Fast and accurate algorithm for the generalized exponential integral $E_\nu(x)$ for positive real order

Guillermo Navas-Palencia *[1]

[1] *Dept. Computer Science, Universitat Politècnica de Catalunya, Barcelona, Spain*
[1] *Numerical Algorithms Group Ltd, Oxford, United Kingdom*

April 26, 2017

Abstract

We describe an algorithm for the numerical evaluation of the generalized exponential integral $E_\nu(x)$ for positive values of ν and x. A detailed description of the numerical methods used in the algorithm is provided, including error bounds. Different approaches from earlier algorithms are also summarised. The performance and accuracy of the resulting algorithm is analysed and compared with open-source software packages. This analysis shows that our implementation is competitive and more robust than other state-of-the-art codes. Finally, a brief study of the implementation of $E_\nu(x)$ in arbitrary-precision arithmetic is discussed.

1 Introduction

The generalized exponential integral is defined by

$$E_\nu(x) = \int_1^\infty e^{-xt} t^{-\nu}\, dt, \quad \nu \in \mathbb{R},\ x > 0. \tag{1}$$

Another interesting integral representation is given by

$$E_\nu(x) = x^{\nu-1} \int_x^\infty e^{-t} t^{-\nu}\, dt. \tag{2}$$

The generalized exponential integral can also be expressed in terms of the upper and lower incomplete gamma functions ($\Gamma(a,x)$ and $\gamma(a,x)$, respectively) by means of the following functional relations,

$$E_\nu(x) = x^{\nu-1}\Gamma(1-\nu, x), \tag{3}$$

$$E_\nu(x) = x^{\nu-1}(\Gamma(1-\nu) - \gamma(1-\nu, x)). \tag{4}$$

Recently, numerical methods for the evaluation of incomplete gamma functions have been extensively investigated in [19, 21], therefore those proposed algorithms could be used when $\nu < 0$. Albeit Equation (3) is being used in several software packages, its direct application may lead to unsatisfactory results, as will be shown throughout this work. The main contribution of this paper is a detailed algorithm for the computation of $E_\nu(x)$ for real and integer order ν, which avoids recursive calculations and includes new numerical methods not present in other existing algorithms. These new computation schema are more efficient and return more accurate results than available software packages in double-precision floating-point arithmetic.

*g.navas.palencia@gmail.com

The function $E_\nu(x)$, with $\nu > 0$, appears in many fields of physics and engineering, in particular is of interest its connection with transport theory and radiative equilibrium [1]. For other values of ν, $E_\nu(x)$ is encountered in the computation of molecular electronic integrals in quantum chemistry and wave acoustics of overlapping sound beams. Some well-known examples are the Schwarzschild-Milne integral equation [6] or the generalization of Chandrasekhar's integrals [10]. Besides the applications in physics, the generalized exponential integral arises in several special cases for more difficult special functions, such as the *confluent hypergeometric functions* ${}_1F_1(a;b;x)$ and $U(a,b,x)$. For instance, $E_\nu(x)$ can be defined in terms of $U(a,b,x)$

$$E_\nu(x) = x^{\nu-1}e^{-x}U(\nu,\nu,x) \tag{5}$$

and using Kummer's transformation $U(a,b,x) = z^{1-b}U(a-b+1,2-b,x)$ we can write (5) in the form

$$E_\nu(x) = e^{-x}U(1,2-\nu,x). \tag{6}$$

Moreover, the generalized exponential integral plays an important role in some exponentially improved asymptotic expansions (also known as hyperasymptotic expansions) for the confluent hypergeometric function $U(a,b,x)$ (U-Kummer function) [31] and [15, §13.7(iii)]

$$U(a,b,x) = x^{-a}\sum_{k=0}^{N-1}\frac{(a)_k(a-b+1)_k}{k!}(-x)^{-k} + R_N(a,b,x) \tag{7}$$

and

$$R_N(a,b,x) = \frac{(-1)^N 2\pi x^{a-b}}{\Gamma(a)\Gamma(a-b+1)}\Bigg(\sum_{k=0}^{M-1}\frac{(1-a)_k(b-a)_k}{k!}(-x)^{-k}G_{N+2a-b-k}(x) + (1-a)_M(b-a)_M R_{M,N}(a,b,x)\Bigg), \tag{8}$$

where M is an arbitrary non-negative integer, and

$$G_p(x) = \frac{e^x\Gamma(p)}{2\pi x^{p-1}}E_p(x), \tag{9}$$

$G_p(x)$ being the so-called *terminant function*. Then as $|x|\to\infty$ with a, b and M fixed

$$R_{M,N}(a,b,x) = \begin{cases} O(e^{-x}x^{-M}), & x>0 \\ O(e^{x}x^{-M}), & x<0 \end{cases} \tag{10}$$

Finally, for a collection of integrals involving $E_\nu(x)$, refer to [10, 29] and [15, §8.19(x)].

An extensive study of different methods for the computation of the generalized exponential integral has been carried out by Chiccoli, Lorenzutta and Maino in [7, 8, 9]. The main method described by the authors, valid for real positive ν and x, is essentially based on recursive calculations starting from a different series representations for the cases $x < 1$ and $x \geq 1$. This algorithm applies the recurrence

$$E_\nu = \frac{1}{x}(e^{-x} - \nu E_{\nu+1}(x)), \quad (x>0,\ \nu\in\mathbb{R}) \tag{11}$$

and combines Taylor series expansion, series expansion in terms of Tricomi functions and uniform asymptotic expansion for large ν.

The outline of the paper is the following: in Section 2 we study some of the main numerical methods considered for the evaluation of $E_\nu(x)$, including error bounds, and we describe a new asymptotic expansion along with several improvements for difficult regions of computation. Then, in Section 3 we review other methods of computation implemented in several

software packages, focusing in quadrature methods and suitable integral representations. In Section 4 we present our algorithm and a detailed description of its implementation. After that, in Section 5 we assess the performance of our algorithm – in terms of relative error and computation time – and we compare to other publicly available codes. Finally, in Section 6 we present our conclusions.

2 Methods of computation

In this section we provide a detailed description of the methods of computation implemented in the algorithm, both for the special case $\nu \equiv n, n \in \mathbb{N}$ and the general case $\nu \in \mathbb{R}^+$.

2.1 Special values

Special cases are typically treated separately, here we list some of the most relevant cases:

$$E_0(x) = \frac{e^{-x}}{x}, \quad x \neq 0, \tag{12}$$

which can be used as starting point for recursive evaluation of $E_n(x)$. For $x = 0$

$$E_\nu(0) = \begin{cases} \frac{1}{\nu-1}, & \nu > 1 \\ \infty, & \nu \in (-\infty, 1] \end{cases} \tag{13}$$

For $\nu = 1/2$ we have the following value, which is also used for recursive evaluation of half-integers

$$E_{\frac{1}{2}}(x) = \frac{\sqrt{\pi}}{\sqrt{x}}\operatorname{erfc}(\sqrt{x}). \tag{14}$$

Finally, $\nu = 1$, the so-called *exponential integral*

$$E_1(x) = -\operatorname{Ei}(-x), \quad x > 0. \tag{15}$$

2.2 Series expansions

Series expansions for the generalized exponential integral are commonly used in the domain $|x| \lesssim 1$. We consider two different series expansions valid when the parameter $\nu \in \mathbb{R} \setminus \mathbb{N}$. The first expansion [15, §8.19.10] is given by

$$E_\nu(x) = \Gamma(1-\nu)x^{\nu-1} - \sum_{k=0}^{\infty} \frac{(-1)^k x^k}{(1-\nu+k)k!}, \quad \nu \in \mathbb{R} \setminus \mathbb{N},\ x \in \mathbb{R} \setminus \{0\}. \tag{16}$$

The terms of the series expansion at the origin decrease for $|x| < 1$, and in practice the best performance is observed in this region, where fast convergence is experimented. The series expansion is alternating and hence the computation of such a sum leads to catastrophic cancellation issues for $x > 1$, especially in finite precision arithmetic. For $x < 0$, ν not being a negative integer, the result is complex. The principal branch of the generalized exponential integral is defined by taking the principal branch of the natural logarithm in $\exp((\nu-1)\log(x)) = x^{\nu-1}$ (with the logarithm branch cut $(-\infty, 0)$). Along the negative real axis, the cancellation errors on the power series are removed, although the required number of terms increases due to slower convergence. For its evaluation, the series is truncated after a level of precision 2^{-p} (p number of bits) is obtained, therefore it can be written as follows,

$$E_\nu(x) = \Gamma(1-\nu)x^{\nu-1} - \left(\sum_{k=0}^{N-1} \frac{(-x)^k}{(1-\nu+k)k!} + \sum_{k=N}^{\infty} \frac{(-x)^k}{(1-\nu+k)k!} \right). \tag{17}$$

For $x > 0$, Equation (16) is an alternating series and thus the remainder is easily bounded by the first neglected term, therefore we choose N such that $x^N/(|1-\nu+N|N!) < 2^{-p}$. For small values of x, the first neglected term tends rapidly to 0 as $k \to \infty$. However, the required number of terms can grow considerably when $x > 1$, leading to initial ascending terms before the series starts to show convergence, which originates a loss of digits due to roundoff errors.

