer to this quantity as $Y = f(y_x)$, where y_x is the argument built from an exact argument x (in base β and precision p) passed by a user,

Published version: Lecture Notes in Computer Science, Volume 6327, Mathematical Software - ICMS 10, pages 32-34
The final publication is available at http://link.springer.com/chapter/10.1007/978-3-642-15582-6_6

and f is the mathematical expression which is to be evaluated in y_x to yield Y. Of course y_x and Y suffer several finite precision and truncation errors, which we now analyze.

2 Round-off error accumulation and control

We denote by $\circledast$ the exactly rounded (to the nearest, with appropriate tiebreaker) floating-point implementation of the basic operation $*$ in the chosen base β and precision p. For floating-point numbers x and y, following the IEEE standards and in the absence of overflow and underflow, the basic operations are carried out with a relative error of at most $u(p) := {}^1\!/_2\,\beta^{-p+1}$ which is also called half a unit-in-the-last-place in precision p:

$$x \circledast y = (x * y)(1+\delta), \quad |\delta| \le u(p), \quad * \in \{+,-,\times,\div\}.$$

The same holds for the square root, the remainder, the conversions between internal formats and the base conversions.

In order to compute a relative error bound for a sequence of operations, it is necessary to keep track of all these error terms. A basic result, given in [2, p. 63], says that if all $|\delta_i| \le u(p), \rho_i = \pm 1$ and $nu(p) < 1$, then

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1+\theta_n, \quad |\theta_n| \le \gamma_n(p) = \frac{nu(p)}{1-nu(p)}. \tag{1}$$

This result is very convenient, as it allows us to rewrite any number of products and quotients of factors $1+\delta_i$ in an error analysis. Note that the reverse does not hold, meaning that not any expression $1+\theta_n$ with θ_n bounded above in absolute value by $\gamma_n(p)$, can be rewritten as a product of n factors $(1+\delta_i)^{\rho_i}$.

Perturbations as in (1) appear in the error analysis of all compound expressions involving the basic operations, square root, remainder and conversions. The values θ_n and bounds $\gamma_n(p)$ keep track of the accumulation of the round-off errors involved.

3 Truncation error accumulation and control

Let $\tilde{Y}_i$ be an approximation of the mathematical quantity Y_i with a relative error ϵ_i,

$$\tilde{Y}_i = Y_i(1+\epsilon_i), \quad i = 1,\ldots,m.$$

Moreover, let the exact quantity Y be given in terms of the Y_i and approximated by the value $\tilde{Y}$, such that with $\sigma_i = \pm 1$,

$$\tilde{Y} = Y(1+\eta_m), \quad 1+\eta_m = \prod_{i=1}^{m}(1+\epsilon_i)^{\sigma_i}, \quad |\eta_m| \le \kappa_m(p).$$

Published version: Lecture Notes in Computer Science, Volume 6327, Mathematical Software - ICMS 10, pages 32-34
The final publication is available at http://link.springer.com/chapter/10.1007/978-3-642-15582-6_6

In [1] we show how to distribute the threshold $\kappa_m(p)$ over the individual truncation errors ϵ_i to guarantee that $|\eta_m| \leq \kappa_m(p)$. Usually, the imposed threshold $\kappa_m(p)$ for $|\eta_m|$ is a small multiple of the half unit-in-the-last-place $u(p)$. If

$$|\epsilon_i| \leq \mu_i \frac{\kappa_m(p)}{1+\kappa_m(p)}, \quad i = 1, \ldots, m, \quad \sum_{i=1}^{m} \mu_i = 1, \tag{2}$$

then

$$\prod_{i=1}^{m} |1+\epsilon_i|^{\sigma_i} \leq 1 + \kappa_m(p).$$

The weights $\mu_i, i = 1, \ldots, m$ are chosen depending on the difficulty with which the operands $\tilde{Y}_i$ in the expression for $\tilde{Y}$ are obtained.

4 Putting it all together

Our aim eventually is to deal with the general situation where

$$\tilde{Y} = Y(1+\eta_m)(1+\theta_n), \quad |(1+\eta_m)(1+\theta_n)| \leq 1 + 2u(p) = 1 + \beta^{-p+1}. \tag{3}$$

Here the floating-point round-off errors δ_i have accumulated in θ_n and all approximation errors of a different nature ϵ_i in η_m.

To achieve (3) all floating-point operations must be carried out in a (slightly larger) working precision $\hat{p}$ than the destination precision p for $\tilde{Y}$. Then the error θ_n is bounded above in absolute value by $\gamma_n(\hat{p})$ which is a fraction of $2u(p)$. In order to guarantee (3), the accumulated error η_m must be bounded above in absolute value by $(2u(p) - \gamma_n(\hat{p}))/(1 + \gamma_n(\hat{p}))$. This in turn leads to individual bounds

$$|\epsilon_i| \leq \mu_i \frac{2u(p) - \gamma_n(\hat{p})}{1 + 2u(p)}, \quad i = 1, \ldots, m.$$

In [1] this toolkit of ideas is illustrated for the computation of the error function and the complementary error function on the real line.

The collection of special functions that can be implemented reliably using this technique includes the incomplete gamma and Gamma functions, Dawson's integral, the error and complementary error function, the Fresnel integrals, the exponential integrals, the hypergeometric and confluent hypergeometric family, the Bessel functions and modified Bessel functions for integer and half integer argument.

References

1. Backeljauw, F., Becuwe, S., Cuyt, A., Van Deun, J.: Validated Evaluation of Special Mathematical Functions. Technical Report 2009-04, Universiteit Antwerpen (2009)
2. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. Second edn. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2002)

Published version: Lecture Notes in Computer Science, Volume 6327, Mathematical Software - ICMS 10, pages 32-34
The final publication is available at http://link.springer.com/chapter/10.1007/978-3-642-15582-6_6