Detailed documentation of the library for special mathematical functions by W. Fullerton

Arjen Markus

May 22, 2022

1 Introduction

The original code to the library can be found on `http://netlib.org`. It has been updated to the current Fortran standard:

- The code has been transformed to free-form.
- Elimination of `GOTO` and introduction of `IMPLICIT NONE`. In some cases calculations for particular cases has been moved to make the control flow easier.
- Error messages because of arguments out of range have been replaced by `NaN` of `Inf`. The original code would generally stop the program.
- Warnings because of loss of accuracy have been removed, as such messages, written to the screen, do not seem appropriate for a general library. It may be useful to add an auxiliary routine that checks the arguments.
- All code is now contained in a small set of modules, where the module `specfunc_fullerton` is the overall module that gives access to all public functions.

For the moment only the single-precision implementation has been updated.

2 Airy functions

The module `fullerton_airy` contains the routines for evaluating the Airy Ai and Bi functions and their first derivatives.

The Airy functions are solutions to the differential equation:

$$\frac{d^2y}{dx^2} - xy = 0 \tag{1}$$

The implementations of Airy Ai and Bi functions are:

```
y = airy_ai(x)
y = airy_bi(x)
```

The implementations of the first derivatives are:

```
y = airy_aiprime(x)
y = airy_biprime(x)
```

Notes:

- The function Ai decays rapidly. For large values of x the implementation returns 0.
- The function Bi grows rapidly. For large values of x the implementation returns `+Inf`. Similarly for its derivative.

3 Modified Bessel functions

The module `fullerton_bessel` implements the modified Bessel functions of the first and second kind, I_0, I_1, K_0, K_1 and I_n, K_n. The ordinary Bessel functions are part of the current Fortran standard.

```
y = bessel_i0(x)
y = bessel_i1(x)
y = bessel_k0(x)
y = bessel_k1(x)
y = bessel_in(n,x)
y = bessel_kn(n,x)
```

The argument x must be positive and n a non-negative integer.

Notes:

- The general modified Bessel functions with the integer parameter n have been implemented via well-known recursive relations.
- The functions I_0, I_1, I_n may overflow if the value of x is too large, `+Inf` is returned in that case.
- The functions K_0, K_1, K_n may underflow if the value of x is too large, `0` is returned in that case.

4 Beta function and related functions

The module `fullerton_beta` contains implementations for the beta function, the incomplete beta function and the logarithm of the beta function.

The beta function is defined as:

$$B(p,q) = \int_0^1 x^{p-1}(1-x)^{q-1}dx = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)} \tag{2}$$

where parameters $p, q > -1$

The incomplete beta function is defined as:

$$B(x;p,q) = \int_0^x u^{p-1}(1-u)^{q-1}du \tag{3}$$

The implementations of these functions and the logarithm are:

```
y = beta(p,q)
y = beta_inc(x,p,q)
y = log_beta(p,q)
```

Notes:

- The implementation of the Beta function requires a and b to be both positive. This is a limitation with respect to the actual mathematical function.
- For very large values of a and b the implementation simply returns 0.

5 Exponential integrals

The module `fullerton_ei` can be used to evaluate a variety of indefinite integrals:

$$\begin{aligned}
E_n(x) &= \int_1^\infty \frac{e^{-xt}}{t^n}dt & (4)\\
Ei(x) &= -E_1(-x) & (5)\\
Ci(x) &= -\int_x^\infty \frac{\cos t}{t}dt & (6)\\
Si(x) &= -\int_0^x \frac{\sin t}{t}dt & (7)\\
Chi(x) &= \gamma + \ln x + \int_0^x \frac{\cosh t - 1}{t}dt & (8)\\
Shi(x) &= \int_0^x \frac{\sinh t - 1}{t}dt & (9)
\end{aligned}$$

where γ is the Euler-Mascheroni constant, 0.5772156649...

The exponential integrals Ei, E_1 and E_n are implemented as:

```
y = integral_ei(x)
y = integral_e1(x)
y = integral_en(n,x)
```

The cosine integral Ci, sine integral Si, hyperbolic cosine integral Chi and hyperbolic sine integral Shi are implemented as:

```
y = integral_ci(x)
y = integral_chi(x)
y = integral_si(x)
y = integral_shi(x)
```

Notes:

- For the logarithmic integral Li the argument x must be positive, otherwise `NaN` is returned.
- Also for $x \approx 1$ the result of Li has low precision.
- THe argument x must not be zero for the exponential integral E_1.
- The implementation of E_1 returns 0 for very large arguments.

6 Gamma function and related functions

The module `fullerton_gamma` implements the Gamma function and several related functions. Note that the Gamma function in this module is an alternative to the standard function and is used by among others the module `fullerton_beta` to ensure consistent results.

