Algorithm 723
Fresnel Integrals

W. VAN SNYDER
Jet Propulsion Laboratory, California Institute of Technology

An implementation of approximations for Fresnel integrals and associated functions is described. The approximations were originally developed by W. J. Cody, but a Fortran implementation using them has not previously been published.

Categories and Subject Descriptors: G.1.2 [**Numerical Analysis**]: Approximation—*rational approximation*; G.4 [**Mathematics of Computing**]: Mathematical Software—*certification and testing*

General Terms: Algorithms

Additional Key Words and Phrases: Special functions

Cody [1968] describes approximations[1] for computing the real Fresnel integrals

$$C(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right) dt$$

$$S(x) = \int_0^x \sin\left(\frac{\pi}{2}t^2\right) dt$$

and the associated functions

$$f(x) = [1/2 - S(x)]\cos\left(\frac{\pi}{2}x^2\right) - [1/2 - C(x)]\sin\left(\frac{\pi}{2}x^2\right)$$

$$g(x) = [1/2 - C(x)]\cos\left(\frac{\pi}{2}x^2\right) + [1/2 - S(x)]\sin\left(\frac{\pi}{2}x^2\right)$$

as defined by equations 7.3.1, 7.3.2, 7.3.5, and 7.3.6 in [Abramowitz and Stegun, 1966].

Fortran implementations of algorithms based on Cody's approximations have not previously been published, but an Algol 60 procedure has appeared

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.
Author's address: Jet Propulsion Laboratory, 4800 Oak Grove Drive, Mail Stop 601-108, Pasadena, CA 91109.

[1]There is an error in the microfiche supplement, in Table VIIIC: Where Cody wrote $x^{-4}/\Sigma_s p_s x^{-4s} \Sigma_s q_s x^{-4s}$, he evidently meant to write $x^{-4}\Sigma_s p_s x^{-4s}/\Sigma_s q_s x^{-4s}$.

in Hemker [1981]. A Fortran algorithm by Morris [1990], based on approximations by Hershey [1971], yields only 12 decimal digit relative accuracy. Cody dismisses approximations by Németh [1965] because they "converge painfully slowly." Later work by Bulirsch [1967], partly based on the work of Németh, is more efficient, but Bulirsch gives only 15 significant digits in the coefficients, while Cody gives 21 digits in some cases, and some of the approximations are accurate to nearly 19S.

ORGANIZATION OF THE ALGORITHM

In the organization of the algorithm, we adopted one principle from the discipline of object-oriented programming, namely, an *object* may have an internal state, and one uses *methods* to affect and observe that state. The Fortran function program unit that constitutes the algorithm may be considered to be an *object*.[2] There are four entries, each of which evaluates one of the four functions described above. Each entry may be considered to be a *method* associated to the object.

Cody provided approximations for $C(x)$ and $S(x)$ for $|x| \leq 1.6$, and for $f(x)$ and $g(x)$ for $x > 1.6$. If $C(x)$ or $S(x)$ is needed for $x > 1.6$ the algorithm evaluates $f(x)$ and $g(x)$ using Cody's approximations, and computes $C(x)$ or $S(x)$ as appropriate from $f(x)$ and $g(x)$. Similarly, if $f(x)$ or $g(x)$ is needed for $|x| \leq 1.6$ the algorithm evaluates $C(x)$ and $S(x)$ using Cody's approximations, and computes $f(x)$ or $g(x)$ as appropriate from $C(x)$ and $S(x)$. In some applications, one may need $C(x)$ and $S(x)$ for the same value of $x > 1.6$. To avoid recomputing $f(x)$ and $g(x)$ at the same value of x as used on the previous invocation, the algorithm retains an internal state consisting of the last value of x at which any of the four functions was evaluated, the last value computed for any of the four functions, and four Fortran LOGICAL variables that indicate whether the value stored for a function corresponds to the stored value of x. By analogy with object-oriented programming, this internal state may be considered to be represented by *instance variables*.