2.2.1 Series in terms of the confluent hypergeometric function

As previously stated, the generalized exponential integral can be defined in terms of the confluent hypergeometric function of the first kind, ${}_1F_1(a;b;x)$,

$$E_\nu(x) = \Gamma(1-\nu)x^{\nu-1} + \frac{{}_1F_1(1-\nu;2-\nu;-x)}{\nu-1}, \tag{18}$$

where ${}_1F_1(a;b;z)$ is defined as

$${}_1F_1(a;b;z) = \sum_{k=0}^{\infty} \frac{(a)_k z^k}{(b)_k k!} \tag{19}$$

for $a \in \mathbb{C}$, $b \in \mathbb{C} \setminus \mathbb{Z}_0^-$ and $z \in \mathbb{C}$. Series (18) is equivalent to series (16), note that $(1-\nu)_k/(2-\nu)_k = (1-\nu)/(1-\nu+k)$. In order to reduce significant cancellation issues, we apply Kummer's transformation ${}_1F_1(a;b;z) = e_1^z F_1(b-a;b;-z)$, thus Equation (18) can be written in this form

$$\begin{aligned} E_\nu(x) &= \Gamma(1-\nu)x^{\nu-1} + \frac{e^{-x}}{\nu-1}\,{}_1F_1(1;2-\nu;x) \\ &= \Gamma(1-\nu)x^{\nu-1} + \frac{e^{-x}}{\nu-1}\sum_{k=0}^{\infty}\frac{x^k}{(2-\nu)_k}. \end{aligned} \tag{20}$$

This series expansion turns out to be more numerically stable, especially for small values of ν, (for $\nu < 2$ all the terms of the expansion are positive), but the convergence is slightly slower. Similarly to the previous series representation, the number of terms diminish for $|x| < 1$. A rigorous error bound for the evaluation of convergent generalized hypergeometric series ${}_pF_q(a_1,\ldots,a_p;b_1,\ldots,b_q;z)$ in devised in [25].

Expansion (20) can be effectively used as an asymptotic expansion for large ν and fixed x. In fact, note that when $\nu \gg x$ the term $\Gamma(1-\nu)x^{\nu-1}$ is not significant compared to the terms in the finite sum and it can be neglected when less than ν terms are used. As pointed out by an anonymous referee, when $\nu = n$, n being a large integer, and not many terms are considered, we can neglect the contribution of $\Gamma(1-\nu)x^{\nu-1}$ because it should be combined with the $k = n-1$ term, which gives the term shown in series expansion (38).

2.2.2 Laguerre series

A globally convergent Laguerre series for the incomplete gamma function $\Gamma(a,x)$ is described in [22]

$$\Gamma(a,x) = x^{a-1}e^{-x}\sum_{k=0}^{\infty}\frac{(1-a)_k}{(k+1)!L_k^{-a}(-x)L_{k+1}^{-a}(-x)}, \tag{21}$$

where $L_k^{-a}(-x)$ is a generalized Laguerre polynomial. The region of validity of the Laguerre series is outside the zeros of Laguerre polynomials. Zeros of $L_k^a(x)$ occur in the interval $x \in (0, 4\kappa)$ where $\kappa = n + \frac{1}{2}(a+1)$, so for our region of interest ($x > 0$ and $a = 1-\nu$) $L_k^{-a}(-x)$ lies in the monotonic region. As remarked in [4], the Laguerre series for (21) is closely related to the continued fraction for the incomplete gamma function; see §3.2 for a comparison to several continued fractions.

Applying the functional relation (3) in (21) we obtain the Laguerre series for $E_\nu(x)$,

$$E_\nu(x) = e^{-x}\sum_{k=0}^{\infty}\frac{(\nu)_k}{(k+1)!L_k^{(\nu-1)}(-x)L_{k+1}^{(\nu-1)}(-x)}. \tag{22}$$

This series has two properties that make it very suitable for our regime of parameters: it does not exhibit cancellations and the error can be made arbitrarily small by increasing the number of terms (in contrast to asymptotic expansions, which have an optimal value).

The generalized Laguerre polynomials satisfy the three-term recurrence relation

$$L_{k+1}^{(\nu-1)}(-x) = \frac{x+2k+\nu}{k+1}L_k^{(\nu-1)}(-x) - \frac{k+\nu-1}{k+1}L_{k-1}^{(\nu-1)}(-x), \tag{23}$$

which is not ill-conditioned in both backward and forward direction, consequently it is used with initial values $L_0^{(\nu-1)}(-x) = 1$ and $L_1^{(\nu-1)}(-x) = x+\nu$. Moreover, given the reduced number of terms needed, the loss of precision is almost negligible. In [4] a sub-exponential error bound valid for non-negative real x is given for $L_k^{-a}(-x)$,

$$L_k^{(-a)}(-x) \sim S_k(a,x)\left(1+O\left(\frac{1}{m^{1/2}}\right)\right), \tag{24}$$

where $m = k+1$ and the sub-exponential[1] term $S_k(a,x)$ is

$$S_k(a,x) = \frac{e^{-x/2}e^{2\sqrt{mx}}}{2\sqrt{\pi}x^{1/4-a/2}m^{1/4+a/2}}. \tag{25}$$

This error bound results to be particularly effective for large n. For moderate and large values of ν and/or x in a "medium-precision" range[2] the Laguerre series converges after a small number of terms and thus the error term of the sub-exponential estimate is significant. We now proceed to calculate approximations for the coefficients of the Laguerre series define as

$$E_\nu(x) = e^{-x}\sum_{k=0}^{\infty}a_k, \quad \text{and} \quad a_k = \frac{(\nu)_k}{(k+1)!L_k^{(\nu-1)}(-x)L_{k+1}^{(\nu-1)}(-x)}. \tag{26}$$

For $k \geq 1$,

$$\sqrt{2\pi}k^{k+1/2}e^{-k} \leq k! \leq ek^{k+1/2}e^{-k},$$

and given $(\nu)_k = \Gamma(\nu+k)/\Gamma(\nu)$ the following inequality [27, Theorem 1] holds for $\Gamma(b)/\Gamma(a)$ and $b > a > 1$,

$$\frac{b^{b-1}}{a^{a-1}}e^{a-b} < \frac{\Gamma(b)}{\Gamma(a)} < \frac{b^{b-1/2}}{a^{a-1/2}}e^{a-b}, \tag{27}$$

so $(\nu)_k/(k+1)!$ satisfy

$$\frac{(\nu)_k}{(k+1)!} < C_k(\nu,x) = \left(\frac{\nu}{2\pi(\nu+k)}\right)^{1/2}\frac{(\nu+k)^{\nu+k}e^{k+1}}{\nu^\nu(k+1)^{k+3/2}}. \tag{28}$$

For $x \gg \nu$ we shall use the relation $L_k^{\nu-1}(-x) = \frac{(-1)^k}{k!}U(-k,\nu,-x)$ and the property $U(a,b,z) \sim z^{-a}$, $z \to \infty$, $|\mathrm{ph}\, z| < \frac{3}{2}\pi$, therefore

$$L_k^{(\nu-1)}(-x) \sim B_k(x) = \frac{(x)^k}{k!}. \tag{29}$$

Combining (28) and (29) the coefficients a_k satisfy

$$a_k \sim a_k^B = \frac{C_k(\nu,x)}{B_k(x)B_{k+1}(x)}. \tag{30}$$

[1] In [4], sub-exponential growth is defined as $\log\log|f(n)| \sim \delta\log n$, for some $0 < \delta < 1$.

[2] We consider medium-precision the range from double precision to up to a few hundred bits of precision.

When k is small and $x \lesssim \nu$, a first order approximation is given by

$$L_k^{(\nu-1)}(-x) \sim A_k(\nu, x) = \binom{k+\nu-1}{k}\left(1+\frac{x}{\nu}\right)^k. \tag{31}$$

Again, combining (28) and (31) we obtain the following approximation for a_k

$$a_k \sim a_k^A = \frac{C_k(\nu, x)}{A_k(x)A_{k+1}(x)}. \tag{32}$$

Note that approximation $A_k(\nu, x)$ indicates that for small values of x, $a_k \sim k!/(\nu)_{k+1}$ and the rate of convergence is generally slow, depending entirely on ν. Therefore, if ν is not sufficiently large, the method is not fast enough for efficient numerical evaluation.

Table 1 shows the first neglected term $a_N < 2^{-53}$ and the corresponding values for a_N^A and a_N^B. Observe that approximation a_N^A acts as an upper bound for large ν, whereas tends to overestimate a_N for smaller values. Nevertheless, one could easily devise a heuristic in order to compensate that overestimation by adding q extra bits such that $a_N < 2^{-p-q}$, $q < p$. For large x, a_N^B bounds a_N adequately, but is too conservative when $\nu > x$.

$\boldsymbol{\nu}$	$\boldsymbol{x}$	$\boldsymbol{N}$	$\boldsymbol{a_N}$	$\boldsymbol{a_N^A}$	$\boldsymbol{a_N^B}$
10	10	17	2.84e−17	3.48e−19	3.99e+00
100	10	10	2.9e−17	2.15e−16	5.66e+06
10	100	6	4.14e−18	3.67e−19	2.63e−17
100	100	7	1.32e−17	1.18e−17	6.26e−13
500	500	5	3.69e−18	3.69e−18	7.94e−15
500	100	6	8.28e−18	8.94e−17	1.17e−07
100	500	5	3.38e−19	2.89e−19	2.75e−18
10000	10	4	2.38e−19	2.44e−19	2.44e+08
10	10000	2	2.19e−18	1.55e−18	2.26e−18

Table 1: Approximation terms a_N.