The implemented functions are:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \tag{10}$$

$$\gamma(a,x) = \int_0^x t^{a-1} e^{-t} dt \tag{11}$$

$$\Gamma(a,x) = \int_x^\infty t^{a-1} e^{-t} dt \tag{12}$$

$$\Psi(x) = \frac{d}{dx} \ln \Gamma(x) \tag{13}$$

$$\gamma^*(x) = \frac{x^{-a}}{\Gamma(x)} \int_0^x t^{a-1} e^{-t} dt \tag{14}$$

where Ψ is the digamma function and γ^* is the incomplete Tricomi gamma function (see, for instance, `https://www.cs.purdue.edu/homes/wxg/selected_works/section_02/155.pdf`.

In addition the module evaluates the logarithm of the Gamma function and the reciprocal. A direct evaluation of these expressions is more accurate than evaluating the Gamma function and then taking the logarithm or the reciprocal of that result.

The functions are implemented as:

```
y = fgamma(x)             ! alternative to the intrinsic function,
                          ! used internally
y = log_gamma(x)
y = inc_gamma(a,x)        ! gamma(a,x)
y = inc_gamma_compl(a,x)  ! Gamma(a,x), the complement
y = gamma_tricomi(x)
y = reciprocal_gamma(x)
y = digamma(x)
```

The subroutine `sign_log_gamma` evaluates the logarithm and the sign of the Gamma function:

```
call sign_log_gamma( x, algam, sgngam )
```

Notes:

- For the Gamma function and its logarithm overflow may occur if the argument is too large or is around zero or a negative integer. In that case `+Inf` is returned (regardless of the sign of the mathematical Gamma function in that neighbourhood).
- Also for x near a negative integer the result has low precision.
- For the incomplete Gamma functions, the value of x must be non-negative. For $x = 0$ and $a \leq 0$, these functions are not defined. In these cases `NaN` is returned.
- The Tricomi gamma function requires x to be non-negative.
- For the digamma function *psi* the result is inaccurate if x is close to zero or a negative integer (poles of $\Gamma(x)$).

7 Inverse exponential integrals

The module `fullerton_inv_ci` implements two inverse functions: the inverse cosine integral and the inverse hyperbolic cosine integral:

$$y = Ci^{-1}(x) \tag{15}$$
$$y = Chi^{-1}(x) \tag{16}$$

These mathematical functions are implemented as:

```
y = inverse_ci(x)
y = inverse_chi(x)
```

Notes:

- The implementation of $Chi^{-1}(x)$ presents a discontinuity around $x = 3$. This is quite unusual within this library. It needs to be further investigated.
- For very small values of x the implementations of Ci and Chi return 0, instead of an underflowing number.

8 Pochhammer symbol and related funcions

The module `fullerton_poch` implements the Pochhammer symbol and two related functions:

$$(a)_x = \frac{\Gamma(a+x)}{\Gamma(a)} \tag{17}$$

$$\frac{(a)_x - 1}{x} = \frac{\Gamma(a+x) - 1}{x\Gamma(a)} \tag{18}$$

$$n! = n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 2 \cdot 1 \tag{19}$$

The second function is meant for special situations where the argument x is much smaller than 1. The third function is the ordinary factorial, but as the value can exceed the range of ordinary integers, it is calculated as a real number, extending the range of the argument.

The functions are implemented as:

```
y = poch(a,x)
y = poch1(a,x)
y = fac(n)
```

Notes:

- For the factorial the argument must be zero or positive, otherwise `Nan` is returned.
- If the argument for the factorial is too large, `+Inf` is returned.
- For the Pochhammer symbol, the exceptions are fairly complicated:
  - If the sum $a + x$ is a negative integer, but a is not, the value is undefined.
  - For large values of x the result is not accurate, because the sum $a+x$ can not be calculated accurately.
  - This is also the case if a or $a + x$ are close to a negative integer.

9 Miscelleaneous functions

The module `fullerton_misc` defines as number of miscellaneous functions:

- The cubic root of a real number (including negative numbers)
- The functions $f(x) = ln(1 + x)$ and $g(x) = (e^x - 1)/x$ with the argument $|x| << 1$
- Dawson's integral:

$$D(x) = e^{-x^2} \int_0^y e^{y^2} dy \tag{20}$$

  and its first derivative $D'(x) = 1 - 2xD(x)$.

- Spence's function:

$$S(x) = -\int_0^x \ln|1-y|dy \tag{21}$$

  though the values as evaluated and the description on `https://mathworld.wolfram.com` are not in agreement. This is something to be examined.

These functions are implemented as:

```
y = cbrt(x)
y = lnrel(x)
y = exprel(x)
y = integral_dawson(x)
y = dawson_prime(x)
y = integral_spence(x)
```

Notes:

- For the implementation `lnrel` the argument must be larger than -1.
- The implementation `lnrel` will have low precision if the argument is too near -1.
- Also for $x \approx 1$ the result has low precision.

10 Complex versions

The module `fullerton_complex` offers the implementation of complex versions of several of the above functions:

- The beta function and the logarithm of the beta function
- The gamma function, reciprocal gamma function and the logarithm of the gamma function
- The digamma function
- Several miscellaneous functions, $\arctan 2(z,w)$, $\log_{10}(z)$, $(e^z-1)/z$, $\ln(1+z)$, $z^{1/3}$

The names of the implementations are the same as the real versions, notably the complex gamma function is implemented. as `fgamma(z0)`.