Fast and accurate evaluation of a generalized incomplete gamma function

Rémy Abergel and Lionel Moisan

Université de Paris, MAP5, CNRS, F-75006 Paris - France.

Abstract

We present a computational procedure to evaluate the integral $\int_x^y s^{p-1}\,e^{-\mu s}\,ds$ for $0 \leq x < y \leq +\infty$, $\mu = \pm 1$, $p > 0$, which generalizes the lower ($x = 0$) and upper ($y = +\infty$) incomplete gamma functions. To allow for large values of x, y, and p while avoiding under/overflow issues in the standard double precision floating point arithmetic, we use an explicit normalization that is much more efficient than the classical ratio with the complete gamma function. The generalized incomplete gamma function is estimated with continued fractions, integrations by parts, or, when $x \approx y$, with the Romberg numerical integration algorithm. We show that the accuracy reached by our algorithm improves a recent state-of-the-art method by two orders of magnitude, and is essentially optimal considering the limitations imposed by the floating point arithmetic. Moreover, the admissible parameter range of our algorithm ($0 \leq p, x, y \leq 10^{15}$) is much larger than competing algorithms and its robustness is assessed through massive usage in an image processing application.

Keywords: Incomplete gamma function, incomplete gamma integral, continued fraction, numerical cancellation, Romberg's method.

1 Introduction

In this work, we focus on the computation of a generalized incomplete gamma function that will be defined below. Let us first recall the definition of Euler's gamma function,

$$\forall p > 0, \quad \Gamma(p) = \int_0^{+\infty} s^{p-1}\,e^{-s}\,ds\,. \tag{1}$$

The lower and upper incomplete gamma functions are respectively obtained by allowing the integration domain to vary in (1),

$$\forall p > 0, \quad \forall x \geq 0, \quad \gamma(p,x) = \int_0^{x} s^{p-1}\,e^{-s}\,ds \quad \text{and} \quad \Gamma(p,x) = \int_x^{+\infty} s^{p-1}\,e^{-s}\,ds\,. \tag{2}$$

The gamma function is usually viewed as an extension of the factorial function since it satisfies $\Gamma(p) = (p-1)!$ for any positive integer p. Note that the gamma function can also be defined for all complex numbers p with positive real part, using the same convergent improper integral as in (1), and can be extended by analytic continuation to all complex numbers except the nonpositive integers, that is, to $p \in \mathbb{C} \setminus \{0, -1, -2, -3, \dots\}$.

These special functions arise in many areas, such as astronomy and astrophysics Cannon and Vardavas [1974], Hills [1975], Collins [1989], Rayleigh scattering Kissel et al. [1980], quantum gravity Bleicher and Nicolini [2010], networks Moreno et al. [2002], financial mathematics Linetsky [2006], image analysis Robin et al. [2010], etc. (see Chaudhry and Zubair [2001] for more examples). From the mathematical viewpoint, the computation of incomplete gamma functions is typically required in applications involving the evaluation of χ^2 distribution functions, exponential integrals, error functions (erf), cumulative Poisson or Erlang distributions. Their practical numerical evaluation is still subject to some flourishing research in the modern literature. The first algorithm dedicated to the numerical evaluation of the incomplete gamma functions was, to the best of our knowledge, proposed in Bhattacharjee [1970], and later in Press et al. [1992]. It evaluates the ratio $\gamma(p,x)/\Gamma(p)$ using a

series expansion when $0 < p \leq x < 1$ or $0 \leq x < p$, or the ratio $\Gamma(p,x)/\Gamma(p)$ using a continued fraction in the remaining part of the whole domain $\{x \geq 0, p > 0\}$. Note that since

$$\gamma(p,x) + \Gamma(p,x) = \Gamma(p), \tag{3}$$

one of the incomplete gamma functions can be deduced from the other as soon as we assume that $\Gamma(p)$ is known. Gautschi [1979] proposed another computational procedure, based on Taylor's series and continued fractions, to evaluate those two functions in the region $\{x \geq 0, p \in \mathbb{R}\}$ (in fact, for $p \leq 0$, Tricomi's version Tricomi [1950], Gautschi [1998] of the lower incomplete gamma function, which remains real for any real numbers x, p, is considered). The criterion proposed in Bhattacharjee [1970] to decide which one of the two integrals should be computed according to the value of (x,p) is refined, and a more suitable normalization is employed, which slightly extends the range over which those two functions can be represented within standard double precision arithmetic. More recently, Winitzki [2003] focused on the computation of the upper incomplete gamma function and used some series expansions, a continued fraction (due to Legendre), some recurrence relations, or, for large values of x, an asymptotic series. The precision of the approximation is controlled by estimating the number of terms required to reach a given absolute precision according to the values of x and p. However, the study is not considered from a practical point of view, and no algorithm or experimental validation are provided to assess the numerical stability of the proposed method. In Guseinov and Mamedov [2004], the lower and upper incomplete gamma functions are computed using backward and forward recurrence relations ; however, we experimentally noticed that a faster convergence was achieved with continued fractions. More recently, Gil et al. [2012] proposed new algorithms to compute the lower and upper ratios $\gamma(p,x)/\Gamma(p)$ and $\Gamma(p,x)/\Gamma(p)$, and to solve for x given one of these ratios. The ratios are computed using Taylor expansions, continued fractions or uniform asymptotic expansions, depending on the values of p and x. They claim a relative accuracy of $7.9 \cdot 10^{-13}$ on the domain $(0, 500]^2$, which we found a bit optimistic, as we shall see in Section 2.5.

One difficulty encountered in the numerical evaluation of Gamma functions (incomplete or not) is their exponential growth that easily causes numerical overflow. Thus, before designing an algorithm, it is necessary to choose an appropriate normalization to avoid this phenomenon. A classical normalization, used in most of the above-mentioned algorithms, computes the lower and upper incomplete Gamma function ratios

$$P(p,x) = \frac{\gamma(p,x)}{\Gamma(p)} \qquad \text{and} \qquad Q(p,x) = \frac{\Gamma(p,x)}{\Gamma(p)}. \tag{4}$$

Since P and Q are nonnegative and satisfy $P + Q = 1$ (as a consequence of (3)), we have $P, Q \leq 1$ so that no overflow can occur with these ratios. However, this normalization is not really satisfactory either because it produces severe underflow. When one of p or x is small and the other grows, $\min(P,Q)$ rapidly decays far below the smallest positive floating point number (around 10^{-308} in IEEE 754 double precision standard arithmetic), so that the best possible numerical approximation, 0, produces a relative error of 100%. A simple example is $P(200,1) \simeq 10^{-375}$. In other terms, any algorithm estimating the gamma integrals (2) from a computation of P and Q in double precision arithmetic will fail in a subpart of the (p,x) plane. On the domain $(0, 500]^2$, the value of $\min(P,Q)$ is representable for only 90% of the domain. On the domain $(0, 10^5]^2$, this ratio drops to less than 15%, and rapidly approaches 0% when the domain grows further. This "underflow region" is represented in Fig. 1 (a).

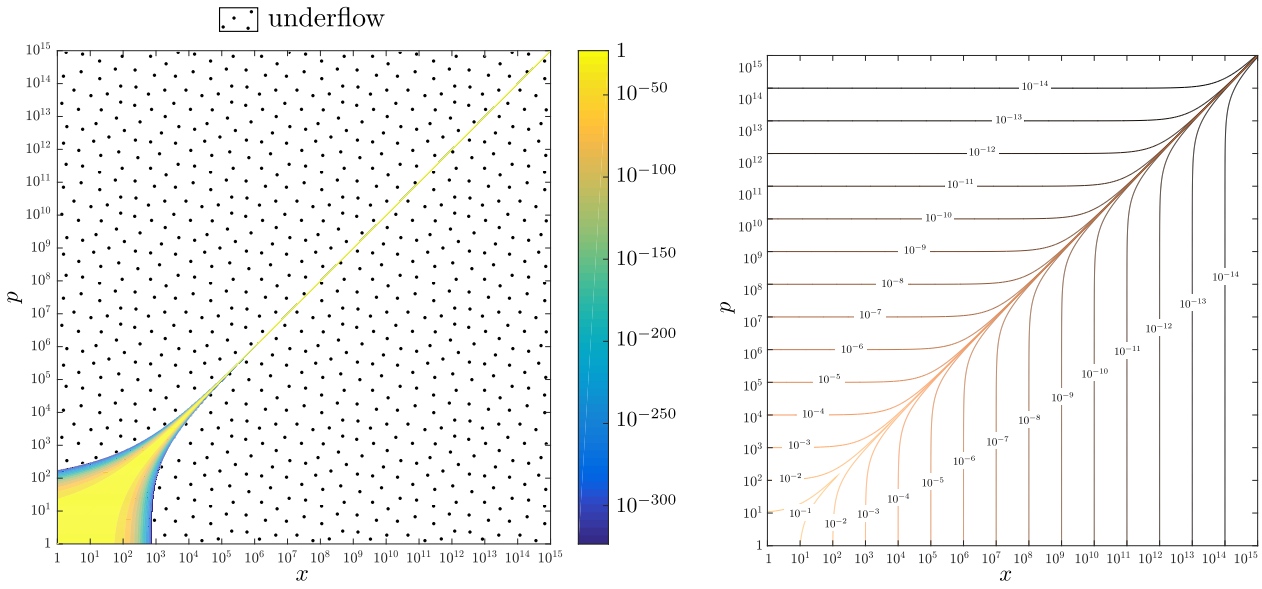
In the present paper, we shall use a different normalization, which considerably extends the range of admissible values of p and x. More precisely, we consider the single function

$$\forall x \geq 0, p > 0, \quad G(p,x) = e^{x - p\log x} \times \begin{cases} \gamma(p,x) & \text{if } x \leq p; \\ \Gamma(p,x) & \text{otherwise.} \end{cases} \tag{5}$$

From $G(p,x)$ and $\Gamma(p)$ it is straightforward to compute $\gamma(p,x)$ and $\Gamma(p,x)$ using (3). Moreover, we can see in Fig. 1 (b) that G is not subject to underflow or overflow issues on a very large domain, much larger than the domain $(0, 10^{15}]^2$ here considered. In the following, it will be useful to extend the definition (5) of G to the case $x < 0, p \in \mathbb{N}^*$ with

$$G(p,x) = e^{x - p\log|x|} \int_0^{|x|} s^{p-1}\, e^s\, ds = e^{x - p\log|x|} (-1)^p \gamma(p,x), \tag{6}$$

where $\gamma(p,x)$ is naturally extended to negative values of x with Equation (2). The possibility of evaluating the lower incomplete gamma function for negative values of x is explored by Thompson [2013], but in a situation different to ours, since he focused on the case $p = n + \frac{1}{2}$, $n \in \mathbb{Z}$. Gil et al. [2016] consider the more general case $x < 0$ and $p \in \mathbb{R}$, but since they directly estimate $\gamma(p,x)$ (without normalization), under- and overflow issues dramatically restrict the range of admissible values of x and p.