In practice, these approximations along with double-precision arithmetic are used to select N using linear search or by inverting these approximations in function of N. This approach is particularly useful in arbitrary-precision interval arithmetic.

2.2.3 Taylor series for $1 \le x < 2$

As previously stated, the series expansions for $x > 1$ exhibit cancellation issues, therefore other methods need to be used, especially if the working precision cannot be increased to compensate the bad condition number of the series. Henceforth we fix the working precision at 53-bit. For values of x such that $x \in [1, 2)$, we consider the following Taylor series described in [7]

$$E_\nu(x-y) = \sum_{k=0}^{\infty} \frac{y^k}{k!} E_{\nu-k}(x), \quad x > 0,\ (\nu, y) \in \mathbb{R}, \tag{33}$$

which is obtained from the Taylor series [7, (10)]

$$E_\nu(x-y) = \sum_{k=0}^{\infty} \frac{(-y)^k}{k!}\left[\frac{\mathrm{d}^k}{\mathrm{d}x^k} E_{\nu-k}(x)\right],$$

making use of the following differential formula [7, (11)]

$$\frac{\mathrm{d}^k}{\mathrm{d}x^k} E_\nu(x) = (-1)^k E_{\nu-k}(x).$$

The Taylor series truncated at $k = N$ is given by

$$E_\nu(x-y) = \sum_{k=0}^{N-1} \frac{y^k}{k!} E_{\nu-k}(x) + \sum_{k=N}^{\infty} \frac{y^k}{k!} E_{\nu-k}(x). \tag{34}$$

Proposition 2.1 *Given $\nu \geq 1$, $x > 0$ and positive integer N, such that $\lfloor \nu + x - 1 \rfloor > N$, the remainder of the Taylor series in (34) for $|y| \leq 1$ satisfies*

$$\sum_{k=N}^{\infty} \frac{y^k}{k!} E_{\nu-k}(x) < \frac{4e^{-x}\lfloor \nu + x - 1 \rfloor}{x} \frac{|y|^N}{N!}. \tag{35}$$

Proof: Starting with the integral definition (2), it immediately results that the generalized exponential integrals is monotonic increasing as $\nu \to -\infty$, satisfying the inequality $E_\nu(x) > E_{\nu+1}(x)$. This implies that

$$E_r(x) \leq E_0(x) = \frac{e^{-x}}{x}, \quad r \in [0, \infty).$$

With the assumption $\lfloor \nu + x - 1 \rfloor > N$ and using the well-known upper bound for the generalized exponential integral in this domain, the following inequality holds

$$E_{\nu-k}(x) \leq \frac{e^{-x}}{\nu + x - k - 1} < \frac{e^{-x}}{x}.$$

Hence,

$$\sum_{k=N}^{\infty} \frac{y^k}{k!} E_{\nu-k}(x) < \frac{\lfloor \nu + x - 1 \rfloor e^{-x}}{x} \sum_{k=N}^{\infty} \frac{y^k}{k!}.$$

Thus it remains to bound the series expansion. By observing that $\sum_{k=N}^{\infty} y^k/k!$ is equivalent to the remainder term after truncating the Taylor series of e^y, we can use the well-known upper bound for $|y| \leq 1$

$$\sum_{k=N}^{\infty} \frac{y^k}{k!} \leq 4\frac{y^N}{N!}.$$

Finally, combining both bounds the upper bound for the remainder is obtained. □

By using bound (35) we can determine the required number of terms N in order to target a level of precision 2^{-53}. For $x \to 1$ and $y \to -1$, $N = 20$ terms suffice, meaning that $\nu \gtrsim 21$; see Figure (1) (left). This constraint demands the use of recurrence relations for smaller values of ν. For example, we can use the following recursion after increasing ν

$$E_{\nu-n}(x) = \frac{(1-\nu)_n}{x^n}\left(E_\nu(x) + e^{-x}\sum_{k=0}^{n-1} \frac{x^k}{(1-\nu)_{k+1}}\right), \quad n \in \mathbb{N}. \tag{36}$$

Furthermore, by using recursion (36), this method can be applied for $\nu < 0$. Regarding the finite series in (36), for $\nu - n > 0$ the minimum value occurs for $k \sim \nu$; see Figure (1) (right). In order to evaluate $E_{\nu-k}(x)$ we make use of the recurrence relation [15, §8.19.12]

$$E_\nu(x) = \frac{1}{x}(e^{-x} - \nu E_{\nu+1}(x)). \tag{37}$$

Hence, we just need to compute the first $E_\nu(x)$ ($k = 0$), in N_1 terms, and for $x \to 1$ and $\nu \approx 21$ we require $N_2 = 18$. Thus, when $x - y \approx 2$, this algorithm would require $N_1 + N_2 + N_3$ terms, where N_3 is the number of recursions in (36). For implementation purposes, a simple choice $x = -y = (x-y)/2$ works well, although an iterative procedure could optimize[3] these parameters.

[3]The optimal choice arises as a solution of a non-convex mix integer nonlinear programming problem, which is prohibitively expensive to compute.

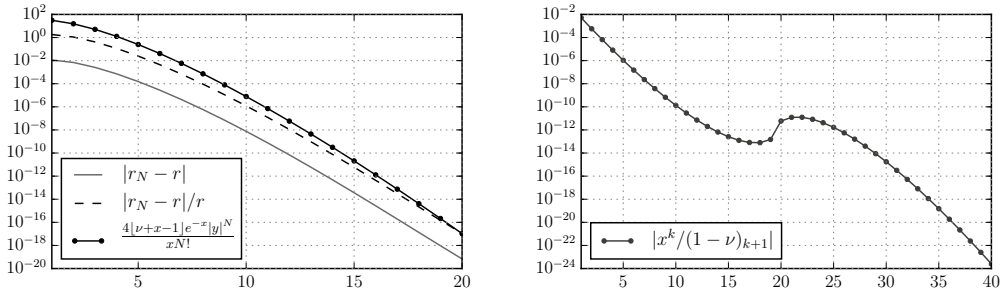

Figure 1: Plot of the absolute and relative errors of $E_{21.05}(1.98)$ and error bound (35) for $N \in [1, 20]$ (left). Plot of $|x^k/(1 - 21.05)_{k+1}|$, $x = 2$ for $k = [1, 40]$ (right).

2.2.4 Series expansions: special cases

Previous series expansions, excepting Laguerre series, cannot be used for integer ν. For this special case we introduce two series expansions, see [15, §8.19]:

Case: $n \in \mathbb{N}$

$$E_n(x) = \frac{(-x)^{n-1}}{(n-1)!}(\psi_0(n) - \log(x)) - \sum_{k=0,\, k\neq n-1}^{\infty} \frac{(-x)^k}{k!(1-n+k)}, \tag{38}$$

where $\psi_0(t)$ is the digamma function, which for integer $t = n$ is denoted as

$$\psi_0(n) = -\gamma + \sum_{k=1}^{n-1} \frac{1}{k} = -\gamma + H_{n-1},$$

where γ is the Euler-Mascheroni constant and H_n is a harmonic number. Alike the series expansions (16) and (20), this series performs better for $x < 1$. Other series expansion in terms of the exponential integral $E_1(x)$ (also known as Theis well function) is given by

$$\begin{aligned} E_n(x) &= \frac{(-x)^{n-1}}{(n-1)!}E_1(x) + \frac{e^{-x}}{(n-1)!}\sum_{k=0}^{n-2}(n-k-2)!(-x)^k \\ &= \frac{(-x)^{n-1}}{(n-1)!}E_1(x) + e^{-x}\sum_{k=0}^{n-2}(n)_{-k-1}(-x)^k. \end{aligned} \tag{39}$$

Case: $n + \frac{1}{2},\ n \in \mathbb{N}$

$$E_{n+\frac{1}{2}}(x) = \frac{(-1)^n\sqrt{\pi}\,x^{n-\frac{1}{2}}}{(1/2)_n}\mathrm{erfc}(\sqrt{\pi}) - e^{-x}\sum_{k=0}^{n-1}\frac{x^k}{(1/2-n)_{k+1}}. \tag{40}$$

This series expansion for half-integers is used for moderate values of n and small x in order to reduce the effect of cancellation.

Case: $n + \epsilon,\ n \in \mathbb{N}$**:** A difficult case arises when $\nu = n + \epsilon$, $|\epsilon| \ll 1$, for small values of x, say $x \lesssim 2$. A direct evaluation of series expansion (16) leads to significant loss of precision both in the series expansion and $\Gamma(1 - n - \epsilon)x^{n-1+\epsilon}$. Moreover, final subtraction of both terms incurs in catastrophic cancellation, since both are of large magnitude. To deal with this issue, series expansion (16) is also implemented in quadruple precision (128-bit) using

the libquadmath[4] library. Our implementation includes two interfaces: `expint(const int vi, const double vf, const double x)` and `expint(const double v, const double x)`, where `vi` and `vf` ($|\texttt{vf}| < 0.5$) denote the integral and decimal fractional part of ν, respectively. A unique interface passing $\nu = n + \epsilon$ produces a similar loss of precision, even splitting ν using functions such as `std::modf` in C++. Two relevant examples using a double precision (53-bit) implementation: $\nu = 2 + 1\text{e}{-14}$ and $x = 1\text{e}{-10}$ returns a result with relative error 1.6e-12, whereas $\nu = 1 - 1\text{e}{-13}$ and $x = 1\text{e}{-01}$ has a relative error 6.5e-06. An implementation in quadruple precision returns an relative error below 2^{-53}.

