# A proposal to add special mathematical functions according to the ISO/IEC 80000-2:2009 standard

**Document number: N3494**
**Version: 1.0**
**Date: 2012-12-19**

Vincent Reverdy (vince.rev@gmail.com)
Laboratory Universe and Theories, Observatory of Paris,
5 place Jules Janssen, 92195 Meudon, France

**Abstract**

This proposal concerns the addition of new special mathematical functions to the current C++ standard library. It can be considered as a proposed amendment to the ISO/IEC 29124:2010: *Extensions to the C++ Library to support mathematical special functions* document taking into account the standardized special mathematical functions of ISO/IEC 80000-2:2009. The proposed functions have a wide range of use in scientific computing and numerical methods and would form a consistent set with the functions of the current standard.

## Contents

# I Motivation

While the scientific community who has been trained and who uses C++ becomes larger, some common mathematical tools are still lacking the standard library. Additions of C++11, particularly the advanced generation of random numbers, the standardization of `long long` integers, and some new mathematical functions are very valuable, but it is necessary to send a strong message to the scientific community so that it could realize that C++ is very well suited for its problematics. Standardization of special mathematical functions of the ISO/IEC 29124:2010 is an important step, and adapting it to the new special functions of ISO/IEC 80000-2:2009 would extend further its capacity and provide a wide range of mathematical functions commonly used by scientific and HPC communities.

This proposal summarizes the special functions of ISO/IEC 80000-2:2009 and some related functions that could be added to the ISO/IEC 29124:2010 to form a consistent set of mathematical tools for scientists. Most of these functions are well known in some mathematical oriented languages. The structure of this proposal is exactly the same as ISO/IEC 29124:2010, providing `double`, `float` and `long double` versions of the proposed functions.

There is no particular difficulty of implementation: a large part of functions are based on existing C++ functions, and the proposed polynomials can be defined recursively.

# II Impact on the standard

This proposal is a pure addition to the existing library and consequently it would not affect existing programs. Nevertheless, as ISO/IEC 29124:2010, it requires modifications of the headers listed in table 1.

Unless otherwise specified, all components described in this proposal would be declared in namespace `std`. Furthermore, unless otherwise specified, all references to components described

**Table 1:** Summary of affected headers

| Subclause | Header(s) |
|---|---|
| IV.1 | `<cmath>` |
| IV.2 | `<math.h>` |
| IV.3 | `<ctgmath>` |
| IV.4 | `<tgmath>` |

in the C++ standard library are assumed to be qualified with `std::`.

# III Design decisions

The design would be the same as ISO/IEC 29124:2010 with `double`, `float` and `long double` versions for each function. The following tables summarize the choice of functions.

Table 2 sums up all the standardized functions of the *Special function* chapter of the ISO/IEC 800000-2:2009 document that are not addressed by ISO/IEC 29124:2010.

By taking into account the fact that the C++ special mathematical functions are only defined for integral and real values and do not define constants, the remaining candidates for a proposal are listed in table 3.

As an extension to the previous list, functions and polynomials well defined and closely related to the special functions of ISO/IEC 800000-2:2009 are listed in table 4 and 5.

All the resulting proposed functions are described in section IV.

**Table 2:** Special functions of ISO/IEC 800000-2:2009 not currently addressed by ISO/IEC 29124:2010.

| Name | Symbol | Expression with $x, \theta, \varphi \in \mathbb{R}$ and $a, b, c, z, w, \nu \in \mathbb{C}$ and $k, l, m, n \in \mathbb{N}$ | Special remarks |
|---|---|---|---|
| Euler constant | $\gamma$ | $= \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - \ln n\right)$ <br> $= 0.5772156\ldots$ | Constant. |
| Logarithmic integral | $\mathrm{li}\,(x)$ | $= \begin{cases} \int_0^x \frac{1}{\ln t}dt, & \text{if } 0 < x < 1 \\ \int_0^x \frac{1}{\ln t}dt, & \text{if } 1 < x \end{cases}$ | |
| Sine integral | $\mathrm{Si}\,(z)$ | $= \int_0^z \frac{\sin t}{t}dt$ | Other trigonometric integral functions are not included in ISO/IEC 80000. |
| Fresnel integrals | $\mathrm{S}\,(z)$ <br> $\mathrm{C}\,(z)$ | $= \int_0^z \sin\left(\frac{\pi}{2}t^2\right) dt$ <br> $= \int_0^z \cos\left(\frac{\pi}{2}t^2\right) dt$ | An alternative definition without the $\frac{\pi}{2}$ factor also exists. |
| Hypergeometric functions | $\mathrm{F}\,(a, b; c; z)$ | $= \sum_{n=0}^{\infty}\frac{(a)_n (b)_n z^n}{(c)_n n!}, -c \notin \mathbb{N}$ | Also named Gaussian or ordinary hypergeometric ${}_2\mathrm{F}_1$ functions. |
| Confluent hypergeometric functions | $\mathrm{F}\,(a; c; z)$ | $= \sum_{n=0}^{\infty}\frac{(a)_n z^n}{(c)_n n!}, -c \notin \mathbb{N}$ | Also named Kummer hypergeometric ${}_1\mathrm{F}_1$ functions. |
| Spherical harmonics | $\mathrm{Y}_l^m\,(\theta, \varphi)$ | $= \left[\frac{(2l+1)}{4\pi}\frac{(l-\lvert m\rvert)!}{(l+\lvert m\rvert)!}\right]^{\frac{1}{2}}$ <br> $\times\, \mathrm{P}_l^{\lvert m\rvert}\,(\cos\theta)\, \mathrm{e}^{im\varphi}, \lvert m\rvert \leq l$ | In the general case, the result is a complex value. |
| Chebyshev polynomials of the first kind | $\mathrm{T}_n\,(z)$ | $= \cos\,(n \arccos z)$ | This expression limits the definition domain of the polynomial. |
| Chebyshev polynomials of the second kind | $\mathrm{U}_n\,(z)$ | $= \frac{\sin\,[(n+1)\arccos z]}{\sin\,(\arccos z)}$ | This expression limits the definition domain of the polynomial. |
| Hankel functions | $\mathrm{H}_\nu^{(1)}\,(z)$ <br> $\mathrm{H}_\nu^{(2)}\,(z)$ | $= \mathrm{J}_\nu\,(z) + i\mathrm{N}_\nu\,(z)$ <br> $= \mathrm{J}_\nu\,(z) - i\mathrm{N}_\nu\,(z)$ | In the general case, the result is a complex value. |
| Spherical Hankel functions | $\mathrm{h}_l^{(1)}\,(z)$ <br> $\mathrm{h}_l^{(2)}\,(z)$ | $= \mathrm{j}_l\,(z) + i\mathrm{n}_l\,(z)$ <br> $= \mathrm{j}_l\,(z) - i\mathrm{n}_l\,(z)$ | In the general case, the result is a complex value. |
| Airy functions | $\mathrm{Ai}\,(z)$ <br> $\mathrm{Bi}\,(z)$ | $= \frac{1}{3}\sqrt{z}\left[\mathrm{I}_{-\frac{1}{3}}\,(w) - \mathrm{I}_{\frac{1}{3}}\,(w)\right]$ <br> $= \frac{1}{3}\sqrt{z}\left[\mathrm{I}_{-\frac{1}{3}}\,(w) + \mathrm{I}_{\frac{1}{3}}\,(w)\right]$ <br> with $w = \frac{2}{3}z^{\frac{3}{2}}$ | |