(a) values of $\min(P, Q)$ and the large "underflow region"

(b) isovalues of G

Fig. 1. Comparison between two normalizations of the incomplete gamma function. (a): values of $\min(P(p,x), Q(p,x))$ (see Equation (4)) as a function of x and p, with values below the smallest positive double precision floating point number ($\simeq 10^{-308}$) marked as underflow. (b): isovalues curves of the function $G(p,x)$ defined in Equation (5), represented on the same domain $(0, 10^{15}]^2$ (log scale). While $\min(P, Q)$ suffers from severe underflow issues as soon as $p \geq 10^3$ or $x \geq 10^3$, G slowly varies in the whole domain, without any risk of under- or overflow. From a numerical point of view, G is thus a much better normalization of the incomplete gamma function.

The aim of this paper is to propose a numerical algorithm to efficiently and accurately compute the generalized incomplete gamma function

$$I_{x,y}^{\mu,p} = \int_x^y s^{p-1}\, e^{-\mu s}\, ds \tag{7}$$

for $0 \leq x < y \leq +\infty$, $\mu = \pm 1$ and $p > 0$. When $\mu = -1$, we also impose the restriction that $y \neq +\infty$ and p is an integer in order to ensure that the integral is real and finite. Notice that for more general values of $\mu \in \mathbb{R}^*$, we have

$$\int_x^y s^{p-1}\, e^{-\mu s}\, ds = |\mu|^{-p}\, I_{|\mu|x,|\mu|y}^{\varepsilon,p}, \quad \text{with} \quad \varepsilon = \frac{\mu}{|\mu|}\,, \tag{8}$$

hence the hypothesis $\mu = \pm 1$ does not cause any loss of generality.

The numerical evaluation of the integral $I_{x,y}^{\mu,p}$ has found applications in the field of astronomy, for instance, in Hills [1975], where its computation was needed to model the dynamical evolution of stellar clusters. It is also needed as a renormalization factor as soon as the truncated version gamma (or Erlang) distribution is considered (see, for example, Verbelen et al. [2015], Philippe [1997] and the references therein). It was also recently needed in the field of image processing in Abergel et al. [2015], where the accurate computation of $I_{x,y}^{\mu,p}$ for a large range of parameters was at the heart of a denoising algorithm for the restoration of images corrupted with Poisson noise.

The generalized incomplete gamma function (7) was actually previously introduced in Fullerton [1972] under the slightly different form

$$J_{x_1,x_2}^{p} = e^{x_1} \int_{x_1}^{x_2} |s|^{p-1}\, e^{-s}\, ds\,, \quad \text{for } (x_1, x_2) \in \mathbb{R}^2 \text{ and } p > 0\,. \tag{9}$$

The integrals I and J are closely related since

$$\forall x, y,\ 0 < x < y, \quad \forall p > 0, \quad I_{x,y}^{\mu,p} = \begin{cases} e^{-x} J_{x,y}^{p} & \text{if } \mu = 1\,, \\ e^{y} J_{-y,-x}^{p} & \text{if } \mu = -1\,. \end{cases} \tag{10}$$

Unfortunately, Fullerton's algorithm, which was not validated for a large range of parameters, presents several weaknesses. As pointed out in Schoene [1978], for some values of the parameters, the algorithm suffers from numerical instabilities, yielding for instance, a computed integral with incorrect sign, or zero digits of precision.

We also observed some overflow issues when we tested the algorithm on a higher range of parameters (typically when $p \approx 10^2$ or higher, but also for many other parameter settings).

Note also that a function dedicated to the evaluation of $I_{x,y}^{1,p}$, Gamma[p,x,y], is available in the scientific computing software Mathematica$^{\text{TM}}$ (see Wolfram Research Inc [1988], and Wolfram Research Inc [1998] for the online evaluation of $I_{x,y}^{1,p}$). Unfortunately, Mathematica's algorithms are not currently disclosed to the public.

It is possible to estimate $I_{x,y}^{\mu,p}$ from the classical incomplete gamma functions, since for $\mu = 1$ one has

$$I_{x,y}^{\mu,p} = \gamma(p,y) - \gamma(p,x) = \Gamma(p,x) - \Gamma(p,y) = \Gamma(p) - \gamma(p,x) - \Gamma(p,y), \tag{11}$$

while for $\mu = -1$,

$$I_{x,y}^{\mu,p} = (-1)^p \Big(\gamma(p,-y) - \gamma(p,-x) \Big). \tag{12}$$

However, the effective computation of $I_{x,y}^{\mu,p}$ using (11) or (12) raises several numerical issues:

1. For some values of the parameters, the lower and upper incomplete gamma functions cannot be well approximated in the computer floating point arithmetic because they take values outside admissible bounds (about $10^{\pm 308}$ for IEEE 754). This is why it is important to choose an appropriate normalization, as we discussed earlier. In the present work, we will use the function G defined in Equation (5). In practice, because $I_{x,y}^{\mu,p}$ itself may not be directly representable, we will represent it under a mantissa-exponent form $\rho \cdot e^{\sigma}$, where ρ and σ are floating point numbers with double precision;

2. Computing $G(p,x)$ efficiently depends on the values of p and x. We shall present a simple division of the plane (p,x) into 3 regions and, in each region, a numerical procedure based on a continued fraction or on recursive integration by part;

3. When $I_{x,y}^{\mu,p}$ is computed as the difference $A - B$, the result may be inaccurate if A and B are close to each other (the well-known *cancellation* effect in floating-point arithmetic), which happens in (11)-(12) when x and y are very close to each other. In that case, the integral $I_{x,y}^{\mu,p}$ will be computed using Romberg's recursive numerical integration algorithm.

Note that the issue (1) detailed above is of great importance when some integrals of the kind $I_{x,y}^{\mu,p}$ appear in more complicated mathematical expressions, such as in Abergel et al. [2015], where the computation of a ratio of sums of generalized incomplete gamma functions is involved, with a numerator and a denominator that may both exceed the highest representable double floating point number, although the ratio itself is representable in the standard computer floating-point arithmetic.

This paper is organized as follows. In Section 2, we describe a new algorithm to evaluate the function $G(p,x)$. It is based on a partition of the parameter plane, which drives three different numerical procedures using continued fractions or recursive integrations by parts. We systematically evaluate the precision of this algorithm on a large domain, and show that it generally outperforms Gil et al. [2012] algorithm by two orders of magnitude, independently of the normalization issues encountered in the latter. We then consider in Section 3 the computation of the (un-normalized) incomplete gamma functions, and derive a theoretical accuracy bound from the limited precision of double precision floating point numbers associated with the required mantissa-exponent representation of these integrals. We then show that our estimates essentially achieve this optimal bound. We also compare several algorithms used for the evaluation of the (complete) Gamma function, which may be required by our algorithm. In Section 4, the more general case of the integral $I_{x,y}^{\mu,p}$ is considered, and we describe an algorithm based on differences or on Romberg's numerical integration method (and an automatic selection of the most appropriate method). This algorithm is compared on several examples with Fullerton's algorithm in Section 5, and its robustness is tested through massive usage in a recent image denoising application in Section 6.

2 Numerical computation of the function G

2.1 Series expansions

If p denotes a positive integer, writing the Taylor series expansion of the exponential function, with order $p-1$ and integral remainder, we get

$$e^x = \sum_{k=0}^{p-1} \frac{x^k}{k!} + \int_0^x \frac{(x-t)^{p-1}}{(p-1)!}\, e^t\, dt \underset{s=x-t}{=} e^x - \sum_{k=p}^{+\infty} \frac{x^k}{k!} + \frac{e^x}{(p-1)!} \int_0^x s^{p-1}\, e^{-s}\, ds\,.$$

For $x \leq p$, we deduce that

$$G(p,x) = \sum_{k=0}^{+\infty} \frac{\Gamma(p) \cdot x^k}{\Gamma(k+p+1)}\,, \tag{13}$$

and this formula remains valid for non-integer values of $p > 0$. Although the power series (13) defined above has an infinite radius of convergence, its convergence can be quite slow and numerically unstable depending on the values of p and x. It is suggested in Press et al. [1992] that the lower ratio P should be evaluated using the series expansion as long as $\frac{|x|}{p+1} < 1$; however, according to our experiments, a better convergence rate can be obtained for the function G by using a continued fraction. Thus, we shall not use (13) in the algorithm we propose.

2.2 Continued fractions

We describe here several formulations of our function G based on continued fractions taken from the literature. First, we focus on the computation of G in the domain $x \leq p$. Let us consider the confluent hypergeometric function M, defined by

$$M(a,b,z) = \sum_{n=0}^{+\infty} \frac{a^{(n)}}{b^{(n)}} \frac{z^n}{n!}\,, \quad \text{where} \quad \alpha^{(0)} = 1 \quad \text{and} \quad \alpha^{(n)} = \alpha(\alpha+1)\cdots(\alpha+n-1) \quad \text{for } n \geq 1.$$

Since for any (b,z) we have $M(0,b,z) = 1$, when $x \leq p$, we can rewrite (13) as

$$G(p,x) = \frac{M(1,p+1,x)}{p \cdot M(0,p,x)}\,. \tag{14}$$

As detailed in Olver et al. [2010], DLMF, Cuyt et al. [2008], Jones and Thron [1980], the ratio $\frac{M(a,b,z)}{M(a+1,b+1,z)}$ can be continued for any $z \in \mathbb{C}$, as long as $a \notin \mathbb{Z} \setminus \mathbb{N}$ and $a - b \notin \mathbb{N}$. Under this assumption (which will be satisfied here, since we will consider the setting $a = 0$, $b = p > 0$), and using the usual notation for continued fractions,

$$\frac{\alpha_1}{\beta_1+}\,\frac{\alpha_2}{\beta_2+}\,\frac{\alpha_3}{\beta_3+}\,\cdots = \frac{\alpha_1}{\beta_1 + \frac{\alpha_2}{\beta_2 + \frac{\alpha_3}{\beta_3 + \cdots}}}\,,$$

we get

$$\frac{M(a,b,z)}{M(a+1,b+1,z)} = 1 + \frac{u_1}{1+}\,\frac{u_2}{1+}\,\frac{u_3}{1+}\,\cdots\,,$$

with

$$\forall n \geq 0, \quad u_{2n+1} = \frac{(a-b-n)z}{(b+2n)(b+2n+1)} \quad \text{and} \quad u_{2n} = \frac{(a+n)z}{(b+2n-1)(b+2n)}.$$

Writing the inverse ratio (with $a = 0$ and $b = p$), and after basic manipulations of the continued fraction, we obtain

$$\frac{M(1,p+1,x)}{p \cdot M(0,p,x)} = \frac{a_1}{b_1+}\,\frac{a_2}{b_2+}\,\frac{a_3}{b_3+}\,\cdots\,,$$

where $a_1 = 1$ and $\forall n \geq 1$, $a_{2n} = -(p-1+n) \cdot x$, $a_{2n+1} = n \cdot x$ and $b_n = p - 1 + n$. Therefore, Equation (14) becomes

$$G(p,x) = \frac{a_1}{b_1+}\,\frac{a_2}{b_2+}\,\frac{a_3}{b_3+}\,\cdots \tag{15}$$

The evaluation of $G(p,x)$ in the domain $x \leq p$ using the continued fraction (15) can be performed using the modified Lentz's method Lentz [1976], Thompson and Barnett [1986], which we recall in Algorithm 1 for the reader's convenience, with a slight adaptation of the initialization process since we observed some instabilities when using the implementation described in Press et al. [1992] (see note in Algorithm 1).