In terms of performance, quadruple precision is about $10 \sim 15$ times slower. Although other approaches are possible, in our experience, quadruple precision is indispensable to return reliable results in this regime of parameters.

2.3 Asymptotic expansions

2.3.1 Large x and fixed ν

Asymptotic expansions for $E_\nu(x)$ as $x \to \infty$ can be derived from the integral representation in Equation (2)

$$E_\nu(x) = x^{\nu-1} \int_x^\infty e^{-t} t^{-\nu}\, dt = e^{-x} \int_0^\infty e^{-xt} (1+t)^{-\nu}\, dt. \tag{41}$$

The transformation gives a Laplace integral and Watson's lemma [33] can be applied, obtaining the following asymptotic expansion

$$E_\nu(x) \sim e^{-x} \sum_{k=0}^{\infty} \frac{(-1)^k (\nu)_k}{x^{k+1}}, \quad x \in \mathbb{R}. \tag{42}$$

The remainder $\varepsilon_N(x)$ of the expansion after truncation can be written as follows,

$$E_\nu(x) = e^{-x} \left(\sum_{k=0}^{N-1} \frac{(-1)^k (\nu)_k}{x^{k+1}} + \varepsilon_N(x) \right). \tag{43}$$

For $x > 0$, the asymptotic series is alternating, and as previously stated, the remainder can be bounded by the absolute value of the first neglected term,

$$|\varepsilon_N(x)| \leq \left| \frac{(\nu)_N}{x^{N+1}} \right|. \tag{44}$$

As pointed out in §2.2.2, in asymptotic expansions the remainder cannot be reduced arbitrarily as $N \to \infty$, in fact, given ν and x, bound (44) first decreases until an optimal value of terms $N_{max} = \lceil x - \nu \rceil$ is reached. Subsequent $N > N_{max}$ increase the bound. Thus, linear search is performed up to a limit $N \leq N_{max}$, without guarantee of finding N such that $\varepsilon_N(x) < 2^{-p}$. Hence, when x ($x > 1$) is not large enough with respect to ν and the required precision bits is moderate/high, this asymptotic expansion cannot be used effectively. For an exponentially-improved asymptotic expansion see [15, §2.11(iii)].

2.3.2 Large ν

A uniform asymptotic expansion when both ν and x are large is introduced in [18]. Using the definition in [15, §8.20(ii)], the expansion is given by

$$E_\nu(x) \sim \frac{e^{-z}}{x+\nu} \sum_{k=0}^{\infty} \frac{A_k(\lambda)}{(\lambda+1)^{2k} \nu^k}, \tag{45}$$

[4]There seems to be a bug in libquadmath for `tgammaq` and `lgammaq`. It returns incorrect signs when $\epsilon < 1\text{e} - 6$. This was fixed in our implementation.

where $\lambda = x/\nu$. The coefficients $A_k(\lambda)$, starting with $A_0(\lambda) = 1$, can be computed using the recursion

$$A_{k+1}(\lambda) = (1 - 2k\lambda)A_k(\lambda) + \lambda(\lambda + 1)\frac{dA_k(\lambda)}{d\lambda}, \quad k = 0, 1, 2, \ldots \tag{46}$$

and the degree of $A_k(\lambda)$ is $k - 1$ when $k \geq 1$. In particular, the first 8 coefficients are

$$\begin{aligned}
A_0(\lambda) &= 1,\\
A_1(\lambda) &= 1,\\
A_2(\lambda) &= 1 - 2\lambda,\\
A_3(\lambda) &= 1 - 8\lambda + 6\lambda^2,\\
A_4(\lambda) &= 1 - 22\lambda + 58\lambda^2 - 24\lambda^3,\\
A_5(\lambda) &= 1 - 52\lambda + 328\lambda^2 - 444\lambda^3 + 120\lambda^4,\\
A_6(\lambda) &= 1 - 114\lambda + 1452\lambda^2 - 4400\lambda^3 + 3708\lambda^4 - 720\lambda^5,\\
A_7(\lambda) &= 1 - 240\lambda + 5610\lambda^2 - 32120\lambda^3 + 58140\lambda^4 - 33984\lambda^5 + 5040\lambda^6.
\end{aligned}$$

Remark 2.2 *It is not hard to observe that polynomials $A_{k+1}(\lambda)$ computed via recursion in (46) can be obtained for $k \geq 2$ using the series*

$$A_{k+1}(\lambda) = 1 + \sum_{j=1}^{k-1}(a_j + b_{j-1} + b_j - 2ka_{j-1})\lambda^j + (b_{k-1} - 2ka_{k-1})\lambda^k, \quad k \geq 2$$

where $a_0 = 1$, $b_0 = 0$ and $a_1 = b_1 = -2$. a_j are the coefficients of each polynomial $A_k(\lambda)$ and b_j are the coefficients of their corresponding derivatives. Given the relation $b_j = ja_j$, the above series can be simplified

$$A_{k+1}(\lambda) = 1 + \sum_{j=1}^{k-1}\{a_j(j+1) + (j - 1 - 2k)a_{j-1}\}\lambda^j + \{(k - 1 - 2)ka_{k-1}\}\lambda^k, \quad k \geq 2 \tag{47}$$

Remark 2.3 *$A_k(\lambda)$ is an Eulerian polynomial of second kind defined by*

$$A_k(\lambda) = \sum_{m=0}^{k}(-1)^m \left\langle\!\left\langle {k \atop m} \right\rangle\!\right\rangle \lambda^m, \tag{48}$$

where $\left\langle\!\left\langle {k \atop m} \right\rangle\!\right\rangle$ are second-order Eulerian numbers[5]*, defined by the recursion equation*

$$\left\langle\!\left\langle {k \atop m} \right\rangle\!\right\rangle = (m+1)\left\langle\!\left\langle {k-1 \atop m} \right\rangle\!\right\rangle + (2k - m - 1)\left\langle\!\left\langle {k-1 \atop m-1} \right\rangle\!\right\rangle, \tag{49}$$

with $\left\langle\!\left\langle {k \atop 0} \right\rangle\!\right\rangle = 1$ and $\left\langle\!\left\langle {k \atop m} \right\rangle\!\right\rangle = 0$ for $m \geq k$.

The remainder after truncating the series $\varepsilon_k(\nu, x)$ satisfies

$$\varepsilon_k(\nu, x) \leq C_k\left(1 + \frac{1}{x + \nu - 1}\right)\frac{1}{\nu^k}. \tag{50}$$

Gautschi in [18] provides rigorous bounds for $\varepsilon_k(\nu, x)$, $k \leq 7$. This uniform asymptotic expansion proves to be very powerful in a wide domain, $\lambda \in [0, \infty)$. However, each term $A_k(\lambda)$ requires the construction of a polynomial and its evaluation, which increase the computational time substantially and requires to keep $k - 1$ temporary coefficients in cache,

[5] `https://oeis.org/A008517`

which may make it unattractive for arbitrary-precision arithmetic. Nonetheless, for a fixed precision (e.g., 53-bit or 113-bit) one can precalculate as many polynomials as needed[6]. We use Horner's scheme for evaluating the polynomials $A_k(\lambda)$ for $k \geq 3$ to reduce the number of multiplications. Furthermore, we use compensated summation algorithms to minimize roundoff errors.

2.3.3 Large ν and fixed x

We introduce an asymptotic expansion which can be used effectively for $\nu \gg x$. The coefficients of the asymptotic expansion have a relatively simple representation, in contrast to the previous uniform asymptotic expansion. We start with the integral representation $\int_1^\infty e^{-xu}t^{-\nu}\,du$ applying a change of variable $e^t = \phi(u) = 1 + u$ to obtain

$$E_\nu(x) = e^{-x}\int_0^\infty e^{-\nu t} f(t)\,dt, \quad f(t) = e^{t-x(e^t-1)}.$$

$f(t)$ is analytic in $[0,\infty)$ and for $x > 0$ the integral is convergent, therefore we can apply Watson's lemma, obtaining the following asymptotic expansion

$$E_\nu(x) \sim e^{-x}\sum_{k=0}^{\infty} c_k \frac{k!}{\nu^{k+1}}, \quad \nu \to \infty, \quad f(t) = \sum_{k=0}^{\infty} c_k(x) t^k, \tag{51}$$

where c_k denote the coefficients of the Maclaurin expansion. The function $f(t)$ is a product of two exponential functions, which exponential generating functions are defined by

$$e^t = \sum_{k=0}^{\infty} \frac{t^k}{k!}, \quad e^{-x(e^t-1)} = \sum_{k=0}^{\infty} \frac{B_k(-x)}{k!} t^k, \tag{52}$$

where $B_k(x)$ are the Bell polynomials. The Bell polynomials have an explicit formula in terms of Stirling numbers of the second kind denoted as $S(k,j)$ and an infinite series known as Dobiński's formula, respectively

$$B_k(x) = \sum_{j=0}^{k} S(k,j) x^j, \quad B_k(x) = e^{-x}\sum_{j=0}^{\infty} \frac{j^k}{j!} x^j. \tag{53}$$