**Table 3:** Special functions of ISO/IEC 800000-2:2009 that can be candidates for a standardization.

| Name | Symbol | Expression with $n \in \mathbb{N}$ and $x, y, a, b, c \in \mathbb{R}$ | Special remarks |
|---|---|---|---|
| Logarithmic integral | $\mathrm{li}\,(x)$ | $= \begin{cases} \int_0^x \frac{1}{\ln t} dt, & \text{if } 0 < x < 1 \\ \int_0^x \frac{1}{\ln t} dt, & \text{if } 1 < x \end{cases}$ | |
| Sine integral | $\mathrm{Si}\,(x)$ | $= \int_0^x \frac{\sin t}{t} dt$ | Other trigonometric integral functions are not included in ISO/IEC 29124:2010. |
| Fresnel integrals | $\mathrm{S}\,(x)$ <br> $\mathrm{C}\,(x)$ | $= \int_0^x \sin\left(\frac{\pi}{2} t^2\right) dt$ <br> $= \int_0^x \cos\left(\frac{\pi}{2} t^2\right) dt$ | An alternative definition without the $\frac{\pi}{2}$ factor also exists. |
| Hypergeometric functions | $\mathrm{F}\,(a, b; c; x)$ | $= \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!}, -c \notin \mathbb{N}$ | Also named Gaussian or ordinary hypergeometric ${}_2\mathrm{F}_1$ functions. |
| Confluent hypergeometric functions | $\mathrm{F}\,(a; c; x)$ | $= \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!}, -c \notin \mathbb{N}$ | Also named Kummer hypergeometric ${}_1\mathrm{F}_1$ functions. |
| Chebyshev polynomials of the first kind | $\mathrm{T}_n\,(x)$ | $= \cos\left(n \arccos x\right)$ | This expression limits the definition domain of the polynomial. |
| Chebyshev polynomials of the second kind | $\mathrm{U}_n\,(x)$ | $= \frac{\sin\left[(n+1) \arccos x\right]}{\sin\left(\arccos x\right)}$ | This expression limits the definition domain of the polynomial. |
| Airy functions | $\mathrm{Ai}\,(x)$ <br> $\mathrm{Bi}\,(x)$ | $= \frac{1}{3}\sqrt{x}\left[\mathrm{I}_{-\frac{1}{3}}(y) - \mathrm{I}_{\frac{1}{3}}(y)\right]$ <br> $= \frac{1}{3}\sqrt{x}\left[\mathrm{I}_{-\frac{1}{3}}(y) + \mathrm{I}_{\frac{1}{3}}(y)\right]$ <br> with $y = \frac{2}{3} x^{\frac{3}{2}}$ | |

**Table 4:** Functions closely related to special mathematical functions of ISO/IEC 800000-2:2009.

| Name | Symbol | Expression with $n \in \mathbb{N}$ and $x, s, u, k, \phi \in \mathbb{R}$ | Use and relation with special mathematical functions |
|---|---|---|---|
| Cardinal sine | $\operatorname{sinc}(x)$ | $= \frac{\sin x}{x}$ | Integrand of Si. Widely used in optics and signal processing. It is also equal to $\mathrm{j}_0$. |
| Trigonometric integrals | $\operatorname{Ci}(x)$<br>$\operatorname{Shi}(x)$<br>$\operatorname{Chi}(x)$ | $= \gamma + \ln x + \int_0^x \frac{\cos t - 1}{t} dt$<br>$= \int_0^x \frac{\sinh t}{t} dt$<br>$= \gamma + \ln x + \int_0^x \frac{\cosh t - 1}{t} dt$ | The trigonometric integrals are well-defined and would form a consistent set with the standardized Si function. |
| Incomplete gamma functions | $\Gamma(s, x)$<br>$\gamma(s, x)$ | $= \int_x^\infty t^{s-1} \mathrm{e}^{-t} dt$<br>$= \int_0^x t^{s-1} \mathrm{e}^{-t} dt$ | Closely related to hypergeometric functions and the generalization of Fresnel functions. |
| Pochhammer symbols | $x^{(n)}$<br>$(x)_n$ | $= \frac{\Gamma(x+n)}{\Gamma(x)}$<br>$= \frac{\Gamma(x+1)}{\Gamma(x-n+1)}$ | Closely related to the definition of hypergeometric functions. |
| Jacobi elliptic functions | $\operatorname{sn}(u;k)$<br>$\operatorname{cn}(u;k)$<br>$\operatorname{dn}(u;k)$ | $u = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}$<br>$\operatorname{sn}(u;k) = \sin \phi$<br>$\operatorname{cn}(u;k) = \cos \phi$<br>$\operatorname{dn}(u;k) = \sqrt{1 - k^2 \sin^2 \phi}$ | Closely related to the incomplete elliptic integral of the first kind. They are used in a lot of physical problems involving elliptic integrals. |