The continued fraction (15) converges for any value of x, but leads to numerical instabilities (due to the fact that its value becomes huge) when x is chosen too large compared to p. However, since we restrict its use to $x \leq p$, this instability does not arise in our algorithm. The convergence of (15) in the domain $x \leq p$ is fast as it requires in general less than 20 approximants to converge (the number of iterations being estimated automatically). The number of required approximants may be higher when $x \approx p$, or when $x < 0$ and p is small. For the latter case, we will switch to a faster estimation method based on recursive integration by parts (Section 2.3).

Algorithm 1: Modified[1] Lentz's method for continued fractions evaluation.

Input: Two real-valued sequences $\{a_n\}_{n\geq 1}$ and $\{b_n\}_{n\geq 1}$, with $b_1 \neq 0$.

Requirements: $\varepsilon_{\text{machine}}$ (machine precision, $2.22 \cdot 10^{-16}$ in double precision)

Output: Accurate estimate f of the continued fraction $\frac{a_1}{b_1+}\ \frac{a_2}{b_2+}\ \frac{a_3}{b_3+}\ \cdots$

Initialization:
$d_m \leftarrow 10^{-300}$ `// Number near the minimal floating-point value`

$f \leftarrow \frac{a_1}{b_1}$; $C \leftarrow \frac{a_1}{d_m}$; $D \leftarrow \frac{1}{b_1}$; $n \leftarrow 2$ `// see note[1] below`

repeat
- $D \leftarrow D \cdot a_n + b_n$
- **if** $D = 0$ **then** $D \leftarrow d_m$
- $C \leftarrow b_n + \frac{a_n}{C}$
- **if** $C = 0$ **then** $C \leftarrow d_m$
- $D \leftarrow \frac{1}{D}$
- $\Delta \leftarrow C \cdot D$
- $f \leftarrow f \cdot \Delta$
- $n \leftarrow n + 1$

until $|\Delta - 1| < \varepsilon_{machine}$

return f

[1] In the initialization step, we manually performed the first pass $n = 1$ of the modified Lentz's algorithm, since we observed some instabilities with the initialization $f = C = d_m$, $D = 0$, presented in Press et al. [1992]. Indeed, the setting $C = d_m$ may yield $C = +\infty$ after the pass $n = 1$ (when a_1/d_m exceed the highest representable number), and then $\Delta = f = +\infty$, which propagates through the next iterations. By computing the first pass manually, even when the initialization $C = a_1/d_m$ yields $C = +\infty$, the pass $n = 2$ yields $C = b_2 + a_2/C = b_2$, which has a finite value.

In the domain $x > p$, the evaluation of $G(p, x)$ can be performed using another continued fraction. Indeed, as detailed in Abramowitz and Stegun [1964], Press et al. [1992], for $x > p$, we can write $G(p, x)$ as

$$G(p,x) = \frac{\alpha_1}{\beta_1+}\ \frac{\alpha_2}{\beta_2+}\ \frac{\alpha_3}{\beta_3+}\ \cdots, \tag{16}$$

with $\alpha_1 = 1$, $\alpha_n = -(n-1)\cdot(n-p-1)$ for any $n > 1$, and $\beta_n = x + 2n - 1 - p$ for any $n \geq 1$. The continued fraction (16) can also be evaluated numerically using Algorithm 1.

2.3 Recursive integrations by parts

In the domain $x < 0$, we observed that the use of (15) to compute $G(p, x)$ may lead to long computation times for small values of p. Since in the case $x < 0$ we restricted the study of $G(p, x)$ to integer values of p, we can consider, as an alternative to (15), the use of a recursive integration by parts closed-form formula to compute $G(p, x)$. We remark that a similar approach is adopted by Guseinov and Mamedov [2004] who make use of backward and forward recurrence relations to evaluate the standard lower incomplete gamma function, $\gamma(p, x)$, for positive values of x. Considering the case $x < 0$ and p integer, we obtain

$$G(p,x) = \frac{1}{x}\left(\frac{(p-1)!\,e^x}{x^{p-1}} - \sum_{k=0}^{p-1}\frac{(p-1)!}{(p-1-k)!}\,x^{-k}\right). \tag{17}$$

Although the computation of (17) is not efficient in general, it happens to be faster than (15) for small values of p. We must however be careful when computing the alternating sum (17) since, as usual with alternating sums, it may suffer from dramatic errors caused by cancellation. In the following, we set $t = -x > 0$ and we rewrite (17) as

$$G(p,x) \underset{t=-x}{=} \frac{1}{t}\left(\frac{(-1)^p(p-1)!\,e^{-t}}{t^{p-1}} + s(t)\right), \quad \text{where} \quad s(t) = \sum_{k=0}^{p-1}(-1)^k\frac{(p-1)!\,t^{-k}}{(p-1-k)!}. \tag{18}$$

By grouping the consecutive terms in pairs with indexes $k = 2l$ and $k = 2l+1$ of the alternating sum $s(t)$, we get

$$s(t) = \tilde{s}(t) := \sum_{l=0}^{\lfloor\frac{p-2}{2}\rfloor}\frac{(p-1)!\,t^{-(2l+1)}}{(p-1-2l)!}\,(t-(p-1-2l)) + \varepsilon_p(t), \tag{19}$$

where $\lfloor z \rfloor$ denotes the integer part of z, and the residual term $\varepsilon_p(t)$ is defined by

$$\varepsilon_p(t) = \begin{cases} (p-1)!\, t^{-(p-1)} & \text{if } p \text{ is odd} \\ 0 & \text{otherwise.} \end{cases}$$

Let us now assume that $t \geq \max(1, p-1)$. First, using $t \geq p-1$, we see that all terms in the sum $\tilde{s}(t)$ are nonnegative, so that we can evaluate the alternating sum (19) without any cancellation. It follows that, when p is even, (18) becomes

$$G(p,x) = \frac{1}{t}\left(\frac{(p-1)!e^{-t}}{t^{p-1}} + \tilde{s}(t)\right),$$

which is a sum of positive terms, so that no cancellation can occur. When p is odd, (18) yields

$$G(p,x) = \frac{1}{t}\left(-\frac{(p-1)!e^{-t}}{t^{p-1}} + \tilde{s}(t)\right). \tag{20}$$

Noting $\alpha(t) = \frac{(p-1)!e^{-t}}{t^{p-1}}$ and using the fact that $t \geq 1$, we get

$$\frac{\tilde{s}(t)}{\alpha(t)} \geq \frac{\varepsilon_p(t)}{\alpha(t)} = \exp(t) \geq \exp(1),$$

which ensures that no cancellation occurs when computing the difference between $\tilde{s}(t)$ and $\alpha(t)$, involved in (20). Finally, we are able to evaluate (18) without cancellation in the region $t \geq \max(1, p-1)$.

Last, from $t > p-1$, we infer that the sequence $\{a_k(t)\}_{k\geq 0}$ defined by

$$\forall k \geq 0, \quad a_k(t) = \begin{cases} \frac{(p-1)!\, t^{-k}}{(p-1-k)!} & \text{if } k \leq p-1 \\ 0 & \text{otherwise,} \end{cases}$$

is nonincreasing, with limit 0. It follows that the remainder $r_n(t) = \sum_{k=n+1}^{+\infty}(-1)^k a_k(t)$ of the alternating series $s(t) = \sum_{k=0}^{+\infty}(-1)^k a_k(t)$ satisfies $|r_n(t)| \leq a_{n+1}(t)$, so that we can numerically estimate $s(t)$ with the partial sum $s_n(t) = \sum_{k=0}^{n}(-1)^k a_k(t)$ as soon as

$$a_{n+1}(t) \leq |s_n(t)| \cdot \varepsilon_{\text{machine}}, \tag{21}$$

$\varepsilon_{\text{machine}}$ being the machine precision, $2.22 \cdot 10^{-16}$ in double precision floating point arithmetic (IEEE 754 standard). Note that (21) may occur for $n < p-1$, with a possible saving in computation time. In practice, we compute $s(t) = \tilde{s}(t)$ with (19) instead of (18), but this stopping criterion can be easily evaluated at each iteration of the summation procedure. Indeed, noting that the sequence $\{a_{2l}(t) - a_{2l+1}(t)\}_{l\geq 0}$ is positive and nonincreasing (because $t > p-1$), we obtain

$$\forall l \in \mathbb{N}, \quad a_{2l+2}(t) \leq a_{2l}(t) - a_{2l+1}(t) + a_{2l+3}(t) \leq a_{2l}(t) - a_{2l+1}(t),$$

so that

$$\forall l \in \mathbb{N}, \quad |r_{2l+1}(t)| \leq a_{2l+2}(t) \leq |a_{2l}(t) - a_{2l+1}(t)|.$$

This yields Algorithm 2.

2.4 Numerical computation of the function G

We estimated $G(p,x)$ using (15) in the domain $x \leq p$, using (16) in the domain $x > p$, and using (17) in the domain $\{x < 0; |x| > \max(1, p-1)\}$, for a large range of parameters, namely

$$x \in [-1000, 1000] \cap \mathbb{Z}, \quad p \in [1, 1000] \cap \mathbb{Z}.$$

For each tested value of (x,p) and each evaluation method, we compared the computed value of $G(p,x)$ to the one computed with Maple™ (version 17), with 30 significant decimal digits (which requires large amounts of memory and a long computation time), using the code

```
sigma:=evalf(x-p*log(abs(x)));
evalf(Int(s^(p-1)*exp(-sign(x)*s),s=0..abs(x),digits=30)*exp(sigma));   the last line only be-
evalf(Int(s^(p-1)*exp(-s),s=x..infinity,digits=30)*exp(sigma));
```

ing computed for positive values of x. The values computed with Maple™ were then used as reference values

Algorithm 2: Accurate evaluation of $G(p,x)$ using (17).

Input: A negative real number $x < 0$ and a positive integer p satisfying $|x| > \max(1, p-1)$.

Output: An accurate estimate of $G(p,x)$ computed using (17).

