ISO/IEC JTC 1/SC 22/WG14 N1244

Date: 2007-05-31

Rationale

for

Mathematical Special Functions

1 General [tr.intro]

1 The choice of functions in this Technical Report is based heavily on ISO 31-11:1992, part 14 (special functions) [5]. The goal was to provide C bindings for a collection of functions widely used in the sciences, but perhaps not of wide enough interest to be included in the Standard C library proper. ISO 31 seemed like an excellent candidate for such a corpus of functions.

1.1 Omitted Functions [tr.omitted]

1 A few of these functions are omitted:

— The error function is omitted because it has already been incorporated into the C Standard as of 1999 [6], as `erf(x)`.

— The gamma function is omitted because it has already been incorporated into the C Standard as of 1999, as `tgamma(x)`.

— The hypergeometric and confluent hypergeometric functions are omitted because they proved to have poor numeric properties when represented in typical floating-point formats. Specifically, small changes in arguments to these functions cause large changes in the corresponding function values over most of the ranges of valid argument values. (They are very *sensitive*.) Aside from the difficulty of implementing these functions, such sensitivity bodes ill for anybody who hopes to perform meaningful calculations involving them. This observation was reinforced by a dearth of reports on prior successes in numerical computing with hypergeometric functions.

1.2 Deviations from ISO 31 [tr.deviates]

1 This Technical Report deviates from ISO 31 in other small ways:

— The functions are defined only where they accept real values and produce real results.

— The order of function arguments does not always follow widespread mathematical practice. Differences were introduced to make the notation more uniform across all the functions described in this Technical Report. (To a lesser extent, some arguments were put at the end if they were more likely to have default values in a related C++ [7] binding).

2 Mathematical special functions [tr.num.sf]

2.1 Additions to header `<math.h>` [tr.num.sh.math]

1 The reference to NaN (Not a Number) codes suggests the need for IEC 60559 (a.k.a. IEEE 754) floating-point arithmetic, but that is not necessarily the case. It is certainly helpful in representing non-finite values if the floating-point types can represent some form of NaN as well as signed infinities.

2.1.1 associated Laguerre polynomials [tr.num.sf.Lnm]

1 The associated Laguerre polynomials are well defined for all `m` and `n`; hence the ISO 31 [5] restriction `m <= n` is omitted. On the other hand, adding the restriction `x >= 0` simplifies implementation at no apparent hardship to programmers.

2 These functions become numerically very sensitive for large m. Hence an implementation is at liberty to produce a poor approximation `for n >= 128 or m >= 128`, and programmers are discouraged from relying on function values for such large values of `n` and `m`.

3 Adding the restriction `|x| >= 0` simplifies implementation at no apparent hardship to programmers.

2.1.2 associated Legendre polynomials [tr.num.sf.Plm]

1 The associated Legendre polynomials are well defined for all `m` and `n`; hence the ISO 31 restriction `m <= n` is omitted. On the other hand, adding the restriction `|x| <= 1` simplifies implementation at no apparent hardship to programmers.

2 These functions become numerically very sensitive for large l. Hence an implementation is at liberty to produce a poor approximation for `l >= 128`, and programmers are discouraged from relying on function values for such large values of l.

3 A factor of $(-1)^m$ is sometimes included in the definition of this function. It is omitted here just as in ISO 31.

2.1.3 beta function [tr.num.sf.beta]

1 The beta function becomes numerically sensitive for argument values below about -5 to -10. For simplicity, the restriction is retained from ISO 31 as `x > 0 and y > 0`.

2.1.4 (complete) elliptic integral of the first kind [tr.num.sf.ellK]

1 The restriction $0 \leq k \leq 1$ on all of the complete elliptic integrals is broadened to include the value 1 because most

modern floating-point formats have a representation for infinity.

2.1.5 (complete) elliptic integral of the second kind [tr.num.sf.ellEx]

1 See 2.1.4.

2.1.6 (complete) elliptic integral of the third kind [tr.num.sf.ellPx]

1 See 2.1.4.

2.1.7 regular modified cylindrical Bessel functions [tr.num.sf.I]

1 The regular modified cylindrical Bessel function is also known as the hyperbolic Bessel function and the modified Bessel function of the first kind.

2 Adding the restriction `|x| >= 0` simplifies implementation at no apparent hardship to programmers.

2.1.8 cylindrical Bessel functions (of the first kind) [tr.num.sf.J]

1 For the cylindrical Bessel function of the first kind, adding the restriction `|x| >= 0` simplifies implementation at no apparent hardship to programmers.