**Table 5:** Polynomials closely related to special mathematical functions of ISO/IEC 800000-2:2009.

| Name | Symbol | Expression with $n, m \in \mathbb{N}$ and $x, s, \alpha, \beta, \rho, \phi \in \mathbb{R}$ | Use and relation with special mathematical functions |
|---|---|---|---|
| Chebyshev polynomials | $\mathrm{T}_n(x)$<br>$\mathrm{U}_n(x)$<br>$\mathrm{V}_n(x)$<br>$\mathrm{W}_n(x)$ | $\mathrm{p}_n(x) = 2x\mathrm{p}_{n-1}(x) - \mathrm{p}_{n-2}(x)$<br>with $\mathrm{p}_0(x) = 1$ and:<br>for $\mathrm{p} = \mathrm{T}:\ \mathrm{p}_1(x) = x$<br>for $\mathrm{p} = \mathrm{U}:\ \mathrm{p}_1(x) = 2x$<br>for $\mathrm{p} = \mathrm{V}:\ \mathrm{p}_1(x) = 2x - 1$<br>for $\mathrm{p} = \mathrm{W}:\ \mathrm{p}_1(x) = 2x + 1$ | There are not 2 types of Chebyshev polynomials but 4 types, and they can be defined over $\mathbb{R}$ by a recursion formula as shown here. |
| Jacobi polynomials | $\mathrm{P}_n^{(\alpha,\beta)}(x)$ | $2(n+1)(c-n+1)c\mathrm{P}_n^{(\alpha,\beta)}(x)$<br>$= \left[(c+1)\left(\alpha^2-\beta^2\right) + (c)_3\,x\right]$<br>$\times\, \mathrm{P}_{n-1}^{(\alpha,\beta)}(x) - 2(n+\alpha)(n+\beta)$<br>$\times\,(c+2)\,\mathrm{P}_{n-2}^{(\alpha,\beta)}(x)$<br>with:<br>$c = 2n + \alpha + \beta$<br>$\mathrm{P}_0^{(\alpha,\beta)}(x) = 1$<br>$\mathrm{P}_1^{(\alpha,\beta)}(x) = (\alpha+1)$<br>$+\,\frac{1}{2}(\alpha+\beta+2)(x-1)$ | Jacobi polynomials are orthogonal polynomials over $[-1,1]$ and are a generalization of many orthogonal polynomials: Chebyshev, Legendre, Gegenbauer and Zernike polynomials. |
| Gegenbauer polynomials | $\mathrm{C}_n^{(\alpha)}(x)$ | $n\mathrm{C}_n^{(\alpha)}(x) = 2(n+\alpha-1)\,x\times$<br>$\mathrm{C}_{n-1}^{(\alpha)}(x) - (n+2\alpha-2)\times$<br>$\mathrm{C}_{n-2}^{(\alpha)}(x)$<br>with:<br>$\mathrm{C}_0^{(\alpha)}(x) = 1$<br>$\mathrm{C}_1^{(\alpha)}(x) = 2\alpha x$ | Gegenbauer polynomials are orthogonal polynomials over $[-1,1]$, are a generalization of Chebyshev and Legendre polynomials and are a special case of Jacobi polynomials. |
| Zernike polynomials | $\mathrm{Z}_n^m(\rho,\phi)$<br>$\mathrm{Z}_n^{-m}(\rho,\phi)$ | $= \mathrm{R}_n^m(\rho)\cos(m\phi)$<br>$= \mathrm{R}_n^m(\rho)\sin(m\phi)$ | Zernike polynomials are a special case of Jacobi polynomials. |
| Radial polynomials | $\mathrm{R}_n^m(\rho)$ | if $n-m$ is odd:<br>$\mathrm{R}_n^m(\rho) = 0$<br>if $n-m$ is even:<br>$\mathrm{R}_n^m(\rho) = \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k (n-k)!}{k!} \times$<br>$\frac{\rho^{n-2k}}{\left(\frac{n+m}{2}-k\right)! \times \left(\frac{n-m}{2}-k\right)!}$ | Radial polynomials are closely related to the Zernike polynomials. |

# IV Proposed extension of special mathematical functions

Important note: the definition intervals in the following section represents the definition intervals of the associated C++ functions and not the mathematical definition: for example ln is considered as being defined over $\mathbb{R}^+$ (C++ definition interval) and not over $\mathbb{R}^{+*}$ (mathematical definition interval) because `std::log(0)` is defined and is equal to `-inf`.

## IV.1 Additions to header `<cmath>`

Table 6 summarizes the proposed functions to be added to header `<cmath>`.