There are no a priori restrictions on the range of arguments. The algorithm will evaluate the desired function without complaint no matter what the argument. For $x < 0$, the algorithm uses $C(-x) = -C(x)$, $S(-x) = -S(x)$, $g(-x) = \cos(\pi/2x^2) + \sin(\pi/2x^2) - g(x)$ and $f(-x) = \cos(\pi/2x^2) - \sin(\pi/2x^2) - f(x)$. The accuracy of trigonometric functions decreases for large $|x|$ in most computing environments, so one should not expect values of $C(x)$ and $S(x)$ to be as accurate for large $|x|$ as for small $|x|$, nor should one expect $f(-x)$ or $g(-x)$ to be as accurate as $f(x)$ or $g(x)$ for large x. The algorithm assumes arguments are exact. That is, if the arguments were to have been presented to arbitrarily greater precision the additional digits would be zero. As a consequence, when $|x| > 2.0/\rho^{1/2}$, where ρ is the round off level (that is, the difference between 1.0 and the next larger representable number), we assume that $\pi/2x^2$ is an integer multiple of 2π, and therefore $\cos(\pi/2x^2) = 1.0$, and $\sin(\pi/2x^2) = 0.0$.

[2] Of course, in FORTRAN there can only be one instance of the object.

When formulating an application, one should if possible choose to use $C(x)$ and $S(x)$ when $|x| \le 1.6$, use $f(x)$ and $g(x)$ when $x > 1.6$, and avoid using $x < -1.6$.

In the text of the program, we provide all of the coefficients given in the microfiche supplement to Cody [1968]; all but the highest degree in each range are present in the form of comments. Each polynomial is evaluated by nested multiplication in a single statement (no loops are present). Thus, moderate but nontrivial editing would be required to specialize the procedure for higher performance at a lower degree of precision. The work required would, however, be substantially less than would be required if one had to enter the coefficients from the microfiche supplement to Cody [1968] manually.

In the microfiche supplement Cody [1968] gave complete details of the truncation error characteristics of the formulae. A summary of the relative truncation error of the highest-degree formulae, that is, those used in the algorithm in its published form, appears in the following table, in which ϵ denotes the relative truncation error. The slightly smaller accuracy of the approximations for $g(x)$ when $x > 1.9$ is not a substantial defect in applications in which $C(x)$ and $S(x)$ are used, because for $x > 1.9$, $f(x) > 10g(x)$. Thus when one computes $C(x)$ or $S(x)$ from $f(x)$ and $g(x)$ the relative error of $g(x)$ is only one tenth as large, relative to the desired function, as the relative error of $f(x)$.

Range	**Function**	$-\log_{10}\epsilon$	**Function**	$-\log_{10}\epsilon$
$\|x\| \le 1.2$	$C(x)$	16.24	$S(x)$	17.26
$1.2 < \|x\| \le 1.6$	$C(x)$	17.47	$S(x)$	18.66
$1.6 < \|x\| \le 1.9$	$f(x)$	17.13	$g(x)$	16.25
$1.9 < \|x\| \le 2.4$	$f(x)$	16.64	$g(x)$	15.65
$2.4 < \|x\|$	$f(x)$	16.89	$g(x)$	15.58

TESTING

We verified correct programming, and Cody's assertions regarding the accuracy of the approximations. For $0 \le x \le 2.4$, we divided each range over which Cody provided a distinct approximation into 200 equal subranges. For $x > 2.4$, for which Cody provides a single approximation, we divided the ranges $2.4 < x \le 6$, $6 < x \le 50$ and $10^{-3} < 1/x < 0.02$ into 200 equal subranges. We selected a point randomly in each subrange, and compared the functions for which Cody provides an approximation to a value computed by an extended-precision algorithm for the Fadeeva function $w(z)$ using relations 7.3.22 and 7.3.23 from Abramowitz and Stegun [1966]. Cody's approximations for $f(x)$ and $g(x)$ have the same asymptotic form as the functions when $x > 2.4$. The first two terms of Cody's approximation for $f(x)$ are exactly the same as the first two terms of the asymptotic expansion when $x > 1/(\rho\pi^5)^{1/9}$; for $g(x)$ they are exactly the same when $x > 1/(\rho\pi^6)^{1/11}$. These values of x are approximately 29 and 14, respectively, when using IEEE format double-precision arithmetic. Since the approximations become more accurate as x increases, one expects results as accurate as round-off error allows when x is sufficiently large.

The results of our testing for $0 \le x \le 1000$ are summarized below. In each interval we report the error in units of the last position of the test value under a column headed ULP, the error relative to the true value under a column headed REL, and the absolute error under a column headed ABS. Calculations were carried out using an IBM PC/AT, for which $\rho \simeq$ 2.22E-16 in double precision.