The coefficients c_k are combinations of the coefficients in both exponential generating functions in Equation (52), thus we obtain

$$c_k = \sum_{j=0}^{k} \frac{1}{j!}\frac{B_{k-j}(-x)}{(k-j)!} = \frac{1}{k!}\sum_{j=0}^{k} \binom{k}{j} B_{k-j}(-x). \tag{54}$$

Let us define the coefficients $d_k = c_k k!$. We observe that d_k is a recurrence formula for Bell polynomials given by $B_n(x) = x\sum_{k=1}^{n}\binom{n-1}{k-1}B_{k-1}(x)$, with $B_0(x) = 1$. Consequently, the coefficients d_k have the following representation, $d_k = -B_{k+1}(-x)/x$. Substituting these coefficients in Equation (51), the asymptotic expansion may be written in this form

$$E_\nu(x) \sim \frac{e^{-x}}{\nu}\sum_{k=0}^{\infty}\frac{d_k}{\nu^k} = -\frac{e^{-x}}{x}\sum_{k=0}^{\infty}\frac{B_{k+1}(-x)}{\nu^{k+1}}, \quad \nu \to \infty. \tag{55}$$

Note that the computation of Bell polynomials with negative argument x leads to substantial cancellation due to the evaluation of large magnitude alternating terms. In order

[6] Auxiliary data can potentially compromise thread safety in a multiple processor configuration, therefore it is convenient to avoid its usage.

to guarantee the required accuracy, the working precision needs to be increased to at least the exponent of the largest term involved.

Table 2 shows the required number of terms k to satisfy $|B_{k+1}(-x)|/\nu^{k+1} < 2^{-53}$ for several values of ν and x when $\nu \gg x$. Numerical experiments reveal that for moderate x not more than roughly 30 terms are needed when $\nu/x \gtrsim 3$.

ν	x	terms	ν	x	terms
20	2	28	1000	200	22
50	10	20	2000	40	10
100	30	21	5000	600	18
500	50	15	5000	10	6

Table 2: Minimum number of terms k to satisfy $|B_{k+1}(-x)|/\nu^{k+1} < 2^{-53}$ for the asymptotic expansion (55).

To determine the number of terms k needed to achieve a required precision 2^{-p}, it is practical to have an upper bound of the truncated term $|B_{k+1}(-x)|/\nu^{k+1}$, particularly to decide whether the asymptotic expansion can be used. In what follows, we proceed to derive an effective upper bound for Bell polynomials for $x \in \mathbb{R}$. In fact, by using Dobiński's formula, the computation of $B_k(x)$ generalizes to $k, x \in \mathbb{C}$, and so thus our upper bound.

An integral representation for Bell polynomials is obtained by direct application of Cauchy's integral formula to the exponential generating function with a parametrization $z(t) = e^{it}$, $t \in [0, 2\pi]$

$$B_n(x) = \frac{n!}{2\pi i}\int_{\mathcal{C}} \frac{e^{x(e^z-1)}}{z^{n+1}}\,dz = \frac{n!}{2\pi}\int_0^{2\pi} e^{x(e^{e^{it}}-1)}e^{-int}\,dt\,. \tag{56}$$

Equivalent formulas are given by

$$\begin{aligned} B_n(x) &= \frac{n!}{\pi}\Re\left(\int_0^{\pi} e^{x(e^{e^{it}}-1)}e^{-int}\,dt\right) \\ &= \frac{n!}{\pi e^x}\int_0^{\pi} e^{x(e^{\cos(t)}\cos(\sin(t)))}\cos(nt - x\sin(\sin(t))e^{\cos(t)})\,dt, \end{aligned}$$

where the latter integrand is the real part of $e^{x(e^{e^{it}})}e^{-int}$. In order to compute an effective upper bound for $B_n(x)$ we develop a saddle-point bound[7] [5]. Let us define the function $g(t)$ representing the integrand in Equation (56) and its derivative with respect to t,

$$g(t) = e^{x(e^{e^{it}}-1)-int}, \quad g'(t) = e^{x(e^{e^{it}}-1)-int}(ixe^{it+e^{it}} - in). \tag{57}$$

The saddle-point t_0 is the point such that $g'(t_0) = 0$, which is given by

$$t_0 = -i(\log(n/x) - W(n/x)), \tag{58}$$

where $W(x)$ is the Lambert-W function which solves $W(x)e^{W(x)} = x$. Substituting t_0 in $g(t)$ we obtain the principal contribution of the bound. It remains to compute the term λ representing the length of the path of the contour joining $[0, 2\pi]$ through t_0 that minimizes $|g(t_0)|$

$$|g(t_0)| = \left|\frac{n!}{2\pi e^x}\frac{e^{xe^{W(n/x)}}}{W(n/x)^n}\right|, \quad \lambda = |0 - t_0| + |2\pi - t_0|. \tag{59}$$

Thus, the resulting upper bound for the Bell polynomials is given by

$$|B_n(x)| \le \lambda\left|\frac{n!}{2\pi e^x}\frac{e^{xe^{W(n/x)}}}{W(n/x)^n}\right|, \tag{60}$$

[7]Saddle-point bound: $|\int_A^B f(t)\,dt\,| \le |C_0||f(t_0)|$, $f'(t_0) = 0$. $|C_0|$ are saddle-point paths made of arcs connecting A and B through the saddle-point t_0.

which as aforementioned can be generalized replacing the factorial by the gamma function. Table 3 shows the closeness of the upper bound (3) for moderate and large values of $n \in \mathbb{N}$ and $x \in \mathbb{R}$.

n	x	$\lvert B_n(x)\rvert$	Bound (60)	n	x	$\lvert B_n(x)\rvert$	Bound (60)
30	20	4.01e+44	7.65e+45	30	-20	1.38e+33	1.69e+35
10	200	1.27e+23	1.66e+24	100	-200	8.12e+22	8.66e+23
500	1000/3	1.53e+1356	1.19e+1358	500	-1000/3	1.16e+1179	5.36e+1180

Table 3: Upper bound for Bell polynomials $B_n(x)$ for $x \in \mathbb{R}$.

For $x = 1$, $B_n(1) = B_n$ is the nth Bell number[8]. As shown in Table 4, in turns out that bound (60) is sharper than other upper bounds for Bell numbers recently established in [2], especially for moderate and large n, and given by

$$B_n < \left(\frac{0.792n}{\log(n+1)}\right)^n \tag{61}$$

n	B_n	Bound (61)	Bound (60)	n	B_n	Bound (61)	Bound (60)
50	1.9e+47	1.4e+50	7.7e+48	1000	3.0e+1927	2.1e+2059	7.7e+1929
100	4.8e+115	2.9e+123	3.0e+117	10000	1.6e+27664	2.8e+29344	1.6e+27667
500	1.6e+843	1.2e+902	2.7e+845	100000	1.0e+364471	8.2e+383753	3.8e+364474

Table 4: Upper bound for Bell numbers B_n.

3 Other numerical methods

This section presents other numerical methods for the evaluation of the generalized exponential integral. Several of these numerical methods are used in other available implementations, even though we do not employ them in our proposed algorithm in Section 4, due to the existence of more robust and/or faster methods in the same domains of computation introduced in Section 2.

3.1 Factorial series

Factorial series are considered an alternative for the summation of divergent inverse power series. The method is a useful numerical tool that can be used for functions defined in terms of Laplace integral, for example integral (41), with which we proceed by applying a change of variable $e^{-t} = w$ and $\varphi(w) = (1 - \log(w))^{-\nu}$ to transform into a convergent expansion

$$E_\nu(x) = e^{-x} \sum_{k=0}^{\infty} \frac{a_k k!}{(x)_{k+1}}, \quad \varphi(w) = \sum_{k=0}^{\infty} a_k (1-w)^k, \tag{62}$$

where a_k are the coefficients of the Maclaurin series of $\varphi(w)$ at $w = 1$. In particular, the first 7 coefficients are

$$a_0 = 1, \quad a_1 = -\nu, \quad a_2 = \frac{1}{2}\nu^2, \quad a_3 = -\frac{1}{6}\nu(\nu^2+1), \quad a_4 = \frac{1}{24}\nu(\nu^3+4\nu-1),$$
$$a_5 = -\frac{1}{120}\nu(\nu^4+10\nu^2-5\nu+8), \quad a_6 = \frac{1}{720}\nu(\nu^5+20\nu^3-15\nu^2+58\nu-26).$$

This factorial series exhibits fast convergence for moderate values of ν and x when $x > \nu$, and it generally outperforms the asymptotic expansion. However, Laguerre series (22) tends to converge more rapidly, whereby factorial series was not included in the proposed algorithm. For an introduction to factorial series we refer to [20, §2.4.4].

[8] Bell numbers represent the number of class-partitions of a finite set with n elements.