**Table 6:** Proposed additions to `cmath` synopsis

| **Functions:** | | |
|---|---|---|
| sinc | fresnel_s | hgeom_ordinary |
| logint | fresnel_c | chebyshev_t |
| sinint | airy_ai | chebyshev_u |
| cosint | airy_bi | chebyshev_v |
| sinhint | gamma_u | chebyshev_w |
| coshint | gamma_l | jacobi |
| jacobi_sn | pochhammer_u | gegenbauer |
| jacobi_cn | pochhammer_l | zernike |
| jacobi_dn | hgeom_confluent | radpoly |

Each of these functions would be provided for arguments of types `float`, `double`, and `long double`. The detailed signatures are:

// [IV.1.1] **cardinal sine:**
```
double sinc(double x);
float sincf(float x);
long double sincl(long double x);
```

// [IV.1.2] **logarithmic integral:**
```
double logint(double x);
float logintf(float x);
long double logintl(long double x);
```

// [IV.1.3] **sine integral:**
```
double sinint(double x);
float sinintf(float x);
long double sinintl(long double x);
```

// [IV.1.4] **cosine integral:**
```
double cosint(double x);
float cosintf(float x);
long double cosintl(long double x);
```

// [IV.1.5] **hyperbolic sine integral:**
```
double sinhint(double x);
float sinhintf(float x);
long double sinhintl(long double x);
```

// [IV.1.6] **hyperbolic cosine integral:**
```
double coshint(double x);
```

```
float coshintf(float x);
long double coshintl(long double x);
```

// [IV.1.7] **Jacobi elliptic sn function:**

```
double jacobi_sn(double k, double u);
float jacobi_snf(float k, float u);
long double jacobi_snl(long double k, long double u);
```

// [IV.1.8] **Jacobi elliptic cn function:**

```
double jacobi_cn(double k, double u);
float jacobi_cnf(float k, float u);
long double jacobi_cnl(long double k, long double u);
```

// [IV.1.9] **Jacobi elliptic dn function:**

```
double jacobi_dn(double k, double u);
float jacobi_dnf(float k, float u);
long double jacobi_dnl(long double k, long double u);
```

// [IV.1.10] **Fresnel sine integral:**

```
double fresnel_s(double x);
float fresnel_sf(float x);
long double fresnel_sl(long double x);
```

// [IV.1.11] **Fresnel cosine integral:**

```
double fresnel_c(double x);
float fresnel_cf(float x);
long double fresnel_cl(long double x);
```

// [IV.1.12] **Airy function of the first kind:**

```
double airy_ai(double x);
float airy_aif(float x);
long double airy_ail(long double x);
```

// [IV.1.13] **Airy function of the second kind:**

```
double airy_bi(double x);
float airy_bif(float x);
long double airy_bil(long double x);
```

// [IV.1.14] **upper incomplete gamma function:**

```
double gamma_u(double s, double x);
float gamma_uf(float s, float x);
long double gamma_ul(long double s, long double x);
```

// [IV.1.15] **lower incomplete gamma function:**

```
double gamma_l(double s, double x);
float gamma_lf(float s, float x);
long double gamma_ll(long double s, long double x);
```

// [IV.1.16] **upper Pochhammer symbol:**

```
double pochhammer_u(double n, double x);
float pochhammer_uf(float n, float x);
long double pochhammer_ul(long double n, long double x);
```

// [IV.1.17] **lower Pochhammer symbol:**

```
double pochhammer_l(double n, double x);
```

```
float pochhammer_lf(float n, float x);
long double pochhammer_ll(long double n, long double x);
```

// [IV.1.18] **confluent hypergeometric functions:**

```
double hgeom_confluent(double a, double c, double x);
float hgeom_confluentf(float a, float c, float x);
long double hgeom_confluentl(long double a, long double c, long double x);
```

// [IV.1.19] **ordinary hypergeometric functions:**

```
double hgeom_ordinary(double a, double b, double c, double x);
float hgeom_ordinaryf(float a, float b, float c, float x);
long double hgeom_ordinaryl(long double a, long double b, long double c, long
double x);
```

// [IV.1.20] **Chebyshev polynomials of the first kind:**

```
double chebyshev_t(unsigned n, double x);
float chebyshev_tf(unsigned n, float x);
long double chebyshev_tl(unsigned n, long double x);
```

// [IV.1.21] **Chebyshev polynomials of the second kind:**

```
double chebyshev_u(unsigned n, double x);
float chebyshev_uf(unsigned n, float x);
long double chebyshev_ul(unsigned n, long double x);
```

// [IV.1.22] **Chebyshev polynomials of the third kind:**

```
double chebyshev_v(unsigned n, double x);
float chebyshev_vf(unsigned n, float x);
long double chebyshev_vl(unsigned n, long double x);
```

// [IV.1.23] **Chebyshev polynomials of the fourth kind:**

```
double chebyshev_w(unsigned n, double x);
float chebyshev_wf(unsigned n, float x);
long double chebyshev_wl(unsigned n, long double x);
```

// [IV.1.24] **Jacobi polynomials:**

```
double jacobi(unsigned n, double alpha, double beta, double x);
float jacobif(unsigned n, float alpha, float beta, float x);
long double jacobil(unsigned n, long double alpha, long double beta, long double
x);
```

// [IV.1.25] **Gegenbauer polynomials:**

```
double gegenbauer(unsigned n, double alpha, double x);
float gegenbauerf(unsigned n, float alpha, float x);
long double gegenbauerl(unsigned n, long double alpha, long double x);
```

// [IV.1.26] **Zernike polynomials:**

```
double zernike(unsigned n, int m, double rho, double phi);
float zernikef(unsigned n, int m, float rho, float phi);
long double zernikel(unsigned n, int m, long double rho, long double phi);
```

// [IV.1.27] **Radial polynomials:**

```
double radpoly(unsigned n, unsigned m, double rho);
float radpolyf(unsigned n, unsigned m, float rho);
long double radpolyl(unsigned n, unsigned m, long double rho);
```

Each of the functions specified above that has one or more `double` parameters (the `double` version) would have two additional overloads:

1. a version with each `double` parameter replaced with a `float` parameter (the `float` version), and
2. a version with each `double` parameter replaced with a `long double` parameter (the `long double` version).