Initialization: $t \leftarrow -x$; $c \leftarrow \dfrac{1}{t}$; $d \leftarrow p-1$; $s \leftarrow c \cdot (t-d)$; $l \leftarrow 1$

repeat

$c \leftarrow \dfrac{d(d-1)}{t^2}$

$d \leftarrow d-2$

$\Delta \leftarrow c(t-d)$ `// now` $\Delta = a_{2l}(t) - a_{2l+1}(t)$

$s \leftarrow s + \Delta$ `// now` $s = s_{2l+1}(t) = \displaystyle\sum_{k=0}^{2l+1} (-1)^k a_k(t)$

$l \leftarrow l+1$

until $l > \lfloor \dfrac{p-2}{2} \rfloor$ *or* $\Delta < s \cdot \varepsilon_{machine}$

if *($\Delta \geq s \cdot \varepsilon_{machine}$) and (p is odd)* **then** $s \leftarrow s + \dfrac{d\,c}{t}$ `// add the term` $\varepsilon_p(t) = (p-1)!\,t^{-(p-1)}$

return $\dfrac{1}{t}\left((-1)^p \cdot e^{-t+\log(p-1)!-(p-1)\log(t)} + s\right)$

to evaluate the relative accuracy of our estimate. After measuring the computation time and relative error for the two concurrent methods in the domain $\{x < 0\}$ (continued fraction versus recursive integration by parts), we designed a nearly optimal boundary to divide the domain $\{x < 0\}$ into two regions that select the most appropriate method. This results in a partition of the whole admissible domain into three regions (see Fig. 2) delimited by the explicit boundary

$$\forall x \in \mathbb{R} \cup \{+\infty\}, \quad p_{\lim}(x) = \begin{cases} 5\sqrt{|x|} - 5 & \text{if } x < -9\,, \\ 0 & \text{if } -9 \leq x \leq 0\,, \\ x & \text{otherwise,} \end{cases} \tag{22}$$

from which Algorithm 3 follows.

Algorithm 3: Fast and accurate evaluation of $G(p,x)$.

Input[1]**:** a number $x \in \mathbb{R} \cup \{+\infty\}$ and a positive real number p.

if $p \geq p_{lim}(x)$ **then** compute $G(p,x)$ using (15) and Algorithm 1
else if $x \geq 0$ **then** compute $G(p,x)$ using (16) and Algorithm 1
else compute $G(p,x)$ using (17) and Algorithm 2

[1] Recall that in the case $x < 0$, p must be an integer.

2.5 Numerical validation and comparison to Gil et al. [2012]

The purpose of our paper is to provide an efficient numerical algorithm to estimate quantities derived from incomplete gamma integrals (be it the integrals $\gamma(p,x)$ and $\Gamma(p,x)$ themselves, or their logarithms, or the ratios $G(p,x)$, $P(p,x)$ and $Q(p,x)$, or their logarithms, etc.). In this section, we focus on the relevance of considering $G(p,x)$ as an alternative function to the more standard ratios $P(p,x)$ and $Q(p,x)$ for that particular purpose. In our comparisons, we will compute $G(p,x)$ using Algorithm 3 and $\min(P(p,x), Q(p,x))$ using the recent state-of-the-art algorithm proposed by Gil et al. [2012] (the higher ratio is deduced from the smaller using the relation $P + Q = 1$). Beyond any numerical experiment, Figure 1 (a) alone already attests that, for most values of (p,x), the incomplete gamma integrals $\gamma(p,x)$ and $\Gamma(p,x)$ cannot be reliably computed from estimates in double precision of $P(p,x)$ or $Q(p,x)$, because in most situations, the quantity $\min(P(p,x), Q(p,x))$ will underflow (that is, smaller than the smallest double precision positive number which is about 10^{-308}), so that one will compute $\min(P(p,x), Q(p,x)) = 0$ and thus obtain a result with 0% relative accuracy. This remark does not question the accuracy of the computation of $P(p,x)$ and $Q(p,x)$ using Gil et al. [2012] algorithm (or even any concurrent algorithm), but simply highlights the inadequacy of the ratios $P(p,x)$ and $Q(p,x)$ for the purpose of computing accurate estimates of quantities derived from the incomplete gamma integrals. Besides,

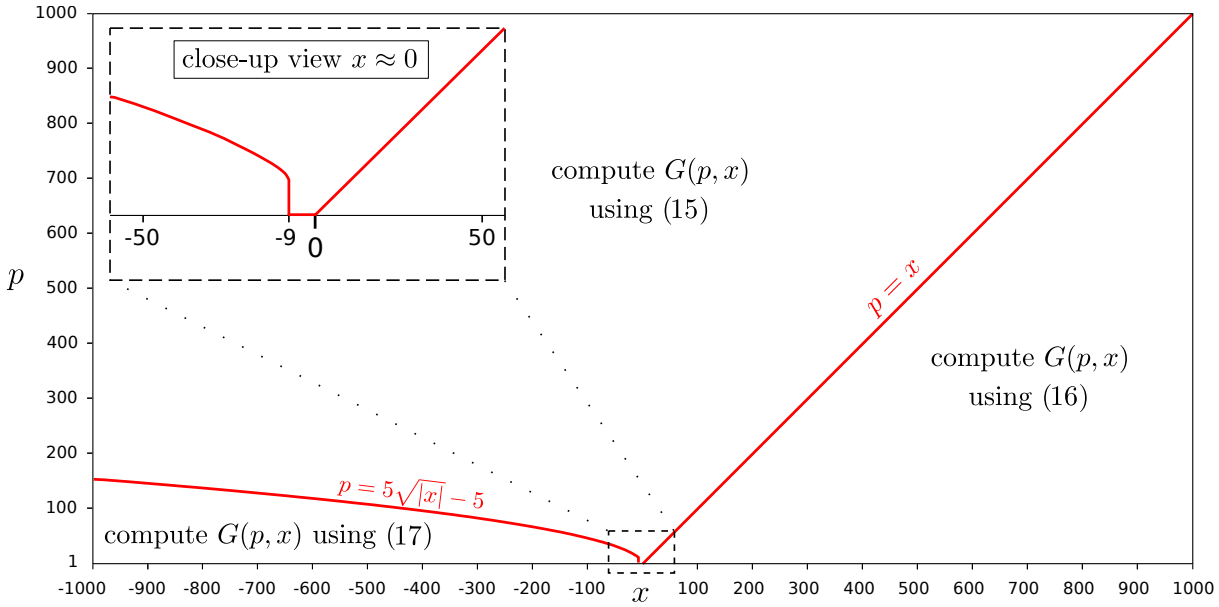


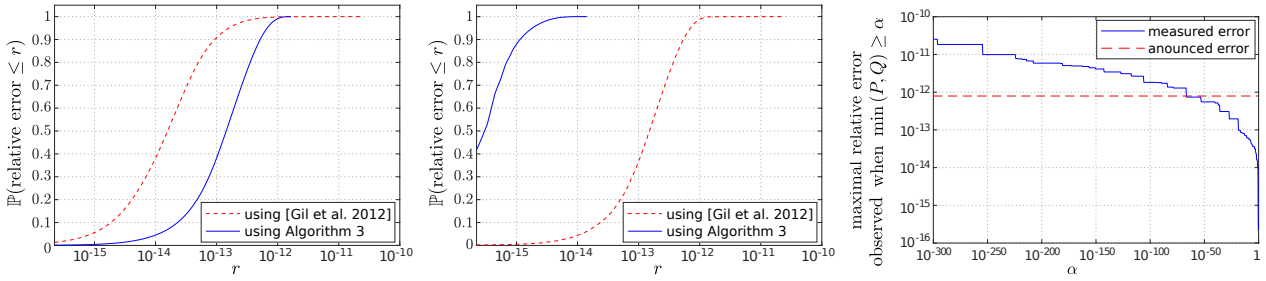
Fig. 2. Partition of the (x,p) domain for the evaluation of $G(p,x)$. The normalized incomplete gamma function G defined in Equation (5) is numerically evaluated using Algorithm 3, which selects the appropriate formula among (15), (16), and (17) according to the location of (p,x) in the partition of the domain delimited by the red curve.

since $G(p,x)$, $P(p,x)$ and $Q(p,x)$ are not the same quantities, it is difficult to compare the algorithms dedicated to their evaluation. Of course, one can use $G(p,x)$ to compute $P(p,x)$ and $Q(p,x)$, or use $P(p,x)$ and $Q(p,x)$ to compute $G(p,x)$ using

$$\forall x \geq 0\,, \forall p > 0\,, \quad G(p,x) \cdot e^{-x+p\log x - \log\Gamma(p)} = \begin{cases} P(p,x) & \text{if } x \leq p \\ Q(p,x) & \text{otherwise}\,. \end{cases} \tag{23}$$

In that process, the accuracy of the desired quantity may strongly depend on the accuracy of the computed rescaling factor $e^{-x+p\log x-\log\Gamma(p)}$. In Fig. 3, we can see that it is indeed the case, since the rescaled quantities ($G(p,x)$ computed from Gil et al. [2012] algorithm, and $\min(P(p,x),Q(p,x))$ computed from Algorithm 3) exhibit the same distribution of relative errors, better than $5 \cdot 10^{-13}$ for 90% of the tested parameters. In fact, this corresponds to the precision obtained for the rescaling factor $e^{-x+p\log x-\log\Gamma(p)}$, which is the limiting factor in the process. In Fig. 3 (b) (blue curve), we can observe that the precision obtained for G using Algorithm 3 is better than 10^{-15} for 90% of the tested parameters. This is substantially better than the precision obtained by Gil et al. [2012] in their evaluation of $\min(P,Q)$, which is close to 10^{-13} for 90% of the tested parameters (see Fig. 3 (a), red curve). This means that improving the estimation of the rescaling factor up to 10^{-15} would result in a better estimation of $\min(P,Q)$ using the rescaled value of G than using the algorithm of Gil et al. [2012]. For that reason, we believe that it is interesting to compare the relative precision of the normalized quantities, keeping aside the accurate computation of the rescaling factor which could be the topic of another study. Moreover, in the scope of this paper, which is to compute quantities derived from the incomplete gamma integrals accurately, the comparison of the normalized quantities is a fair choice (for instance, we could have compared the estimates of $\gamma(p,x)$, $\log\gamma(p,x)$, or $\log P(p,x)$, but no particular choice would impose itself).

In this paper, all evaluations of Gil et al. [2012] algorithm were made using the Fortran implementation publicly available at the address `http://personales.unican.es/gila/incgam.zip`, which is the link given in Gil et al. [2012]. With this implementation, we noticed some failure cases for parameter values that should produce a correct value of $\max(P,Q)$ and raise an error (underflow) flag for $\min(P,Q)$. An example is given for $(p,x) = (4000, 7000)$, for which the code returns $P = 0, Q = 1$ and no error flag (instead of $P = 1, Q = 0$ and an error flag indicating underflow for Q). This failure case is relatively frequent for large values of p and x. We conclude this section with Table 1, which shows that Algorithm 3 and Gil et al. [2012] algorithm exhibit comparable statistics in terms of computation time.