3.2 Continued fractions

The Stieltjes fraction (S-fraction) [13, §14.1.16] is given by

$$E_\nu(x) = e^{-x}\left(\frac{1/x}{1+}\ \underset{m=2}{\overset{\infty}{\mathrm{K}}}\left(\frac{a_m/x}{1}\right)\right), \quad x>0,\ \nu\in\mathbb{R}^+, \tag{63}$$

where $a_{2j} = j+n-1$, $a_{2j+1} = j$, $\quad j \geq 1$. Since $\lim_{m\to\infty} a_m = +\infty$, the modification [13, §7.7]

$$w_{2k}(x) = \frac{-1+\sqrt{4kx^{-1}+1}}{2}, \quad w_{2k+1}(x) = \frac{-1+\sqrt{4(n+k)x^{-1}+1}}{2}. \tag{64}$$

The S-fraction is evaluated using a forward recursion algorithm based on the three-term recurrence relations. This algorithm requires successive rescaling to avoid numerical difficulties. Cephes library implementation includes the S-fraction for the domain $x > 1$ and $\nu \leq 5000$, without modification.

The C-fraction [13, §14.1.19] is given by

$$E_\nu(x) = e^{-x}\ \underset{m=1}{\overset{\infty}{\mathrm{K}}}\left(\frac{a_m(\nu)x^{-1}}{1}\right), \quad x>0,\ \nu\in\mathbb{C}, \tag{65}$$

where the coefficients are given by

$$a_1(\nu) = 1, \quad a_{2j}(\nu) = j+\nu-1, \quad a_{2j+1}(\nu) = j, \quad j\in\mathbb{N}. \tag{66}$$

The Jacobi fraction (J-fraction) [13, §14.1.23], obtained by taking the even part of the C-fraction is given by

$$E_\nu(x) = e^{-x}\left(\frac{1}{\nu+x+}\ \underset{m=2}{\overset{\infty}{\mathrm{K}}}\left(\frac{(1-m)(\nu+m-2)}{\nu+2m-2+x}\right)\right), \quad x>0,\ \nu\in\mathbb{C}. \tag{67}$$

The C-fraction and J-fraction are evaluated using the modified Lentz algorithm, which is implemented taking into account the suggestions in [20, §6.6.2] to improve numerical robustness. Table 5 shows the number of terms and relative error for each continued fraction for regions where asymptotic expansions do not apply due to the amount of terms required. The last column corresponds to the Laguerre series.

ν	x	(65)	(67)	(63)	(22)
2.3	1.6	142 (0)	63 (9e−16)	117 (0)	67 (0)
0.3	5.6	52 (9e−16)	23 (9e−16)	41 (3e−16)	21 (0)
10.3	15.6	32 (4e−16)	16 (1e−16)	28 (1e−16)	14 (1e−16)
100.3	15.6	26 (7e−16)	13 (3e−16)	25 (1e−16)	10 (4e−16)
10.3	150.6	14 (7e−16)	8 (0)	13 (2e−16)	6 (2e−16)
100.3	150.6	18 (1e−16)	10 (1e−16)	17 (1e−16)	7 (1e−16)

Table 5: Comparison of different continued fractions and Laguerre series, number of terms and relative errors. Precision is set to 53-bit.

The results confirm the significant superiority of the Laguerre series with respect to both C-fraction and S-fraction. On the other hand, the J-fraction exhibits rapid convergence, but some loss of precision is observed for small values of ν and x. We refer the reader, e.g., to Cuyt *et al.* [13] for a theoretical background and numerical methods for continued fractions.

3.3 Numerical integration

From the definition of the generalized exponential integral (2), we apply a change of variable $t = x + q$ to obtain

$$E_\nu(x) = x^{\nu-1}e^{-x}\int_0^\infty e^{-q}(x+q)^{-\nu}\,dq, \tag{68}$$

which can be directly evaluated by means of Gauss-Laguerre quadrature. Furthermore, the integrand has the decay property as a single exponential function $q \to \infty$, therefore we can exploit this by applying a double-exponential transformation (DE-transformation) to an integral over a half-infinite interval. Then,

$$E_\nu(x) = x^{\nu-1}e^{-x}\int_{-\infty}^\infty e^{-\phi(t)}(x+\phi(t))^{-\nu}\phi'(t)\,dt, \tag{69}$$

where $q = \phi(t) = e^{-t-e^{-t}}$, and $\phi'(t) = (1+e^{-t})\phi(t)$. Another possible change of variable is $\phi(t) = \pi/2\sinh(t)$ and $\phi'(t) = \pi/2\cosh(t)\phi(t)$, although the latter does not provide better results in our experiments. We truncate the infinite summation at $k = -n$ and $k = n$, where the total number of function evaluations is $N = 2n+1$ using the trapezoidal rule with equal mesh size,

$$E_\nu(x)^{(n,h)} = x^{\nu-1}e^{-x}h\sum_{k=-n}^{n} e^{-\phi(kh)}(x+\phi(kh))^{-\nu}\phi'(kh). \tag{70}$$

Two methods of computation are used for the evaluation of above integrals; the extended trapezoidal rule and Ooura's implementation for DE-transformation over half-infinite interval [32]. For the extended trapezoidal rule we use symmetric truncations at ± 6, which performs well for moderate values of ν and x.

3.3.1 Other integrals

The integral in §2.3.3, can be computed effectively using numerical quadrature methods, since the integrand decays exponentially for ν and double exponentially for z. If we start with $\lambda = x/\nu$, where $\lambda > 0$, we can write the integral as follows

$$E_\nu(x) = e^{-x}\int_0^\infty e^{-\nu\lambda t}(1+t)^{-\nu}\,dt\,. \tag{71}$$

Now by applying a change of variable $e^u/\lambda = \phi(t) = 1+t$, $e^u/\lambda\,du = dt$ we obtain

$$E_\nu(x) = \lambda^{\nu-1}\int_{\log(\lambda)}^\infty e^{-\nu e^u}e^{-u(\nu-1)}\,du, \tag{72}$$

where the integrand only depends on ν and it is entirely non-increasing. Our experimental results showed that computing these integrals by using the extended trapezoidal rule was 1.5-4 times slower than other available methods in the same domain of computation, consequently these were discarded.

4 Algorithm and implementation

We have devised an accurate algorithm along with an efficient implementation in double-precision floating-point arithmetic of $E_\nu(x)$ for integer and real order ν. The program[9] written in C++ is about 800 lines of code and includes python bindings. It is released under MIT licence.

Our implementation allows the use of internal computations using higher precision arithmetic implemented in software, in particular we use the so-called *error free transformations*

[9] `https://sites.google.com/site/guillermonavaspalencia/software/expint.zip`

and double-double numbers, for difficult regions prone to numerical instability, thus diminishing the effect of round off errors. This approach is particularly intended for the evaluation of series expansions, where cancellation errors in the regime $x \in [1, 1.5)$ occur.

A double-double (DD) number is a *multiple-term representation* in which a number is expressed as the unevaluated sum of two standard floating-point (FP) numbers. The DD number is capable of representing at least 106-bit of significant, roughly 31 digits of accuracy and is, therefore, similar to IEEE 754 quadruple-precision. A reference library using this approach is Bailey's library[10] QD [23]. There are several reasons for using DD numbers instead of quadruple-precision (e.g., using libquadmath included in GCC): operations with DD numbers use highly optimized hardware implementation of floating-point operations, and quadruple-precision is still not available for all compilers and programming languages, which would limit the implementation of the algorithm.

An error free transformation (EFT) is an algorithm which transforms any arithmetic operation of two FP numbers a and b into a sum of two FP numbers s and t, a floating-point approximation and an exact error term, respectively. Therefore, these algorithms keep track of all accumulated rounding errors, avoiding the lost of information. The basic brick for our implementation is Algorithm 3. This algorithm requires Algorithm 1 [28], which computes the exact sum of two FP numbers and returns the result under s and t. It requires 6 native FP operations (*flops*).

Algorithm 1 Error-free transformation of the sum of two floating-point numbers.

Input: a, b
Output: s, t
1: **function** TwoSum(a, b)
2: $\quad s \leftarrow \mathrm{RN}(a + b)$ ▷ RN: Rounding to nearest mode.
3: $\quad c \leftarrow \mathrm{RN}(s - a)$
4: $\quad t \leftarrow \mathrm{RN}(\mathrm{RN}(a - \mathrm{RN}(s - c)) + \mathrm{RN}(b - c))$
5: **end function**

Furthermore, it also uses Algorithm 2 [14], requiring 3 flops. This algorithm is applicable when the exponent of a is larger or equal to that of b.

Algorithm 2 Error-free transformation of the sum of two floating-point numbers ($|a| \geq |b|$).

Input: a, b
Output: s, t
1: **function** FastTwoSum(a, b)
2: $\quad s \leftarrow \mathrm{RN}(a + b)$
3: $\quad z \leftarrow \mathrm{RN}(s - a)$
4: $\quad t \leftarrow \mathrm{RN}(b - z)$
5: **end function**

Algorithm 3 computes the exact sum of a DD number and a FP number, storing the resulting operation into the DD number, performing an operation in-place. This algorithm is used to accumulate the intermediate summation of terms in series expansions (16), (20) and (38) and for the final subtraction or addition operation. Although the use of EFTs for every single operation would definitely enhance the accuracy of the algorithm, we aim to provide a good balance between achievable accuracy and computational time, so that our implementation is competitive with other software packages only using FP operations. Other alternatives to reduce cancellations consist of grouping two consecutive terms in descending order, so the subtraction of the second term does not produce cancellation. However, given the satisfactory results obtained with EFTs, we discarded those methods.