The return type of each such `float` version would be `float`, and the return type of each such `long double` version would be `long double`.

Moreover, each `double` version would have sufficient additional overloads to determine which of the above three versions to actually call, by the following set of rules:

1. First, if any argument corresponding to a `double` parameter in the `double` version has type `long double`, the `long double` version is called.
2. Otherwise, if any argument corresponding to a `double` parameter in the `double` version has type `double` or has an integer type, the `double` version is called.
3. Otherwise, the `float` version is called.

If any argument value to any of the functions specified above is a NaN (Not a Number), the function returns a NaN but it would not report a domain error. Otherwise, the function would report a domain error for just those argument values for which:

a) the function description's *Returns* clause explicitly specifies a domain and those argument values fall outside the specified domain, or

b) the corresponding mathematical function value has a non-zero imaginary component, or

c) the corresponding mathematical function is not mathematically defined.

Unless otherwise specified, each function is defined for all finite values, for negative infinity, and for positive infinity.

### IV.1.1 cardinal sine

```
double sinc(double x);
float sincf(float x);
long double sincl(long double x);
```

1 *Effects*: These functions compute the cardinal sine functions of their respective arguments `x`.

2 *Returns*: The `sinc` functions return

$$\operatorname{sinc}(x) = \begin{cases} \frac{\sin x}{x}, & \text{if } x \in \mathbb{R}^* \\ 1, & \text{if } x = 0 \end{cases}$$

3 *Status*: Integrand of the Si function defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (see IV.1.3).

### IV.1.2 logarithmic integral

```
double logint(double x);
float logintf(float x);
long double logintl(long double x);
```

1 *Effects*: These functions compute the logarithmic integrals of their respective arguments `x`.

2 *Returns*: The `logint` functions return

$$\mathrm{li}\,(x) = \begin{cases} \int_0^x \frac{1}{\ln t} dt, & \text{if } 0 \le x \in \mathbb{R} \le 1 \\ \int_0^x \frac{1}{\ln t} dt, & \text{if } 1 \le x \in \mathbb{R} \end{cases}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (same definition).

### IV.1.3 sine integral

```
double sinint(double x);
float sinintf(float x);
long double sinintl(long double x);
```

1 *Effects*: These functions compute the sine integrals of their respective arguments `x`.

2 *Returns*: The `sinint` functions return

$$\mathrm{Si}\,(x) = \int_0^x \frac{\sin t}{t} dt, \qquad \text{for } x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (same definition).

### IV.1.4 cosine integral

```
double cosint(double x);
float cosintf(float x);
long double cosintl(long double x);
```

1 *Effects*: These functions compute the cosine integrals of their respective arguments `x`.

2 *Returns*: The `cosint` functions return

$$\mathrm{Ci}\,(x) = \gamma + \ln x + \int_0^x \frac{\cos t - 1}{t} dt, \qquad \text{for } x \in \mathbb{R}^+$$

3 *Status*: Complementary trigonometric integral of the Si function defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (see IV.1.3).

### IV.1.5 hyperbolic sine integral

```
double sinhint(double x);
float sinhintf(float x);
long double sinhintl(long double x);
```

1 *Effects*: These functions compute the hyperbolic sine integrals of their respective arguments `x`.

2 *Returns*: The `sinhint` functions return

$$\mathrm{Shi}\,(x) = \int_0^x \frac{\sinh t}{t} dt, \qquad \text{for } x \in \mathbb{R}$$

3 *Status*: Complementary trigonometric integral of the Si function defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (see IV.1.3).

### IV.1.6 hyperbolic cosine integral

```
double coshint(double x);
float coshintf(float x);
long double coshintl(long double x);
```

1 *Effects*: These functions compute the hyperbolic cosine integrals of their respective arguments `x`.

2 *Returns*: The `coshint` functions return

$$\mathrm{Chi}\,(x) = \gamma + \ln x + \int_0^x \frac{\cosh t - 1}{t} dt, \qquad \text{for } x \in \mathbb{R}^+$$

3 *Status*: Complementary trigonometric integral of the Si function defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (see IV.1.3).

### IV.1.7 Jacobi elliptic sn function

```
double jacobi_sn(double k, double u);
float jacobi_snf(float k, float u);
long double jacobi_snl(long double k, long double u);
```

1 *Effects*: These functions compute the Jacobi elliptic sn functions of their respective arguments `k` and `u`.

2 *Returns*: The `jacobi_sn` functions return

$$\mathrm{sn}\,(k,u) = \sin\phi, \qquad \text{with } u = \mathrm{F}\,(k,\phi) = \int_0^\phi \frac{d\theta}{\sqrt{1-k^2\sin^2\theta}} \text{ and } |k| \in \mathbb{R} \le 1$$

3 *Status*: Inverse of the incomplete elliptic integral of the first kind defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard.

### IV.1.8 Jacobi elliptic cn function

```
double jacobi_cn(double k, double u);
float jacobi_cnf(float k, float u);
long double jacobi_cnl(long double k, long double u);
```

1 *Effects*: These functions compute the Jacobi elliptic cn functions of their respective arguments `k` and `u`.

2 *Returns*: The `jacobi_cn` functions return

$$\mathrm{cn}\,(k, u) = \cos\phi, \qquad \text{with } u = \mathrm{F}\,(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2\theta}} \text{ and } |k| \in \mathbb{R} \leq 1$$

3 *Status*: Inverse of the incomplete elliptic integral of the first kind defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard.

