Science of Computer Programming 90 (2014) 2–20

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Validated evaluation of special mathematical functions

Franky Backeljauw [a], Stefan Becuwe [a], Annie Cuyt [a,*], Joris Van Deun [a], Daniel W. Lozier [b]

[a] Universiteit Antwerpen, Department of Mathematics and Computer Science, Middelheimlaan 1, B-2020 Antwerpen, Belgium
[b] National Institute of Standards and Technology, 100 Bureau Drive, Stop 8910, Gaithersburg, MD 20899-8910, USA

ARTICLE INFO

Article history:
Received 1 October 2012
Received in revised form 16 March 2013
Accepted 20 March 2013
Available online 28 May 2013

Keywords:
Validated software
Special functions

ABSTRACT

Because of the importance of special functions, several books and a large collection of papers have been devoted to their use and computation, the most well-known being the Abramowitz and Stegun handbook (Abramowitz and Stegun, 1964) [1] and its successor (Olver et al. 0000) [2]. However, until now no environment offers routines for the provable correct multiprecision and radix-independent evaluation of these special functions. We point out how we make good use of series and limit-periodic continued fraction representations in a package that is being developed at the University of Antwerp. Our scalable precision technique is mainly based on the use of sharpened a priori truncation and round-off error upper bounds for real arguments. The implementation is validated in the sense that it returns a sharp interval enclosure for the requested function evaluation, at the same cost as the evaluation.

1. Introduction

Special functions are pervasive in all fields of science. The most well-known application areas are in physics, engineering, chemistry, computer science and statistics. Because of their importance, several books and websites and a large collection of papers have been devoted to these functions. Of the standard work on the subject, the *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables* edited by Milton Abramowitz and Irene Stegun [1] and published nearly 50 years ago, the American National Institute of Standards and Technology (NIST) claims to have sold over 700 000 copies (over 150 000 directly and more than four times that number through commercial publishers)! This old handbook became obsolete in 2010 when NIST released the online DLMF: *NIST Digital Library of Mathematical Functions* edited by Frank W.J. Olver, Daniel W. Lozier, Ronald F. Boisvert and Charles W. Clark [2]. The DLMF updates, completely rewrites, and greatly expands the material contained in the old handbook. Together with its simultaneously published print edition, the DLMF is receiving steadily increasing usage (measured by citations).

Due to their regular appearance in mathematical models of scientific problems, special functions are also pervasive in numerical computations. Consequently, there is no shortage of numerical routines for evaluating many of the special functions in widely used mathematical software packages, systems and libraries such as Maple, Mathematica, MATLAB, IMSL, CERN and NAG. However, until now none of these contains *reliable*, or *validated* routines. In this paper, a routine must do more than just compute an accurate approximation. In addition to this, it must provide a *guaranteed bound* on the error of the computed numerical value. In the case of a real-valued function, this error bound determines an interval within which the exact function value is guaranteed to lie.

* Corresponding author. Tel.: +32 32653898; fax: +32 32653777.
E-mail addresses: franky.backeljauw@ua.ac.be (F. Backeljauw), stefan.becuwe@ua.ac.be (S. Becuwe), annie.cuyt@ua.ac.be (A. Cuyt), daniel.lozier@nist.gov (D.W. Lozier).

0167-6423/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.scico.2013.05.006

Constructing algorithms that are accurate and stable, even without guaranteeing bounds on errors, is difficult enough. The following quotes from 1998 and 2002 are still relevant today:

"Algorithms with strict bounds on truncation and rounding errors are not generally available for special functions. These obstacles provide an opportunity for creative mathematicians and computer scientists."

[Dan Lozier, General Editor of the DLMF project [3]]

"The decisions that go into these algorithm designs – the choice of reduction formulae and interval, the nature and derivation of the approximations – involve skills that few have mastered. The algorithms that MATLAB uses for gamma functions, Bessel functions, error functions, Airy functions, and the like are based on Fortran codes written 20 or 30 years ago."