[10]The operations implemented in this library are not compliant with the IEEE 754-2008 standard.

Algorithm 3 Addition of a double-double number and a double number in-place.

Input: sh, sl, a
Output: sh, sl
1: **function** ADD_DD_D_IP(sh, sl, a)
2: $(th, tl) \leftarrow$ TWOSUM(sh, a)
3: $tl \leftarrow \mathrm{RN}(th + tl)$
4: $(sh, sl) \leftarrow$ FASTTWOSUM(th, tl)
5: **end function**

Finally, our algorithm uses some mathematical functions from the standard library defined in `<cmath>`, for instance `tgamma` and `lgamma`, which compute the gamma function and the natural logarithm of the absolute value of the gamma function, respectively.

4.1 Algorithm for integer order

The algorithm of integer order $\nu \equiv n, n \in \mathbb{N}$ combines asymptotic expansions, Laguerre series, series expansions and Chebyshev approximations. Laguerre series is the dominant method for $x > 2$ and it is used as a backup method wherever asymptotic expansions are not applicable. For small x series expansions are employed, since they show faster convergence and return more accurate results than Laguerre series. For the special case $n = 1$ we use the Chebyshev approximations in [11], which require fewer terms and provide more accurate results for moderate values of x. Note that we avoid the evaluation of $\mathrm{Ei}(-x)$ in the vicinity of $x_0 \approx -0.372507$, which corresponds to a single zero.

Algorithm 4 Algorithm for $E_n(x)$, $n \in \mathbb{N}$ and $x > 0$

Input: $n \in \mathbb{N}$, $x \in \mathbb{R}$
Output: $E_n(x)$
1: **if** $n == 1$ **and** $x \in (0.9, 10)$ **then**
2: compute $E_1(x) = -\mathrm{Ei}(-x)$ using Chebyshev approximations [11]
3: **else if** $x \leq 1.5$ **and** $n < 20$ **then**
4: series expansion (38)
5: **else if** $x \leq 2.0$ and $n \leq 10$ **then**
6: series expansion (39) ▷ Faster than Laguerre series
7: **else if** $n \geq x$ **then**
8: **if** $x < 5$ **then**
9: check if asymptotic expansion (55) can be used, otherwise Laguerre series (22)
10: **end if**
11: Laguerre series (22) ▷ Backup method
12: **else**
13: **if** $x/\nu > 100$ **then**
14: check if asymptotic expansion (43) can be used, otherwise Laguerre series (22)
15: **end if**
16: Laguerre series (22) ▷ Backup method
17: **end if**

4.2 Algorithm for real order

The algorithm for real order differs on the computational methods applied for small x. The alternating series (16) converges faster and it is used for large ν/x and as a backup method. Series (20) is used when the value ν guarantees that all terms of the expansion are positive. Finally, Laguerre series (22) is employed in regions where results returned by series expansions are not sufficiently accurate.

Algorithm 5 Algorithm for $E_\nu(x)$, $\nu \in \mathbb{R}^+$ and $x > 0$

Input: $\nu \in \mathbb{R}^+$, $x > 0$
Output: $E_\nu(x)$

if $x \leq 1$ **and** $x < 20$ **then**
  if $\nu/x > 10$ **then**
    series expansion (16) ▷ Fast convergence
  end if
  if $\nu > 1.5$ and $x > 0.5$ **then**
    Laguerre series (22) ▷ Slow but more accurate
  else if $\nu < 0.9$ **then**
    series expansion (20) ▷ All terms of expansion are positive
  else
    series expansion (16) ▷ Backup method
  end if
else if $\nu \geq x$ **then**
  if $x < 5$ **then**
    check if asymptotic expansion (55) can be used, otherwise Laguerre series (22)
  end if
  Laguerre series (22) ▷ Backup method
else
  if $x/\nu > 100$ **then**
    check if asymptotic expansion (43) can be used, otherwise Laguerre series (22)
  end if
  Laguerre series (22) ▷ Backup method
end if

5 Benchmarks

Publicly available implementations of the generalized exponential integral in double-precision arithmetic are Cephes [30], Boost [3] and GNU Scientific Library (GSL) [17]. These libraries provide implementations for the special case $\nu \equiv n, n \in \mathbb{N}$. To our knowledge, there are no numerical libraries in double-precision arithmetic implementing $E_\nu(x)$ for real values of ν. We compare our implementation to the aforementioned software for ν integer and to mpmath [26] and Arb [24], both supporting arbitrary-precision arithmetic, for real ν.

To compare our implementation to other software packages we use performance profiles. Performance profiles [16] are widely used tools for benchmarking and evaluating the performance of several solvers, particularly in the fields of optimization and linear algebra, when run on a large test set. Performance profiles provide a convenient procedure of assessing the performance of a code relative to the best code of the set.

Regarding the other codes, Cephes and Boost use similar algorithms, both use continued fractions as a main method and the power series for small x. In addition, Cephes includes the uniform asymptotic expansion in (45) for $n > 5000$. The implementation in GSL is purely based on the applications of the functional relation with the incomplete gamma function (3). As shown in Figure 2 and Table 6, this simplistic approach in double-precision has several limitations, especially for large n, as the number of failures indicate.

Figure 2 compares all four codes in terms of the relative error using as a reference the value computed by mpmath with 1000 digits of precision. If relevant discrepancies arise using different levels of precision, we use the result from Arb or Mathematica, which tend to be more reliable. The test samples are generated non-uniformly, in fact we select certain input values around regions of transition between computational methods, in order to test the worst cases. Figure 3 shows a comparison in terms of computation time. All measurements were obtained by evaluating each test sample 100 times are returning the average time.

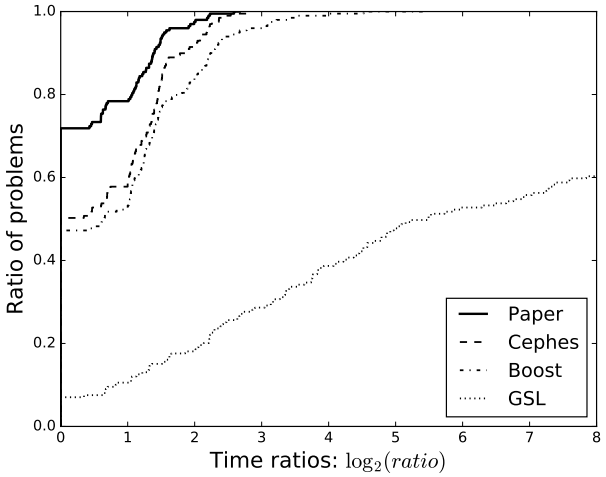

Figure 2: Accuracy profiles case $n \in \mathbb{N}$ and $x > 0$.

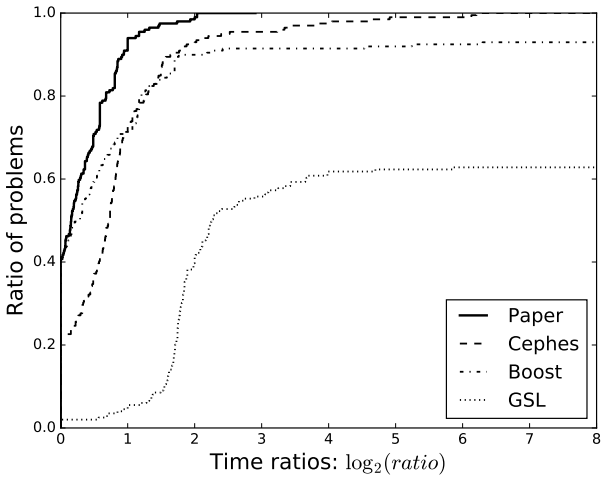

Figure 3: Performance profiles case $n \in \mathbb{N}$ and $x > 0$.

Observing the performance profiles and the statistics in Table 6, it seems safe to claim that our algorithm outperforms the other available codes. In terms of computation time, Boost comes very close in median, however, for some large values of n the computation time is ridiculously high. Cephes does not include an asymptotic expansion for large $n < 5000$, so cases where n is large and $x < 1$ require the computation of n terms, being prohibitively expensive. Nevertheless, Cephes implements a more effective computation scheme than Boost, the former being possibly improved by replacing the S-fraction by the J-fraction, as shown in Section 3.

Library	Max. error	Avg. error	Avg. time (μs)	Stdev. time (μs)	fails
Paper	**9.7e−16**	**1.3e−16**	**0.25**	**0.21**	**0/200**
Cephes	1.4e−15	2.0e−16	0.73	2.47	**0/200**
Boost-1.61.0	4.8e−15	3.3e−16	63.76	558.36	**0/200**
GSL-2.2.1	5.2e−14	6.1e−15	1.34	1.19	75/200

Table 6: Error statistics for each library. gcc-5.4.0 compiler running Cygwin. Time in microseconds. Fails: returns Incorrect/NaN/Inf. Intel(R) Core(TM) i5-3317 CPU at 1.70GHz.