(a) evaluation of $\min(P, Q)$ (b) evaluation of G (c) relative errors observed using the algorithm of Gil et al. [2012]

Fig. 3. Relative accuracy of Algorithm 3 and Gil et al. [2012] algorithm. Left (a): over the domain $(p, x) \in \{1, 2, \ldots, 1000\}^2 \cap \mathcal{S}$ (recall that $\mathcal{S} = \{(p, x),\ \min(P(p, x), Q(p, x)) \geq 10^{-300}\}$), we systematically computed (using Maple$^{\text{TM}}$ as a reference) the relative error observed for $\min(P(p, x), Q(p, x))$ when using Gil et al. [2012] algorithm and when using $G(p, x)$ (computed with Algorithm 3) with the rescaling (23). We display the proportion of relative errors smaller than r as a function of r in both cases. **Middle (b):** Symmetrically, over the same domain, we computed the relative error observed when evaluating $G(p, x)$ using Algorithm 3 or using $\min(P(p, x), Q(p, x))$ (computed using the algorithm of Gil et al. [2012]) with the rescaling (23). Again, we display the proportion of relative errors smaller than r as a function of r in both cases. **Right (c)**: we evaluated the relative error of Gil et al. [2012] over the domain $(x, p) \in \{1, 2, \ldots, 500\}^2 \cap \mathcal{S}$. The maximum over the domain $\{(x, p); \min(P(x, p), Q(x, p)) \geq \alpha\}$ is displayed as a function of α. We can see that the accuracy of $7.9 \cdot 10^{-13}$ (red dashed line) claimed by Gil et al. [2012] after 10^7 random trials on that domain is not systematically attained, and that many larger errors appear for estimated values of $\min(P, Q)$ that are far from underflow (10^{-308}), for example $P(1, 51)$ =2.41819039187902807E-67 ($1.3 \cdot 10^{-12}$ relative error) or $P(1, 99)$ =3.98167886822098022E-157 ($4.8 \cdot 10^{-12}$ relative error).

Execution times (ns)	minimum	median	average	maximum
G (using Algorithm 3)	288 ns	529 ns	442 ns	2038 ns
P and Q (using Gil et al. [2012])	378 ns	1010 ns	1450 ns	3456 ns

Table 1: Comparison of the execution time between the algorithm of Gil et al. [2012] and Algorithm 3. Both algorithms were evaluated in the range $(x, p) \in \{0, \ldots, 1000\} \times \{1, \ldots, 1000\}$. For each tested value of (x, p), the value of $G(p, x)$ and that of $(P(p, x), Q(p, x))$ were computed as many times as possible during 10 ms. Dividing 10 ms by the number of achieved evaluations, we get an accurate estimate of the computation time for a single evaluation. In this Table, for each algorithm, we report the minimal, median, average and maximum execution time observed in the tested range of parameters.

3 Numerical computation of the non-normalized lower and upper incomplete gamma integrals

3.1 Mantissa-exponent representation of the incomplete gamma integrals

When the evaluation of the non-normalized lower and upper incomplete gamma integrals $\gamma(p, x)$ and $\Gamma(p, x)$ is needed, we can compute

$$\forall x \geq 0\,,\ \forall p > 0\,,\quad G(p, x) \times e^{-x+p\,\log x} = \begin{cases} \gamma(p, x) & \text{if } x \leq p \\ \Gamma(p, x) & \text{otherwise} \end{cases} \tag{24}$$

and, by subtracting $G(p, x) \times e^{-x+p\,\log x}$ from the complete gamma function $\Gamma(p)$, we can also retrieve the lower and upper incomplete gamma integrals on the complementary domains:

$$\forall x \geq 0\,,\ \forall p > 0\,,\quad \Gamma(p) - G(p, x) \times e^{-x+p\,\log x} = \begin{cases} \Gamma(p, x) & \text{if } x \leq p \\ \gamma(p, x) & \text{otherwise.} \end{cases} \tag{25}$$

Notice that no cancellation can occur in the subtraction (25) since $G(p, x) \times e^{-x+p\,\log x}$ is at most roughly equal to $0.5 \cdot \Gamma(p)$. Moreover, Equations (24) and (25) naturally provide a kind of mantissa-exponent representation of the type $I = \rho \cdot e^{\sigma}$ for the incomplete gamma functions, as

$$\begin{cases} \gamma(p, x) = \rho_1 \cdot e^{\sigma_1} \\ \Gamma(p, x) = \rho_2 \cdot e^{\sigma_2} \end{cases} \text{if } 0 \leq x \leq p, \text{ and } \begin{cases} \gamma(p, x) = \rho_2 \cdot e^{\sigma_2} \\ \Gamma(p, x) = \rho_1 \cdot e^{\sigma_1} \end{cases} \text{if } x > p, \tag{26}$$

$$\text{with } \sigma_1 = -x + p\,\log|x|,\ \ \rho_1 = G(p, x), \tag{27}$$

$$\text{and } \sigma_2 = \log\Gamma(p),\ \ \rho_2 = 1 - e^{-x+p\,\log x - \log\Gamma(p)} G(p, x). \tag{28}$$

Note that when $x < 0$, we also have a similar representation for the lower incomplete gamma function, that is, $\gamma(p, x) = (-1)^p \rho_1 \cdot e^{\sigma_1}$.

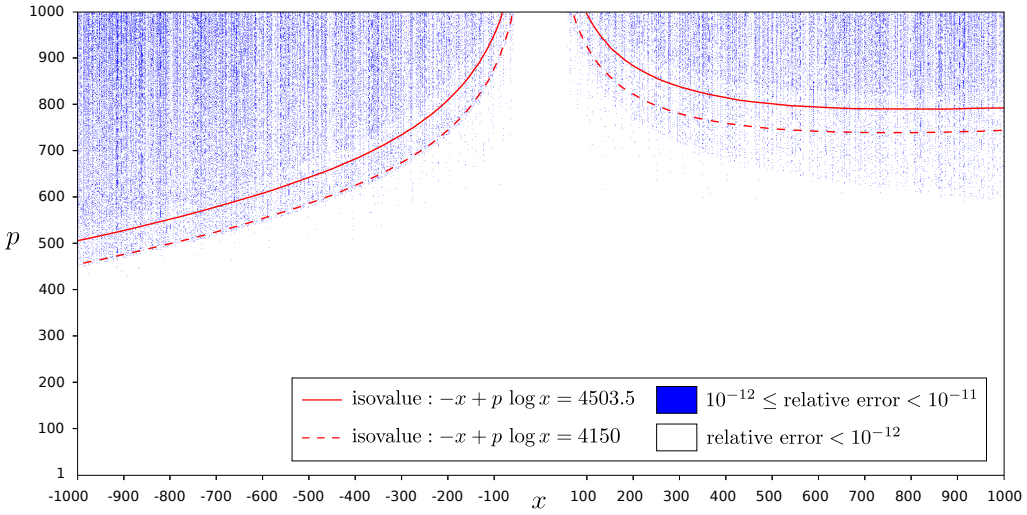


Fig. 4. Relative accuracy associated with the computation of the generalized lower and upper incomplete gamma functions. We estimated the relative accuracy of the incomplete gamma functions given by $e^{-x+p\log|x|}G(p,x)$ (see Equations (5) and (6)) on the domain $(x,p) \in \{-1000,\dots,1000\} \times \{1,\dots,1000\}$. The relative error is smaller than 10^{-11} everywhere. The blue points, corresponding to the parameters (x,p) for which the relative accuracy is larger than 10^{-12}, reveal perceptible boundaries whose form is predicted by Equation (31), from which the red curves are obtained. This shows that the main source of error is due to the mantissa-exponent representation and that our estimates can be considered as nearly optimal with respect to this representation. In terms of execution time for the computation of $G(x,p)$ over this tested range of parameters (x,p), we measured an average computation time of 0.49 microseconds, a median execution time of 0.45 microseconds, and 2.1 microseconds as highest execution time (using a 3.1GHz Intel™ i7-7920HQ processor).

Such a mantissa-exponent representation considerably extends the range over which the integrals $\gamma(p,x)$ and $\Gamma(p,x)$ can be represented (in comparison with a direct evaluation of those integrals in double precision), since both the mantissa ρ and the exponent σ are computed in double precision floating-point (thus, with range $[10^{-308}, 10^{308}]$) and we can explicitly format the quantity $\rho \cdot e^{\sigma}$ in scientific notation (that is $\rho \cdot e^{\sigma} = a \cdot 10^{b}$, where $a \in [1,10)$ and $b \in \mathbb{Z}$) using

$$a = 10^{c-\lfloor c \rfloor}, \quad b = \lfloor c \rfloor, \quad \text{where} \quad c = \frac{\sigma}{\log(10)} + \log_{10}(\rho). \tag{29}$$

Notice, however, that the relative accuracy of the mantissa-exponent representation strongly depends on the magnitude of the exponent, as

$$\left|\frac{\Delta(\rho \cdot e^{\sigma})}{\rho \cdot e^{\sigma}}\right| \approx \left|\frac{\Delta\rho}{\rho}\right| + \left|\frac{\Delta(e^{\sigma})}{e^{\sigma}}\right| = \left|\frac{\Delta\rho}{\rho}\right| + |\Delta\sigma| = \left|\frac{\Delta\rho}{\rho}\right| + |\sigma| \cdot \left|\frac{\Delta\sigma}{\sigma}\right| := E \tag{30}$$

where $|\Delta X|$ and $|\Delta X/X|$ respectively denote the absolute and relative errors between the actual value of X and its computed value. As $|\sigma|$ increases, since ρ and σ are at best estimated at machine precision (i.e., $|\Delta\rho/\rho| = |\Delta\sigma/\sigma| = \varepsilon_{\text{machine}}$), we have

$$E \approx (1 + |\sigma|) \cdot \varepsilon_{\text{machine}}. \tag{31}$$

For instance, when $\sigma \approx 4503.5$, the best relative accuracy that can be expected is $E \approx 4504.5 \times 2.22 \cdot 10^{-16} = 10^{-12}$ with double precision floating point arithmetic. Therefore, we can predict approximate bounds for the relative error E achievable when computing $\gamma(p,x)$ or $\Gamma(p,x)$ under the form $\rho \cdot e^{\sigma}$ using (26)-(28). We can see in Fig. 4 that formatting the estimated values of the incomplete gamma functions in scientific notation using (29) yields a relative error similar to that predicted in (31).