### IV.1.9 Jacobi elliptic dn function

```
double jacobi_dn(double k, double u);
float jacobi_dnf(float k, float u);
long double jacobi_dnl(long double k, long double u);
```

1 *Effects*: These functions compute the Jacobi elliptic dn functions of their respective arguments `k` and `u`.

2 *Returns*: The `jacobi_dn` functions return

$$\mathrm{dn}\,(k, u) = \sqrt{1 - k^2 \sin^2\phi}, \qquad \text{with } u = \mathrm{F}\,(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2\theta}} \text{ and } |k| \in \mathbb{R} \leq 1$$

3 *Status*: Inverse of the incomplete elliptic integral of the first kind defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard.

### IV.1.10 Fresnel sine integral

```
double fresnel_s(double x);
float fresnel_sf(float x);
long double fresnel_sl(long double x);
```

1 *Effects*: These functions compute the Fresnel sine integrals of their respective arguments `x`.

2 *Returns*: The `fresnel_s` functions return

$$\mathrm{S}\,(x) = \int_0^x \sin\left(\frac{\pi}{2} t^2\right) dt, \qquad \text{for } x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (same definition).

### IV.1.11 Fresnel cosine integral

```
double fresnel_c(double x);
float fresnel_cf(float x);
long double fresnel_cl(long double x);
```

1 *Effects*: These functions compute the Fresnel cosine integrals of their respective arguments `x`.

2 *Returns*: The `fresnel_c` functions return

$$\mathrm{C}\,(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right) dt, \qquad \text{for } x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (same definition).

### IV.1.12 Airy function of the first kind

```
double airy_ai(double x);
float airy_aif(float x);
long double airy_ail(long double x);
```

1 *Effects*: These functions compute the Airy functions of the first kind of their respective arguments `x`.

2 *Returns*: The `airy_ai` functions return

$$\mathrm{Ai}\,(x) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{1}{3}t^3 + xt\right) dt, \qquad \text{for } x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (equivalent definition).

### IV.1.13 Airy function of the second kind

```
double airy_bi(double x);
float airy_bif(float x);
long double airy_bil(long double x);
```

1 *Effects*: These functions compute the Airy functions of the second kind of their respective arguments `x`.

2 *Returns*: The `airy_bi` functions return

$$\mathrm{Bi}\,(x) = \frac{1}{\pi}\int_0^\infty \left[\exp\left(-\frac{1}{3}t^3 + xt\right) \sin\left(\frac{1}{3}t^3 + xt\right)\right] dt, \qquad \text{for } x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (equivalent definition).

### IV.1.14 upper incomplete gamma function

```
double gamma_u(double s, double x);
float gamma_uf(float s, float x);
long double gamma_ul(long double s, long double x);
```

1 *Effects*: These functions compute the upper incomplete gamma functions of their respective arguments `s` and `x`.

2 *Returns*: The `gamma_u` functions return

$$\Gamma\left(s, x\right) = \int_{x}^{\infty} t^{s-1} \mathrm{e}^{-t} dt, \qquad \text{for } s, x \in \mathbb{R}$$

3 *Status*: Related to the Γ and hypergeometric functions defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.16, IV.1.17, IV.1.18 and IV.1.19)

### IV.1.15 lower incomplete gamma function

```
double gamma_l(double s, double x);
float gamma_lf(float s, float x);
long double gamma_ll(long double s, long double x);
```

1 *Effects*: These functions compute the lower incomplete gamma functions of their respective arguments `s` and `x`.

2 *Returns*: The `gamma_l` functions return

$$\gamma\left(s, x\right) = \int_{0}^{x} t^{s-1} \mathrm{e}^{-t} dt, \qquad \text{for } s, x \in \mathbb{R}$$

3 *Status*: Related to the Γ and hypergeometric functions defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.16, IV.1.17, IV.1.18 and IV.1.19)

### IV.1.16 upper Pochhammer symbol

```
double pochhammer_u(double n, double x);
float pochhammer_uf(float n, float x);
long double pochhammer_ul(long double n, long double x);
```

1 *Effects*: These functions compute the upper Pochhammer symbols of their respective arguments `n` and `x`.

2 *Returns*: The `pochhammer_u` functions return

$$x^{(n)} = \begin{cases} \frac{\Gamma(x+n)}{\Gamma(x)}, & \text{if } n \in \mathbb{R}^{+*}, x \in \mathbb{R} \\ 1, & \text{if } n = 0, x \in \mathbb{R} \end{cases}$$

3 *Status*: Related to the Γ and hypergeometric functions defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.18 and IV.1.19)

### IV.1.17 lower Pochhammer symbol

```
double pochhammer_l(double n, double x);
float pochhammer_lf(float n, float x);
long double pochhammer_ll(long double n, long double x);
```

1 *Effects*: These functions compute the lower Pochhammer symbols of their respective arguments `n` and `x`.

2 *Returns*: The `pochhammer_l` functions return

$$(x)^n = \begin{cases} \frac{\Gamma(x+1)}{\Gamma(x-n+1)}, & \text{if } n \in \mathbb{R}^{+*}, x \in \mathbb{R} \\ 1, & \text{if } n = 0, x \in \mathbb{R} \end{cases}$$

3 *Status*: Related to the $\Gamma$ and hypergeometric functions defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.18 and IV.1.19)

### IV.1.18 confluent hypergeometric functions

```
double hgeom_confluent(double a, double c, double x);
float hgeom_confluentf(float a, float c, float x);
long double hgeom_confluentl(long double a, long double c, long double x);
```

1 *Effects*: These functions compute the hypergeometric confluent functions of their respective arguments `a`, `c` and `x`.

2 *Returns*: The `hgeom_confluent` functions return

$$\mathrm{F}\left(a, c, x\right) = \sum_{n=0}^{\infty} \frac{(a)_n \, x^n}{(c)_n \, n!}, \qquad \text{for } a, c, x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (same definition).