2.1.9 irregular modified cylindrical Bessel functions [tr.num.sf.K]

1 The irregular modified cylindrical Bessel function is also known as the Basset function, the modified Bessel function of the third kind, and MacDonals function, and the Modified Hankel function.

2 Adding the restriction `|x| >= 0` simplifies implementation at no apparent hardship to programmers.

2.1.10 cylindrical Neumann functions [tr.num.sf.N]

1 For the cylindrical Neumann function of the first kind, adding the restriction `|x| >= 0` simplifies implementation at no apparent hardship to programmers.

2.1.11 (incomplete) elliptic integral of the first kind [tr.num.sf.ellF]

1 The restriction $0 \leq k \leq 1$ on all of the incomplete elliptic integrals is broadened to include the value 1 because most modern floating-point formats have a representation for infinity.

2.1.12 (incomplete) elliptic integral of the second kind [tr.num.sf.ellE]

1 See 2.1.11.

2.1.13 (incomplete) elliptic integral of the third kind [tr.num.sf.ellP]

1 The incomplete elliptic integral of the third kind has the term

$$\frac{1}{(1+n\sin^2(\theta))}$$

in ISO 31 [5]. That term was changed to

$$\frac{1}{(1-n\sin^2(\theta))}$$

in this Technical Report to bring it more in line with current mathematical practice. (There is considerable variation in the literature for this function.)

2 See 2.1.11.

2.1.14 exponential integral [tr.num.sf.ei]

2.1.15 Hermite polynomials [tr.num.sf.Hn]

2.1.16 Laguerre polynomials [tr.num.sf.Ln]

1 For the Laguerre polynomials, adding the restriction `|x| >= 0` simplifies implementation at no apparent hardship to programmers.

2 These functions become numerically very sensitive for large `n`. Hence an implementation is at liberty to produce a poor approximation for `n >= 128`, and programmers are discouraged from relying on function values for such large values of `n`.

2.1.17 Legendre polynomials [tr.num.sf.Pl]

1 For the Legendre polynomials, adding the restriction `|x| <= 1` simplifies implementation at no apparent hardship to programmers.

2 These functions become numerically very sensitive for large l. Hence an implementation is at liberty to produce a poor approximation for `l >= 128`, and programmers are discouraged from relying on function values for such large values of l.

2.1.18 Riemann zeta function [tr.num.sf.riemannzeta]

1 ISO 31 [5] defines the Riemann zeta function only for `x > 1`, but it is well defined for all `x` (except `x == 1`) and no harder to compute than the gamma function. Thus, this Technical Report extended the range to all `x`.

2.1.19 spherical Bessel functions (of the first kind) [tr.num.sf.j]

1 The spherical Bessel functions of the first kind become numerically very sensitive for large `n`. Hence an implementation is at liberty to produce a poor approximation for `n >= 128`, and programmers are discouraged from relying on function values for such large values of `n`.

2.1.20 spherical associated Legendre functions [tr.num.sf.Ylm]

1 The spherical associated Legendre function is defined with the ϕ dependence factored out to avoid any need to return non-real values.

2 These functions become numerically very sensitive for large l. Hence an implementation is at liberty to produce a poor approximation for `l >= 128`, and programmers are discouraged from relying on function values for such large values of l.

Bibliography

[1] International Standards Organization: *Technical Report 1 on C++ Library Extensions*. International Standard ISO/IEC TR 19768:2006.

[2] Milton Abramowitz and Irene A. Stegun (eds.): *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, volume 55 of National Bureau of Standards Applied Mathematics Series. U. S. Government Printing Office, Washington, DC: 1964. Reprinted with corrections, Dover Publications: 1972.

[3] Walter Brown, "A Proposal to Add Mathematical Special Functions to the C++ Standard Library," WG21 Document N1422 = 03-0004, 2003.

[4] P. J. Plauger, "Proposed Special Math functions." WG14 Document N1023, 2003.

[5] International Standards Organization: *Quantities and units, Third edition*. International Standard ISO 31-11:1992. ISBN 92-67-10185-4.

[6] International Standards Organization: *Programming Languages – C, Second edition*. International Standard ISO/IEC 9899:1999.

[7] International Standards Organization: *Programming Languages – C++*. International Standard ISO/IEC 14882:2003.

[8] William H. Press, et al;: *Numerical Recipes in C : The Art of Scientific Computing*. Second Edition ISBN 0-521-43108-5

[9] Jerome Spanier and Keith B. Oldman: *An Atlas of Functions*, Hemisphere Publishing Corp., 1987.

Index