For real order ν we have generated two sample sets with the following characteristics:

- Large set: $\nu \in [0.0, 10000]$ and $x \in [10^{-9}, 1000]$
- Small set: $\nu \in [0.04, 70]$ and $x \in [0.00075, 1.5]$

The large set was generated to test the accuracy of asymptotic expansions and Laguerre series, whereas the small set is testing the region in x where a loss of significant digits is expected. As shown in Table 7, the maximum relative error determines that about 5 bits of precision might be lost. These results suggest that the computation in this region could be improved by re-implementing both series expansions using DD operations, at the cost of worsening the computation time.

Library	Max. error	Avg. error	Avg. time (μs)	Stdev. time (μs)	fails
Large set	9.8e−16	1.1e−16	0.14	0.10	0/1500
Small set	3.1e−15	1.7e−16	0.52	0.37	0/500

Table 7: Error statistics for each library. gcc-5.4.0 compiler running Cygwin. Time in microseconds. Fails: returns Incorrect/NaN/Inf. Intel(R) Core(TM) i5-3317 CPU at 1.70GHz.

5.1 Arbitrary-precision floating-point libraries

We evaluate the implementation of the generalized exponential integral in the arbitrary-precision packages Arb 2.8 and mpmath 0.19. Test were run on an Intel(R) Core(TM) i7-6700HQ CPU at 2.60GHz. For testing Arb we use Sage 7.3. Both software packages implement the generalized exponential integral using the functional relation with the incomplete gamma function, which is implemented using the series expansion of the confluent hypergeometric function for small x and the asymptotic expansion of the U-Kummer function for large x, for more details refer to [25]. Table 8 summarizes the results obtained by mpmath for all three test sets using 53-bit of precision. Mpmath automatically chooses guard bits to achieve the requested accuracy, however it is usually unable to return a correct results for large parameters. Additionally, mpmath doest not return a flag indicating that the result is incorrect.

Set	Max. error	Avg. error	Avg. time (μs)	Stdev. time (μs)	fails
Integer	1.0	4.4e−13	125	243	5/200
Large real	1.2e+163	2.2e−15	16397	128024	75/1500
Small real	0.0	0.0	672	187	0/500

Table 8: Error statistics `mpmath` library. Average error is computed after excluding relative errors $\geq$ 1e−10. A result is considered wrong if relative error is $>$ 1e−14.

Arb uses interval arithmetic and efficiently tracks errors. The usage of the functional relation works reasonably well in most of the cases, but as shown below for large parameters one needs to increase the working precision considerably to obtain a solution with 16

significant digits. For example, evaluating $E_{500.25}(400)$ at 53, 1000 and 1210 bits with Arb produces:

```
sage: CBF = ComplexBallField(53)
sage: CBF(400).exp_integral_e(CBF(500+1/4))
nan + nan*I
sage: CBF = ComplexBallField(1000)
sage: CBF(400).exp_integral_e(CBF(500+1/4))
[+/- 5.05e-130]
sage: CBF = ComplexBallField(1210)
sage: CBF(400).exp_integral_e(CBF(500+1/4))
[2.128687916150507e-177 +/- 6.64e-194]

sage: %timeit CBF(400).exp_integral_e(CBF(500+1/4))
1000 loops, best of 3: 1.15 ms per loop
```

Hence, it is necessary to systematically increase the precision in order to obtain results at the desired accuracy. On the other hand, our implementation uses the Laguerre series in this domain reporting fast convergence (5 terms), see below:

```
In [1]: import expint
In [2]: expint.expint_v(500+1/4,400)
Out[2]: 2.128687916150507e-177

In [3]: %timeit expint.expint_v(500+1/4,400)
1000000 loops, best of 3: 475 ns per loop
```

Finally, we encourage the use of more sophisticated computation scheme and addition of specific numerical methods for the computation of the generalized exponential integral in arbitrary-precision arithmetic.

6 Conclusions

In this paper, we proposed an efficient algorithm for the computation of the generalized exponential integral. The algorithm includes a new asymptotic expansion for large order ν and other methods not implemented in existing software. Numerical experiments confirmed the benefits of using internal higher precision arithmetic for regions where numerical instability appears, thus obtaining more reliable results. This resulted in an implementation capable of outperforming available software packages in terms of accuracy and computation time.

We believe the improvements and suggested numerical methods in this paper should be considered for inclusion in arbitrary-precision arithmetic software packages, which in general implement simplistic computation schema and rely on continuously increasing the working precision to obtain a solution satisfying the user's precision. Finally, our implementation in C++ was made freely available.

7 Acknowledgement

The author acknowledges Argimiro Arratia for useful advice while this manuscript was being prepared and Javier Segura for reading the preliminary version of the manuscript and providing useful remarks. The author also thanks the anonymous reviewer for indicating errors and suggesting improvements.

References

[1] Z. Altaç. Integrals involving Bickley and Bessel functions in radiative transfer, and generalized exponential integral functions. *J. Heat Transfer*, 118(3):789–792, 1996.

[2] D. Berend and T. Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

[3] Boost. Boost C++ Libraries. `http://www.boost.org/`, 2016. Last accessed 2016-12-29.

[4] D. Borwein, J. M. Borwein, and R. E. Crandall. Effective laguerre asymptotics. *SIAM J. Numerical Analysis*, 46(6):3285–3312, 2008.

[5] N.G. de Bruijn. *Asymptotic Methods in Analysis*. Bibliotheca mathematica. Dover Publications, 1970.

[6] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960.

[7] C. Chiccoli, S. Lorenzutta, and G. Maino. A numerical method for generalized exponential integrals. *Comput. Math. Appl.*, 14(4):261–268, 1987.

[8] C. Chiccoli, S. Lorenzutta, and G. Maino. An algorithm for exponential integrals of real order. *Computing*, 45(3):269–276, 1990.

[9] C. Chiccoli, S. Lorenzutta, and G. Maino. Recent results for generalized exponential integrals. *Comput. Math. Appl.*, 19(5):21–29, 1990.

[10] C. Chiccoli, S. Lorenzutta, and G. Maino. Concerning some integrals of the generalized exponential-integral function. *Comput. Math. Appl.*, 23(11):13–21, 1992.

[11] W. J. Cody and H. C. Thacher, Jr. Chebyshev approximations for the exponential integral $Ei(x)$. *Math. Comp.*, 23(106):289–303, 1969.

[12] H. Cohen, F. Rodriguez Villegas, and D. Zagier. Convergence acceleration of alternating series. *Experiment. Math.*, 9(1):3–12, 2000.

[13] A. Cuyt, V. Petersen, B. Verdonk, H. Waadeland, W. B. Jones, and C. Bonan-Hamada. *Handbook of Continued Fractions for Special Functions*. Kluwer Academic Publishers Group, Dordrecht, 2007.

[14] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math.*, 18(3):224–242, 1971.

[15] *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.0.14 of 2016-12-21. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller and B. V. Saunders, eds.

[16] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[17] Galassi et al. *Gnu Scientific Library: Reference Manual (3rd Ed.)*. Network Theory Ltd., 2003.

[18] W. Gautschi. Exponential integral $\int_1^\infty e^{-xt}t^{-n}dt$ for large values of n. *J. Res. Nat. Bur. Standards*, 62:123–125, 1959.

[19] A. Gil, D. Ruiz-Antolín, J. Segura, and N. M. Temme. Algorithm 969: Computation of the incomplete gamma function for negative values of the argument. *ACM Trans. Math. Softw.*, 43(3):26:1–26:9, November 2016.

[20] A. Gil, J. Segura, and N. M. Temme. *Numerical methods for special functions*. SIAM, 2007.

[21] A. Gil, J. Segura, and N. M. Temme. Efficient and accurate algorithms for the computation and inversion of the incomplete gamma function ratios. *SIAM J. Scientific Computing*, 34(6), 2012.

[22] P. Henrici. *Applied and Computational Complex Analysis. Vol. 2: Special FunctionsIntegral TransformsAsymptoticsContinued Fractions*. Wiley-Interscience [John Wiley & Sons], New York, 1977. Reprinted in 1991.

[23] Y. Hida, X. S. Li, and D. H. Bailey. Algorithms for quad-double precision floating point arithmetic. In *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001*, pages 155–162, 2001.

[24] F. Johansson. Arb: A c library for ball arithmetic. *ACM Commun. Comput. Algebra*, 47(3/4):166–169, 2014.

[25] F. Johansson. Computing hypergeometric functions rigorously. working paper or preprint, 2016.

[26] F. Johansson et al. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.19)*, December 2014. `http://mpmath.org/`.

[27] J. D. Kečlić and P. M. Vasić. Some inequalities for the gamma function. *Publ. Inst. Math. (Beograd) (N. S.)*, 11:107–114, 1971.

[28] D. E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[29] J. LeCaine. *A Table of Integrals Involving the Functions $En(x)$*. N.R.C. (National Research Council). National Research Council Canada, Division of Atomic Energy, 1949.

[30] S. L. Moshier. Cephes mathematical function library, 2000.

[31] A. B. Olde Daalhuis. Hyperasymptotic expansions of confluent hypergeometric functions. *IMA Journal of Applied Mathematics*, 49:203–216, 1992.

[32] T. Ooura and M. Mori. The double exponential formula for oscillatory functions over the half infinite interval. *Journal of Computational and Applied Mathematics*, 38(1):353–360, 1991.

[33] G. N. Watson. The harmonic functions associated with the parabolic cylinder. *Proceedings of the London Mathematical Society*, s2-17(1):116–148, 1918.