[Cleve Moler, Founder of MATLAB [4]]

When we compute, we have come to expect absolute reliability of arithmetic operations, input–output procedures, and certain very elementary functions such as the square root, and we count upon the results of these operations, procedures and functions to lie strictly within certain bounds. Our confidence in these matters is the result of the current IEEE floating-point standard [5].

Many books and papers have been published addressing the need to extend the scope of absolute reliability to more areas of scientific computing. Examination of these sources indicates that attention until now has been directed mainly to processes involving finite sequences of arithmetic operations such as those arising in matrix computations and standard numerical methods for solving polynomials, differential equations, optimization problems, and such. However, elementary functions are barely covered, and special functions even less. On the whole, not enough attention is paid to the reliable evaluation of functions, with the exception of [6] and the coverage of multiprecision special and elementary functions in MPFR [7].

As it stands now, the DLMF puts a wealth of mathematical information at the fingertips of scientists who use special functions. No provision is made for a computational component. Abramowitz and Stegun included voluminous numerical tables that are omitted in the DLMF in favor of references to existing mathematical software. We feel the addition of a built-in facility for computing function values on the fly would be found useful, for example by software developers. Therefore, the University of Antwerp authors have begun an active collaboration with NIST to leverage the DLMF as a way to bring the value of reliable computing to the attention of a wider audience. To this end we are developing a validated, multiprecision and radix-independent library of special (and elementary) functions.

Our goal in this paper is to present our approach to the development of general algorithms that are applicable to the reliable arbitrary-precision computation of functions. Convergent and asymptotic series expansions, differential equations, recurrence relations, continued fractions and numerical quadrature are applicable approaches that have been employed successfully in a variety of multiple-precision packages such as [8–10,7]. Among these we have chosen to use convergent power series, for which truncation errors are well understood and sharply bounded, and continued fractions, for which error-theoretic improvements in recent years [11,12] have led to similarly sharp bounds. Together the convergence domains of these representations often cover the full area of interest for users of these functions. In the following sections we describe how a combination of both techniques, with the new results from Sections 3, 6–9, leads to validated special function software. The current implementation builds on a complete a priori error analysis, in contrast to the partial implementation described in [13], which makes use of a running error analysis (also see Section 8).

In Table 1 we indicate which functions and argument ranges (on the real line) are covered by our implementation. For the definition of the special functions we refer to [14]. A similar implementation in the complex plane is the subject of future research.

2. Round-off error accumulation

Let us assume to have at our disposal a scalable precision, IEEE 754-854 compliant [5], floating-point implementation of the basic operations, comparisons, base and type conversions, in the rounding modes upward, downward, truncation and round-to-nearest. Such an implementation is characterized by four parameters: the internal base β, the precision p and the exponent range $[L, U]$. The IEEE 754-854 standard was revised in 2008 [15] but most of this work was carried out in line with the floating-point standard that was valid for the last 20 years. For a concise analysis of the differences we refer to [16].

Here we aim at least at implementations for $\beta = 2$ with precisions $p \geqslant 53$, and at implementations for use with $\beta = 2^i$ or $\beta = 10^i$ where $i > 1$. Our internal exponent range $[L, U]$ is determined by the largest available integer hardware format (when this turns out to be insufficient, and underflow or overflow results are being generated, then an error message is returned because the subsequent error analysis breaks down).

Rounding a value to the nearest base β precision p floating-point number (with tie break even when $\beta/2$ is odd and odd when $\beta/2$ is even) is indicated by the operation $\bigcirc_p(\cdot)$. Further we denote by $\oplus$, $\ominus$, $\otimes$, $\oslash$ the exactly rounded (to the nearest, with appropriate tie break) floating-point implementation of the basic operations $+$, $-$, $\times$, $\div$ in the chosen base β and precision p. Sometimes the generic notation $\circledast$ is used where $* \in \{+, -, \times, \div\}$. Hence, for floating-point numbers x