### IV.1.19 ordinary hypergeometric functions

```
double hgeom_ordinary(double a, double b, double c, double x);
float hgeom_ordinaryf(float a, float b, float c, float x);
long double hgeom_ordinaryl(long double a, long double b, long double c, long
double x);
```

1 *Effects*: These functions compute the hypergeometric ordinary functions of their respective arguments `a`, `b`, `c` and `x`.

2 *Returns*: The `hgeom_ordinary` functions return

$$\mathrm{F}\left(a, b, c, x\right) = \sum_{n=0}^{\infty} \frac{(a)_n \, (b)_n \, x^n}{(c)_n \, n!}, \qquad \text{for } a, b, c, x \in \mathbb{R}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (same definition).

### IV.1.20 Chebyshev polynomials of the first kind

```
double chebyshev_t(unsigned n, double x);
float chebyshev_tf(unsigned n, float x);
long double chebyshev_tl(unsigned n, long double x);
```

1 *Effects*: These functions compute the Chebyshev polynomials of the first kind of their respective arguments `n` and `x`.

2 *Returns*: The `chebyshev_t` functions return the value of the polynomial defined recursively by:

$$\begin{cases} \mathrm{T}_n(x) = 1, & \text{if } n = 0, \text{ for } x \in \mathbb{R} \\ \mathrm{T}_n(x) = x, & \text{if } n = 1, \text{ for } x \in \mathbb{R} \\ \mathrm{T}_n(x) = 2x\mathrm{T}_{n-1}(x) - \mathrm{T}_{n-2}(x), & \text{if } n \geq 2, \text{ for } n \in \mathbb{N}, x \in \mathbb{R} \end{cases}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (extended recursive definition).

### IV.1.21 Chebyshev polynomials of the second kind

```
double chebyshev_u(unsigned n, double x);
float chebyshev_uf(unsigned n, float x);
long double chebyshev_ul(unsigned n, long double x);
```

1 *Effects*: These functions compute the Chebyshev polynomials of the second kind of their respective arguments `n` and `x`.

2 *Returns*: The `chebyshev_u` functions return the value of the polynomial defined recursively by:

$$\begin{cases} \mathrm{U}_n(x) = 1, & \text{if } n = 0, \text{ for } x \in \mathbb{R} \\ \mathrm{U}_n(x) = 2x, & \text{if } n = 1, \text{ for } x \in \mathbb{R} \\ \mathrm{U}_n(x) = 2x\mathrm{U}_{n-1}(x) - \mathrm{U}_{n-2}(x), & \text{if } n \geq 2, \text{ for } n \in \mathbb{N}, x \in \mathbb{R} \end{cases}$$

3 *Status*: Defined as a special mathematical function of the ISO/IEC 80000-2:2009 standard (extended recursive definition).

### IV.1.22 Chebyshev polynomials of the third kind

```
double chebyshev_v(unsigned n, double x);
float chebyshev_vf(unsigned n, float x);
long double chebyshev_vl(unsigned n, long double x);
```

1 *Effects*: These functions compute the Chebyshev polynomials of the third kind of their respective arguments `n` and `x`.

2 *Returns*: The `chebyshev_v` functions return the value of the polynomial defined recursively by:

$$\begin{cases} \mathrm{V}_n(x) = 1, & \text{if } n = 0, \text{ for } x \in \mathbb{R} \\ \mathrm{V}_n(x) = 2x - 1, & \text{if } n = 1, \text{ for } x \in \mathbb{R} \\ \mathrm{V}_n(x) = 2x\mathrm{V}_{n-1}(x) - \mathrm{V}_{n-2}(x), & \text{if } n \geq 2, \text{ for } n \in \mathbb{N}, x \in \mathbb{R} \end{cases}$$

3 *Status*: Complementary of the Chebyshev polynomials of the first and second kind defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.20 and IV.1.21).

### IV.1.23 Chebyshev polynomials of the fourth kind

```
double chebyshev_w(unsigned n, double x);
float chebyshev_wf(unsigned n, float x);
long double chebyshev_wl(unsigned n, long double x);
```

1 *Effects*: These functions compute the Chebyshev polynomials of the fourth kind of their respective arguments `n` and `x`.

2 *Returns*: The `chebyshev_w` functions return the value of the polynomial defined recursively by:

$$\begin{cases} \mathrm{W}_n(x) = 1, & \text{if } n = 0, \text{ for } x \in \mathbb{R} \\ \mathrm{W}_n(x) = 2x + 1, & \text{if } n = 1, \text{ for } x \in \mathbb{R} \\ \mathrm{W}_n(x) = 2x\mathrm{W}_{n-1}(x) - \mathrm{W}_{n-2}(x), & \text{if } n \geq 2, \text{ for } n \in \mathbb{N}, x \in \mathbb{R} \end{cases}$$

3 *Status*: Complementary of the Chebyshev polynomials of the first and second kind defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.20 and IV.1.21).

### IV.1.24 Jacobi polynomials

```
double jacobi(unsigned n, double alpha, double beta, double x);
float jacobif(unsigned n, float alpha, float beta, float x);
long double jacobil(unsigned n, long double alpha, long double beta, long double
x);
```

1 *Effects*: These functions compute the Jacobi polynomials of their respective arguments `n`, `alpha`, `beta` and `x`.