Function	Argument Interval	Mean ULP	Max ULP	Mean REL	Max REL	Mean ABS	Max ABS
$C(x)$	[0..1.2]	0.57 ρ	2.18 ρ	0.40 ρ	1.29 ρ	0.19 ρ	0.83 ρ
	(1.2..1.6]	0.70 ρ	2.52 ρ	0.48 ρ	1.55 ρ	0.19 ρ	0.63 ρ
$S(x)$	[0..1.2]	0.74 ρ	2.42 ρ	0.52 ρ	1.39 ρ	0.09 ρ	0.65 ρ
	(1.2..1.6]	0.75 ρ	2.20 ρ	0.55 ρ	1.55 ρ	0.38 ρ	1.10 ρ
$f(x)$	(1.6..1.9]	0.51 ρ	1.50 ρ	0.36 ρ	1.10 ρ	0.06 ρ	0.19 ρ
	(1.9..2.4]	0.30 ρ	0.96 ρ	0.26 ρ	0.77 ρ	0.04 ρ	0.12 ρ
	(2.4..6.0]	0.43 ρ	1.15 ρ	0.29 ρ	0.80 ρ	0.02 ρ	0.07 ρ
	(6.0..50.0]	0.45 ρ	1.05 ρ	0.31 ρ	0.71 ρ	0.00 ρ	0.03 ρ
	(50..1000]	0.39 ρ	1.02 ρ	0.27 ρ	0.72 ρ	0.00 ρ	4E-3 ρ
$g(x)$	(1.6..1.9]	0.53 ρ	1.92 ρ	0.38 ρ	1.11 ρ	0.01 ρ	0.02 ρ
	(1.9..2.4]	1.06 ρ	3.43 ρ	0.75 ρ	1.93 ρ	0.01 ρ	0.02 ρ
	(2.4..6.0]	1.51 ρ	4.04 ρ	1.01 ρ	2.61 ρ	0.00 ρ	0.01 ρ
	(6.0..50.0]	1.40 ρ	3.62 ρ	0.97 ρ	2.11 ρ	0.00 ρ	3E-4 ρ
	(50..1000]	1.09 ρ	2.40 ρ	0.75 ρ	1.50 ρ	0.00 ρ	2E-7 ρ

Cody's testing of the approximations, as described earlier and in [Cody, 1968], indicated a relative accuracy of 15 to 18 digits. Thus one should expect to achieve only slightly more accuracy, at most two or three digits, by carrying out the calculations using more precision, e.g., by using double-precision arithmetic on a Cray computer.

Cody noted the possibility of substantial cancellation in evaluating approximations for $C(x)$ and $S(x)$ over the interval $1.2 < |x| \le 1.6$. He suggested that the numerator and denominator polynomials should be transformed to the equivalent finite Chebyshev polynomial expansions, and the more expensive Clenshaw-Rice scheme described in Rice [1965] should be used for evaluation. We were unable to detect this difficulty, perhaps because our testing apparatus used full IEEE standard arithmetic, including guard digits.

We verified correct programming of $f(x)$ and $g(x)$ for $|x| \le 1.6$, and for $C(x)$ and $S(x)$ for $x > 1.6$, by comparing selected results to values published in Abramowitz and Stegun [1966]. Extensive accuracy testing would simply have validated the trigonometric function routines.

REFERENCES

ABRAMOWITZ, M., AND STEGUN, I. 1966. *Handbook of Mathematical Functions. Appl. Math. Ser. 55*, National Bureau of Standards.

BULIRSCH, R. 1967. Numerical calculation of the sine, cosine and Fresnel integrals. *Numer. Math. 9*, 380–385.

CODY, W. J. 1968. Chebyshev approximations for the Fresnel integrals. *Math. Comput. 22*, 450–453, plus microfiche supplement.

HEMKER, P. W., Ed. 1981. *NUMAL, Numerical Procedures in Algol 60* MC Syllabus 47.6, Mathematisch Centrum, Amsterdam.

HERSHEY, A. V. 1971. *Approximations of Functions by Sets of Poles*. Internal report TR-2564, Naval Weapons Laboratory, Dahlgren, Va.

MORRIS, A. H., JR. 1990. *NSWC Library of Mathematics Subroutines*. Internal report NSWC TR 90-21, Naval Surface Warfare Center, Dahlgren, Va., Jan.

NÉMETH, G. 1965. Chebyshev expansions for Fresnel integrals. *Numer. Math. 7*, 310–312.

RICE, J. R. [1965]. On the conditioning of polynomial and rational forms. *Numer. Math. 7*, 426–435.

Received February 1992; revised September 1992; accepted November 1992