It is interesting to notice that this limitation on the achievable relative error is not fundamentally due to the mantissa-exponent representation, but to the fact that the derivatives of the incomplete gamma functions grow rapidly with p and x. When approximating a function $f(x)$, the minimum achievable relative error is

$$\left|\frac{\Delta f(x)}{f(x)}\right| = \left|\frac{\Delta f(x)}{\Delta x}\right| \cdot \left|\frac{\Delta x}{x}\right| \cdot \left|\frac{x}{f(x)}\right| \gtrapprox \left|\frac{x f'(x)}{f(x)}\right| \cdot \varepsilon_{\text{machine}}. \tag{32}$$

In the case of the lower incomplete Gamma function $f(x) = \gamma(p,x)$ (with p fixed and $x \leq p$), one easily checks that $xf'(x)/f(x) = 1/G(x,p)$, so that the relative error due to the limited representation of x is about $\varepsilon_{\text{machine}}/G(x,p)$. From Fig. 1 (b), we observe that $G(x,p) \gtrapprox 1/x$, which means that independently of the representation of the result, evaluating the lower incomplete gamma function for $x \geq 10^{16}$ does not make much sense when x is stored as a double precision floating point number (roughly speaking, the first digit of $\gamma(p,x)$ result depends on the last digit of x).

3.2 Discussion on the evaluation of the complete gamma function

The computation of the complete gamma function $\Gamma(p)$ for $p \in \mathbb{N}$, $\mathbb{R}$ or $\mathbb{C}$ constitutes in itself a wide area of research. The object of this section is to compare several methods from the literature and derive a practical efficient algorithm for computing the quantity $\log\Gamma(p)$, which is needed to compute ρ_2 and σ_2 in Equation (28). Note that the relative error on Γ is, when small enough, numerically equal to the absolute error on $\log\Gamma(p)$, since

$$|\Delta\log\Gamma(p)| = |\log(\Gamma(p)+\Delta\Gamma(p)) - \log\Gamma(p)| = \left|\log\left(1+\frac{\Delta\Gamma(p)}{\Gamma(p)}\right)\right| \approx \left|\frac{\Delta\Gamma(p)}{\Gamma(p)}\right|. \tag{33}$$

Consequently, since the absolute error on $\log\Gamma(p)$ is at best $\varepsilon_{\text{machine}}\cdot\log\Gamma(p)$, the minimum achievable relative error on $\Gamma(p)$ is also $\varepsilon_{\text{machine}}\cdot\log\Gamma(p)$ when $\Gamma(p)$ is represented by its logarithm.

When p is a positive integer, we have $\Gamma(p) = (p-1)!$ and thus

$$\log\Gamma(p) = \sum_{k=1}^{p-1}\log k\,.$$

However, the numerical computation of this sum becomes rapidly inaccurate when p is large, because of the accumulation of small numerical errors made at each step of the summation. Besides, we do not want to be limited to integer values of p.

The first evaluation method that we will consider was proposed in Lanczos [1964], and uses a Stirling formula-like approximation:

$$\forall p>0,\quad \Gamma(p) = \sqrt{2\pi}\left(p+\gamma-\tfrac{1}{2}\right)^{p-\frac{1}{2}} e^{-(p+\gamma-\frac{1}{2})}\left(A_\gamma(p-1)+\varepsilon_\gamma\right), \tag{34}$$

where $\gamma>0$ is a numerical parameter (different from the Euler-Mascheroni constant), $A_\gamma(p-1)$ is a truncated rational fraction that can be written

$$A_\gamma(p-1) = c_0(\gamma) + \sum_{k=1}^{N_\gamma}\frac{c_k(\gamma)}{p-1+k}, \tag{35}$$

and N_γ and the coefficients $\{c_k(\gamma)\}_{0\le k\le N_\gamma}$ depend on the value of γ. In the case $\gamma=5$, Lanczos claims that the relative error $|\varepsilon_5|$ associated with (34) satisfies $|\varepsilon_5|<2\cdot10^{-10}$, and this claim was confirmed by our numerical experiments. In the case $\gamma=5$, we have $N_\gamma=6$ and the numerical values of the coefficients $\{c_k(\gamma)\}_{0\le k\le N_\gamma}$ are available in Lanczos [1964]. These values are refined to double floating-point precision in Press et al. [1992], so we used them in our implementation of (34).

A more recent computation method (see Char [1980], Olver et al. [2010], Cuyt et al. [2008] and references therein), also based on a Stirling approximation, consists in computing

$$\forall p>0,\quad \Gamma(p) = \sqrt{2\pi}\,e^{-p}\,p^{p-\frac{1}{2}}\,e^{J(p)},\quad \text{where}\quad J(p) = \frac{a_0}{p+}\,\frac{a_1}{p+}\,\frac{a_2}{p+}\,\cdots, \tag{36}$$

where some numerical approximations, to 40 decimal digits of precision, of the coefficients $\{a_k\}_{0\le k\le 40}$ of the continued fraction $J(p)$ are given in Char [1980].

The last approximation we present is a refinement of the Lanczos formula (34), proposed by Pugh [2004]:

$$\forall p>0,\quad \Gamma(p) \approx 2\sqrt{\frac{e}{\pi}}\left(\frac{p+r-\frac{1}{2}}{e}\right)^{p-\frac{1}{2}}\left[d_0+\sum_{k=1}^{N_r}\frac{d_k}{p-1+k}\right], \tag{37}$$

where r is again a numerical parameter. Pugh studied the accuracy of the approximation (37) for different values of r. In the case $r=10.900511$, he sets $N_r=11$, and gives the numerical values of the coefficients $\{d_k\}_{0\le k\le 10}$ to 20 significant decimal digits (see Table 2). According to Pugh, this setting yields a relative error less than 10^{-19}, which is effectively what we observed when computing (37) with Maple$^{\text{TM}}$ for $p\in\{1,2,\dots10^3\}$ in multiprecision (30 digits).

In order to select which method will be used in our algorithms, we computed $\log\Gamma(p)$ for $1\le p\le10^4$, using the three approximations (34), (36), and (37). The accuracy was evaluated using Maple$^{\text{TM}}$, and the results (restricted to $1\le p\le5000$) are displayed in Fig. 5. It follows from this experiment that the best estimate is obtained with Pugh's method (37). The continued fraction (36) yields similar results for most values of p, but is quite inaccurate for very small values of p. Note also that, except for small values of p, these two methods are roughly optimal, since they deliver an absolute error approaching the theoretical limit $\varepsilon_{\text{machine}}\cdot\log\Gamma(p)$ mentioned earlier. In the end, we selected Pugh's method for the numerical evaluation of $\log\Gamma(p)$ in (28) because of its superior accuracy, its simplicity and the nice theoretical study provided in Pugh [2004]. Our implementation is described in Algorithm 4.

k	d_{3k}	d_{3k+1}	d_{3k+2}
0	2.48574089138753565546E-05	1.05142378581721974210E+00	-3.45687097222016235469E+00
1	4.51227709466894823700E+00	-2.98285225323576655721E+00	1.05639711577126713077E+00
2	-1.95428773191645869583E-01	1.70970543404441224307E-02	-5.71926117404305781283E-04
3	4.63399473359905636708E-06	-2.71994908488607703910E-09	

Table 2: Coefficients $\{d_k\}_{0\leq k\leq 10}$ of Equation (37) with 20 significant decimal digits Pugh [2004].

Algorithm 4: Accurate computation of $\log\Gamma(p)$ using Pugh's method.

Input: A real number $p > 0$.

Output: An accurate estimation of $\log\Gamma(p)$.

Require: Coefficients $\{d_k\}_{0\leq k\leq 10}$ defined in Table 2.

return $\log\left(2\sqrt{\frac{e}{\pi}}\left[d_0+\sum_{k=0}^{10}\frac{d_k}{p-1+k}\right]\right)-\left(p-\frac{1}{2}\right)+\left(p-\frac{1}{2}\right)\log\left(p+10.900511-\frac{1}{2}\right)$

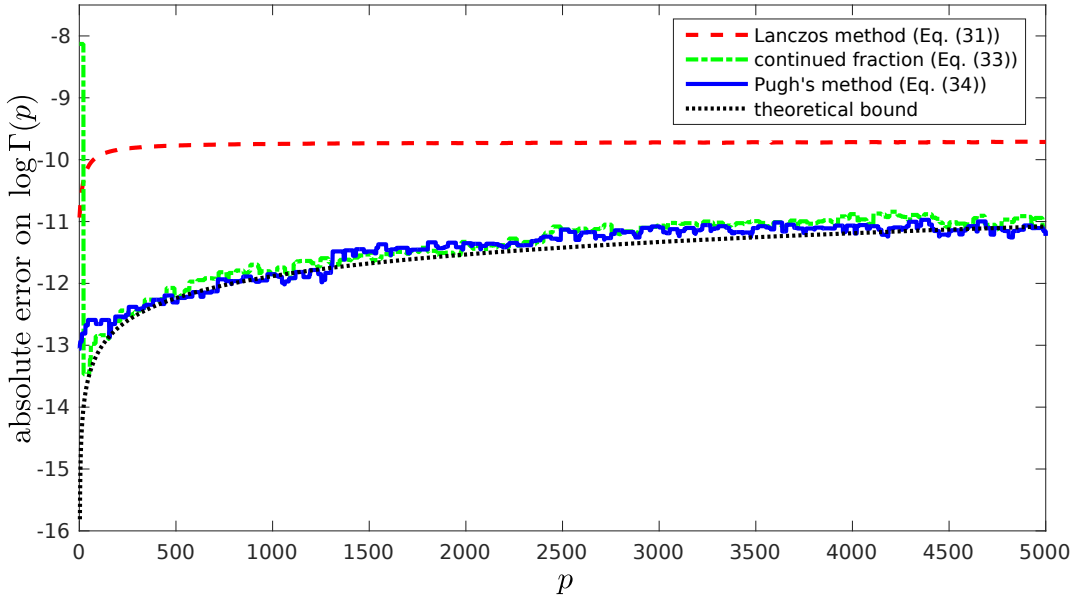


Fig. 5. Comparison of three algorithms estimating $\log\Gamma(p)$. We display the absolute error (estimated with Maple^{TM}) made on the estimation of $\log\Gamma(p)$ by the approximations (34), (36) and (37) as a function of $p \in \{1, 2, \ldots, 5000\}$. To ease the interpretation, the curves were smoothed by replacing the error for p by the maximum error on the range $\{p-20, \ldots, p+20\}$. We can observe that the best approximation is obtained with Pugh's formula (37) (blue curve), and that except for small values of p, it delivers an absolute error close to the theoretical limit $\varepsilon_{\text{machine}} \cdot \log\Gamma(p)$ (black curve).