2 *Returns*: The `jacobi` functions return the value of the polynomial defined recursively by:

$$\begin{cases} \mathrm{P}_n^{(\alpha,\beta)}(x) = 1, & \text{if } n = 0, \text{ for } \alpha, \beta, x \in \mathbb{R} \\ \mathrm{P}_n^{(\alpha,\beta)}(x) = (\alpha + 1) + \frac{1}{2}(\alpha + \beta + 2)(x - 1), & \text{if } n = 1, \text{ for } \alpha, \beta, x \in \mathbb{R} \\ \\ \begin{aligned} & 2(n+1)(n+\alpha+\beta+1)(n+\alpha+\beta)\mathrm{P}_n^{(\alpha,\beta)}(x) \\ & = \left[(n+\alpha+\beta+1)\left(\alpha^2 - \beta^2\right) + (n+\alpha+\beta)_3 x\right] \times \mathrm{P}_{n-1}^{(\alpha,\beta)}(x) \\ & - 2(n+\alpha)(n+\beta) \times (n+\alpha+\beta+2)\mathrm{P}_{n-2}^{(\alpha,\beta)}(x) \end{aligned} & \begin{array}{l} \text{if } n \geq 2 \\ \text{for } n \in \mathbb{N}, \alpha, \beta, x \in \mathbb{R} \end{array} \end{cases}$$

3 *Status*: Generalization of Legendre and Chebyshev polynomials defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.20, IV.1.21, IV.1.22 and IV.1.23).

### IV.1.25 Gegenbauer polynomials

```
double gegenbauer(unsigned n, double alpha, double x);
float gegenbauerf(unsigned n, float alpha, float x);
long double gegenbauerl(unsigned n, long double alpha, long double x);
```

1 *Effects*: These functions compute the Gegenbauer polynomials of their respective arguments `n`, `alpha`, and `x`.

2 *Returns*: The `gegenbauer` functions return the value of the polynomial defined recursively by:

$$\begin{cases} \mathrm{C}_n^{(\alpha)}(x) = 1, & \text{if } n = 0, \text{ for } \alpha, x \in \mathbb{R} \\ \mathrm{C}_n^{(\alpha)}(x) = 2\alpha x, & \text{if } n = 1, \text{ for } \alpha, x \in \mathbb{R} \\ n\mathrm{C}_n^{(\alpha)}(x) = 2(n+\alpha-1)\, x\mathrm{C}_{n-1}^{(\alpha)}(x) - (n+2\alpha-2)\,\mathrm{C}_{n-2}^{(\alpha)}(x), & \begin{array}{l}\text{if } n \geq 2 \\ \text{for } n \in \mathbb{N}, \alpha, x \in \mathbb{R}\end{array} \end{cases}$$

3 *Status*: Generalization of Legendre and Chebyshev polynomials defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.20, IV.1.21, IV.1.22 and IV.1.23).

### IV.1.26 Zernike polynomials

```
double zernike(unsigned n, int m, double rho, double phi);
float zernikef(unsigned n, int m, float rho, float phi);
long double zernikel(unsigned n, int m, long double rho, long double phi);
```

1 *Effects*: These functions compute the Zernike polynomials of their respective arguments `n`, `m`, `rho` and `phi`.

2 *Returns*: The `zernike` functions return the value of the polynomial defined by:

$$\mathrm{Z}_n^m(\rho, \phi) = \mathrm{R}_n^{|m|}(\rho) \times \begin{cases} \cos(m\phi), & \text{if } m \geq 0 \\ \sin(m\phi), & \text{if } m < 0 \end{cases} \qquad \text{for } |m| \leq n, m \in \mathbb{Z}, n \in \mathbb{R} \text{ and } \rho, \phi \in \mathbb{R}$$

3 *Status*: Orthogonal polynomials as the Legendre and Chebyshev polynomials defined as special mathematical functions of the ISO/IEC 80000-2:2009 standard (see IV.1.20, IV.1.21, IV.1.22 and IV.1.23).

### IV.1.27 radial polynomials

```
double radpoly(unsigned n, unsigned m, double rho);
float radpolyf(unsigned n, unsigned m, float rho);
long double radpolyl(unsigned n, unsigned m, long double rho);
```

1 *Effects*: These functions compute the radial polynomials of their respective arguments `n`, `m` and `rho`.

2 *Returns*: The `radpoly` functions return the value of the polynomial defined by:

$$\mathrm{R}_n^m(\rho) = \begin{cases} 0, & \text{if } n-m \text{ odd} \\ \sum_{k=0}^{\frac{n-m}{2}} \frac{(-1)^k (n-k)!}{k!\left(\frac{n+m}{2}-k\right)!\left(\frac{n-m}{2}-k\right)!}\rho^{n-2k}, & \text{if } n-m \text{ even} \end{cases} \qquad \text{for } m \leq n, n, m \in \mathbb{N}, \rho \in \mathbb{R}$$

3 *Status*: Specialization of Jacobi polynomials required by Zernike polynomials (see IV.1.24 and IV.1.26).

## IV.2 Additions to header `<math.h>`

The header `<math.h>` shall have sufficient additional using declarations to import into the global name space all of the function names specified in the previous section.

### IV.3 The header `<ctgmath>`

The header `<ctgmath>`, if provided by the implementation, effectively includes the header `<math.h>`.

### IV.4 The header `<tgmath>`

The header `<tgmath>`, if provided by the implementation, effectively includes the header `<math.h>`.

## V References


- ISO/IEC 29124:2010 Information technology – Programming languages, their environments and system software interfaces – Extensions to the C++ Library to support mathematical special functions
- ISO/IEC 80000-2:2009 Quantities and units – Part 2: Mathematical signs and symbols to be used in the natural sciences and technology
- William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery - Numerical Recipes: The Art of Scientific Computing, 3rd Edition - Cambridge University Press, 2007
- John C. Mason and David C. Handscomb - The Chebyshev Polynomials - CRC Press, 2003
- Wikipedia: List of mathematical functions
- Wolfram MathWorld: http://mathworld.wolfram.com