4 Evaluation of the generalized incomplete gamma function

As stated above, the accurate evaluation of $I_{x,y}^{\mu,p}$ raises several issues. First, this integral can be estimated using the difference $A-B$ between two terms $A \geq B \geq 0$ involving the evaluation of the generalized upper and lower incomplete gamma functions using the relations (11)-(12). In that case, we must select one of these relations, in function of the parameters μ, x, y, p, to obtain the best possible estimate. This selection process is discussed in Section 4.1. Second, we must be aware that the accurate evaluation of A and B is not sufficient to guarantee an accurate evaluation of the difference $A-B$, because errors caused by cancellation arise when A and B are too close to each other, which happens in practice when $x \approx y$. In that case, we propose to approximate the integral $I_{x,y}^{\mu,p}$ using Romberg's numerical integration method, as discussed in Section 4.2. Last, we need a criterion to decide, in function of the parameters μ, x, y, p, which of the difference or Romberg's method will be used. This is the purpose of Section 4.3, where the resulting Algorithm 5, computing $I_{x,y}^{\mu,p}$ with a mantissa-exponent representation, is explicitly described.

4.1 Computing $I_{x,y}^{\mu,p}$ as a difference of generalized incomplete gamma functions

Given μ, x, y, p, we can estimate $G(p, \mu x)$ and $G(p, \mu y)$ with Algorithm 3. Then, considering the definition of G in Equation (5), there is only one relation from (11)-(12) that allows us to compute $I_{x,y}^{\mu,p}$, in combination (or not) with $\Gamma(p)$. The corresponding formulae are given in table 3. In each case, a mantissa-exponent representation

of $I_{x,y}^{\mu,p} = A - B$ can be obtained from a mantissa-exponent representation of $A = m_A \cdot e^{n_A}$ and $B = m_B \cdot e^{n_B}$ with

$$A - B = \rho_{\text{diff}} \cdot e^{\sigma_{\text{diff}}}, \quad \text{where} \quad \rho_{\text{diff}} = m_A - m_B\, e^{n_B - n_A}, \quad \sigma_{\text{diff}} = n_A\,. \tag{38}$$

case	value of A	value of B
(a) $\mu = -1$	$A = G(p,-y)\, e^{y+p\log y}$,	$B = G(p,-x)\, e^{x+p\log x}$
(b) $\mu = 1,\ p < p_{\lim}(x)$	$A = G(p,x)\, e^{-x+p\log x}$,	$B = G(p,y)\, e^{-y+p\log y}$
(c) $\mu = 1,\ p_{\lim}(x) \le p < p_{\lim}(y)$	$A = \Gamma(p)$,	$B = G(p,x)\, e^{-x+p\log x} + G(p,y)\, e^{-y+p\log y}$
(d) $\mu = 1,\ p_{\lim}(y) \le p$	$A = G(p,y)\, e^{-y+p\log y}$,	$B = G(p,x)\, e^{-x+p\log x}$

Table 3: Computing $I_{x,y}^{\mu,p}$ as the difference $A - B$.

In the cases (a,b,d), the mantissa-exponent representations of A and B are straightforward. In the case (c), the mantissa-exponent representation of $A = \Gamma(p)$ is $m_A = 1$, $n_A = \log\Gamma(p)$. For $B = G(p,x)e^{-x+p\log x} + G(p,y)e^{-y+p\log y}$, we choose the largest exponent and write $n_B = \max(-x+p\log x, -y+p\log y)$, then $m_B = G(p,x)e^{-x+p\log x-n_B} + G(p,y)e^{-y+p\log y-n_B}$. These formulae can be seen as the main part of Algorithm 5, which is written in the more general (and straightforward) case $\mu \in \mathbb{R}^*$.

4.2 Computing $I_{x,y}^{\mu,p}$ using Romberg's method

The computation of $I_{x,y}^{\mu,p}$ using a difference $I_{\text{diff}} = A - B$ described in Section 4.1 is not efficient when x and y are too close to each other due to errors caused by cancellation. Hopefully, in the case $x \approx y$ the integral $I_{x,y}^{\mu,p}$ can be efficiently estimated using the Romberg's methods Romberg [1955], which is a recursive numerical integration scheme that achieves a fast convergence rate. Using this method, the definite integral

$$I := \int_x^y f(s)\, ds \tag{39}$$

of a smooth function f is estimated by setting $R(0,0) = \frac{y-x}{2} \cdot (f(x) + f(y))$ and iterating for $n \ge 1$ the recursion

$$\begin{cases} R(n,0) &= \dfrac{1}{2} R(n-1,0) + h_n \displaystyle\sum_{j=1}^{2^n-1} f\left(x + (2j-1)\, h_n\right) \quad \text{where} \quad h_n = \dfrac{y-x}{2^n} \\ R(n,m) &= \dfrac{4^m R(n,m-1) - R(n-1,m-1)}{4^n - 1} \quad \text{for} \quad m \in \{1,2,\dots,n\}\,. \end{cases} \tag{40}$$

In practice, the scheme (40) is stopped when

$$\frac{|R(n,n) - R(n,n-1)|}{|R(n,n)|} \le \alpha \cdot \varepsilon_{\text{machine}}\,, \tag{41}$$

where $\alpha > 0$ denotes a tolerance parameter, or after a maximal number of iterations has been reached. Then, the integral (39) is estimated by $R(n,n)$. We implemented Romberg's method to a normalized version of $I_{x,y}^{\mu,p}$, that is for

$$f(s) = s^{p-1} \exp\left(-\mu s + \mu y - p\, \log y\right),$$

so that $I_{x,y}^{\mu,p}$ can be computed under a mantissa-exponent representation $I_{x,y}^{\mu,p} \approx \rho \cdot e^{\sigma}$ where $\sigma = -\mu y + p\, \log y$ and ρ denotes the output of the Romberg's approximation scheme.

4.3 Selection of the approximation method for the generalized incomplete gamma function

We saw in Sections 4.1 and 4.2 that the generalized incomplete gamma function $I_{x,y}^{\mu,p}$ could be estimated using a difference $I_{\text{diff}} = A - B$ or with Romberg's numerical integration method. We must now decide which approximation method should be used according to the values of x, y, μ, p. When A and B are too close to each

other, the computation of the difference $A - B$ suffers from cancellations. Indeed, for $A \geq B \geq 0$, a simple first order study of the relative accuracy of $A - B$ yields

$$\left| \frac{\Delta(A-B)}{A-B} \right| \leq \frac{2 \cdot |\Delta A|}{A-B} \leq \frac{2}{1 - B/A} \cdot \varepsilon_{\text{machine}} \cdot$$

Thus, we can see that we lose k digits of precision as soon as $1 - B/A < 2 \cdot 10^{-k}$. Based on this observation, we propose to avoid the approximation of $I_{x,y}^{\mu,p}$ by I_{diff} as soon as more than one digit of precision is lost, that is, as soon as $1 - B/A < 0.2$. Based on the same criterion, we decided to set the tolerance factor α equal to 10 in (41) in order to stop the Romberg iteration when the accuracy of the estimate is roughly $10 \cdot \varepsilon_{\text{machine}}$. This selection criterion was numerically tested, the resulting precision is displayed in Table 4. Finally, we describe in Algorithm 5 our computation method dedicated to the evaluation of $I_{x,y}^{\mu,p}$.

$\delta_r = \frac{y-x}{y}$	$\log_{10}$ of the maximal relative error	$\log_{10}$ of the mean relative error	average execution time (microseconds)	median execution time (microseconds)
10^{-2}	-11.2	-12.5	0.8 μs	0.7 μs
10^{-3}	-11.1	-12.3	1.0 μs	0.7 μs
10^{-4}	-11.8	-12.6	1.5 μs	1.7 μs
10^{-5}	-11.7	-12.5	1.5 μs	1.3 μs
10^{-6}	-11.8	-12.6	1.4 μs	1.3 μs
10^{-7}	-11.7	-12.5	1.4 μs	1.3 μs
10^{-8}	-11.8	-12.6	1.4 μs	1.3 μs
10^{-9}	-11.7	-12.5	1.4 μs	1.7 μs
10^{-10}	-11.7	-12.5	1.3 μs	1.2 μs
10^{-11}	-11.8	-12.5	1.3 μs	1.2 μs
10^{-12}	-11.7	-12.5	1.3 μs	1.2 μs
10^{-13}	-11.7	-12.5	0.9 μs	0.9 μs
10^{-14}	-11.7	-12.6	0.9 μs	0.9 μs
10^{-15}	-11.7	-12.5	0.8 μs	0.8 μs

Table 4: Control of maximum and mean relative errors associated to the computation of $I_{x,y}^{\mu,p}$. For several values of δ_r, we computed $I_{x,y}^{\mu,p}$ using Algorithm 5 for a large range of parameters ($\mu = \pm 1$, p integer in $[1, 1000]$, y integer in $[1, 1000]$, and x being the floating-point number closest to $y\,(1-\delta_r)$). For each value of δ_r, we display the maximal (second column) and the average (third column) relative errors observed over all computed values of $I_{x,y}^{\mu,p}$. We see that the relative error reached by Algorithm 5 over those simulations is quite low, in particular the approximation errors due to cancellation are avoided, thanks to the ability of Algorithm 5 to select automatically the most appropriate estimation method (difference or Romberg) according to the values of the parameters (x, y, μ, p). Note that a maximum relative error of 10^{-11} is roughly what we could at best expect considering the impact of the limited machine precision on the mantissa-exponent representation of the incomplete gamma functions, as discussed in Fig. 4. In columns four and five, we display the average and median execution time corresponding to the computation of $I_{x,y}^{\mu,p}$ over the whole range of tested parameters (computation was done using a a 3.1GHz intel™ i7-7920HQ processor).

5 Comparison with algorithm 435

In this section we compare Algorithm 5 with Algorithm 435, proposed in Fullerton [1972] for the evaluation of the generalized incomplete gamma function $I_{x,y}^{\mu,p}$. As far as we know, Fullerton's algorithm is the most recent work dedicated to the computation of an approximation to the I-integral. More precisely, Fullerton focused on the integral $J_{x,y}^{p}$, defined in (9), which is slightly different from $I_{x,y}^{\mu,p}$. However, the computation of $I_{x,y}^{\mu,p}$ using $J_{x,y}^{p}$, or conversely of $J_{x,y}^{p}$ using $I_{x,y}^{\mu,p}$, is straightforward, as we explained in Section 1.

Fullerton proposed an algorithm for the numerical evaluation of

$$\gamma'(p,x) = \int_0^x |s|^{p-1}\, e^{-s}\, ds, \quad -\infty < x \leq +\infty\,,$$

and thence the integral $J_{x,y}^{p}$ using the difference

$$J_{x,y}^{p} = e^x \int_x^y |s|^{p-1}\, e^{-s}\, ds = e^x\, (\gamma'(a,y) - \gamma'(a,x))$$

when $1 \leq p \leq 2$, and using a forward (when $p > 2$) or backward (when $p < 1$) recurrence relation, for approximating $J_{x,y}^{q}$ with $1 \leq q \leq 2$. In the case $1 \leq p \leq 2$, the evaluation of $\gamma'(p,x)$ relies on different approximation methods (such as continued fractions, Chebyshev polynomials, or asymptotic expansions), depending to the value of x.

Algorithm 5: Accurate computation of $I_{x,y}^{\mu,p} = \int_x^y s^{p-1}\, e^{-\mu s}\, ds$.

Input: Three numbers $\mu \in \mathbb{R}^*$, $x \in \mathbb{R}_+$, $y \in \mathbb{R}_+ \cup \{+\infty\}$ such as $y \geq x$, and a positive real number $p > 0$. Recall that the value $y = +\infty$ is allowed only when $\mu > 0$, and p must be an integer if $\mu < 0$.

Output: Two numbers $\rho \in \mathbb{R}$ and $\sigma \in \mathbb{R} \cup \{-\infty\}$ such as $I_{x,y}^{\mu,p} \approx \rho e^{\sigma}$.

Requirements: the function $p_{\lim} : \mathbb{R} \cup \{+\infty\} \to \mathbb{R} \cup \{+\infty\}$ defined in (22).

if $x = y$ *to machine precision* **then** $(\rho, \sigma) \leftarrow (0, -\infty)$
else
  $m_x \leftarrow G(p, \mu x)$ `// using Algorithm 3`
  $m_y \leftarrow G(p, \mu y)$ `// using Algorithm 3`
  $n_x \leftarrow -\mu x + p \log x$
  if $y < +\infty$ **then**
    $n_y \leftarrow -\mu y + p \log y$
  else
    $n_y \leftarrow -\infty$
  `// Evaluate` (m_A, n_A), (m_B, n_B) `and` $(\rho_{\text{diff}}, \sigma_{\text{diff}})$, `mantissa-exponent // representations of` A, B `and` I_{diff}, `such that` $A = m_A\, e^{n_A}$, $B = m_B\, e^{n_B}$,
  `//` $I_{\text{diff}} = \rho_{\text{diff}}\, e^{\sigma_{\text{diff}}} = A - B$, `and` $I_{x,y}^{\mu,p} \approx I_{\text{diff}}$.
  if $\mu < 0$ **then**
    $(m_A, n_A) \leftarrow (m_y, n_y)$
    $(m_B, n_B) \leftarrow (m_x, n_x)$
  else if $\mu > 0$ **then**
    if $p < p_{lim}(\mu x)$ **then**
      $(m_A, n_A) \leftarrow (m_x, n_x)$
      $(m_B, n_B) \leftarrow (m_y, n_y)$
    else if $p_{lim}(\mu x) \leq p < p_{lim}(\mu y)$ **then**
      $(m_A, n_A) \leftarrow (1, \log \Gamma(p))$
      $n_B \leftarrow \max(n_x, n_y)$
      if $n_B = -\infty$ **then** $m_B \leftarrow 0$ `// may happen when` $x = 0$ `and` $y = +\infty$
      else $m_B \leftarrow m_x\, e^{n_x - n_B} + m_y\, e^{n_y - n_B}$
    else
      $(m_A, n_A) \leftarrow (m_y, n_y)$
      $(m_B, n_B) \leftarrow (m_x, n_x)$
  $(\rho_{\text{diff}}, \sigma_{\text{diff}}) \leftarrow \left(m_A - m_B \cdot e^{n_B - n_A}, n_A\right)$
  `// Check whether or not the estimation of` $I_{x,y}^{\mu,p}$ `by` $I_{\text{diff}} = A - B$ `involves a`
  `// significant loss of precision. If` $1 - B/A < 0.2$, `estimate` $I_{x,y}^{\mu,p}$ `using`
  `// Romberg's method instead of the difference` I_{diff}
  if $y < +\infty$ *and* $\rho_{diff}/m_A < 0.2$ **then**
    set $\sigma = -\mu y + p \log y$ and compute an estimate ρ of the normalized integral $I_{x,y}^{\mu,p} \cdot e^{-\sigma}$ using Romberg's method described in Section 4.2.
  else $(\rho, \sigma) \leftarrow (\rho_{\text{diff}}, \sigma_{\text{diff}})$

return (ρ, σ)

As already reported in Schoene [1978], Algorithm 435 suffers from numerical instabilities, when $p > 2$. We indeed observed in our own numerical experiments, presented in Tables 5 and 6, computed values with very low accuracy, or incorrect sign, typically when $p \geq 10$, or when $x \leq p \leq y$. We also also observed some overflow issues, for instance when working with $p \geq 100$, which is of course not to be considered as a failure of Algorithm 435 since this algorithm was developed in single precision.

In the experiments of Table 5 and Table 6, we evaluated the integral $I_{x,y}^{\mu,p}$, for several sets of parameters x, y, μ, p, using both Fullerton's Algorithm 435 and Algorithm 5. The accuracy of the returned result was controlled using the softwares Maple$^{\text{TM}}$ (with the instruction

```
evalf(Int(s^(p-1)*exp(-mu*s),s=x..y,digits=30));
```

to approximate the integral with 30 digits of precision), and Mathematica$^{\text{TM}}$ in Wolfram Research Inc [1998] for the online evaluation of $I_{x,y}^{\mu,p}$. Considering that Fullerton's algorithm was developed in single precision and in 1972 (more than 10 years before the IEEE 754 Standard for Floating-Point Arithmetic was established), it is not surprising to see its accuracy largely outperformed by Algorithm 5. As mentioned earlier, this comparison is motivated by the fact that Fullerton's algorithm is the most recent one focusing on the computation of the I-integral.

	Parameters	Algorithm 435 in Fullerton [1972]	Relative error	Algorithm 5	Relative error
(a) $\mu = 1$	$x = 9, y = 11, p = 1$	1.067081029759719E-04	$3 \cdot 10^{-9}$	1.0670810329643395E-04	$6 \cdot 10^{-16}$
	$x = 9, y = 11, p = 5$	9.567113518714904E-01	$1 \cdot 10^{-4}$	9.5661698023023700E-01	$1 \cdot 10^{-15}$
	$x = 9, y = 11, p = 10$	1.085447578125000E+05	$2 \cdot 10^{-1}$	8.9594201765236983E+04	$1 \cdot 10^{-14}$
	$x = 9, y = 11, p = 12$	1.632943040000000E+08	≥ 1	8.9310494815538749E+06	$3 \cdot 10^{-15}$
	$x = 9, y = 11, p = 14$	-2.977905664000000E+10	≥ 1	9.0203414117080807E+08	$2 \cdot 10^{-15}$
	$x = 100, y = 120, p = 1$	3.783505853677006E-44	$2 \cdot 10^{-2}$	3.7200759683531697E-44	$5 \cdot 10^{-15}$
	$x = 100, y = 120, p = 5$	3.873433252162870E-36	$3 \cdot 10^{-9}$	3.8734332644314730E-36	$4 \cdot 10^{-15}$
	$x = 100, y = 120, p = 10$	4.083660502797843E-26	$2 \cdot 10^{-8}$	4.0836605881700520E-26	$8 \cdot 10^{-15}$
	$x = 100, y = 120, p = 20$	4.579807864502072E-06	$9 \cdot 10^{-8}$	4.5798082802928473E-06	$2 \cdot 10^{-14}$
(b) $\mu = -1$	$x = 5, y = 10, p = 1$	2.187916015625000E+04	$5 \cdot 10^{-5}$	2.1878052635704163E+04	$1 \cdot 10^{-15}$
	$x = 5, y = 10, p = 3$	1.803647750000000E+06	$3 \cdot 10^{-7}$	1.8036471714694066E+06	$1 \cdot 10^{-16}$
	$x = 5, y = 10, p = 10$	1.129511596851200E+13	$4 \cdot 10^{-8}$	1.1295115549498505E+13	$4 \cdot 10^{-15}$
	$x = 20, y = 25, p = 1$	7.151973171200000E+10	$3 \cdot 10^{-8}$	7.1519734141975967E+10	$2 \cdot 10^{-15}$
	$x = 20, y = 25, p = 10$	2.068890077987267E+23	$3 \cdot 10^{-2}$	2.0016822370845540E+23	$9 \cdot 10^{-16}$
	$x = 20, y = 25, p = 20$	1.821993954177914E+37	$2 \cdot 10^{-1}$	1.4733948083664500E+37	$1 \cdot 10^{-15}$

Table 5: Comparison between Algorithm 435 in Fullerton [1972] and Algorithm 5, for the computation of $I_{x,y}^{\mu,p}$. **(a)** We tested, for $\mu = 1$, $(x, y) = (9, 11)$, and $(x, y) = (100, 120)$, different integer values of p. In the second column, we display the values of $I_{x,y}^{\mu,p}$ returned by Fullerton's Algorithm, slightly adapted to compute $I_{x,y}^{\mu,p}$ instead of $J_{x,y}^{\mu,p}$. The corresponding relative errors, evaluated using Mathematica$^{\text{TM}}$ and Maple$^{\text{TM}}$ softwares (both softwares yield the same relative error), are displayed on the third column. We can see that some numerical instabilities arise when $x \leq p$. Some inaccurate results are also observed for small values of p, when $x = 100, y = 120, p = 1$ (but also for many other values of (x, y, p), not represented here). In the fourth column, we display the values returned by Algorithm 5, using a C implementation with standard double precision. The relative errors reached by Algorithm 5 (last column) are nearly optimal, as they are close to the optimality bounds given in (31), which estimate the minimum error achievable with the mantissa-exponent representation. **(b)** Same experiment in the case $\mu = -1$. We tested, for $(x, y) = (5, 10)$ and $(x, y) = (20, 25)$, different values of p. We can observe that Fullerton's algorithm rapidly delivers inaccurate estimates as p increases. In contrast, the relative errors reached by Algorithm 5 remain nearly optimal.

Parameters setting	Algorithm 435 in Fullerton [1972]	Relative error	Algorithm 5	Method	Relative error
$\mu = 1, y = 5, p = 10$					
$x = d(5 - 10^{0})$	8.598737304687500E+03	$2 \cdot 10^{-8}$	8.5987371691242424 3E+03	D	$7 \cdot 10^{-17}$
$x = d(5 - 10^{-1})$	1.263989379882812E+03	$8 \cdot 10^{-7}$	1.2639903706449722 4E+03	R	$7 \cdot 10^{-17}$
$x = d(5 - 10^{-3})$	1.315382766723632E+01	$7 \cdot 10^{-5}$	1.3154789325749973 7E+01	R	$8 \cdot 10^{-16}$
$x = d(5 - 10^{-4})$	1.317400574684143E+00	$1 \cdot 10^{-3}$	1.3159526536590288 1E+00	R	$1 \cdot 10^{-15}$
$x = d(5 - 10^{-5})$	1.322335302829742E-01	$5 \cdot 10^{-3}$	1.3160000091941087 6E-01	R	$5 \cdot 10^{-16}$
$x = d(5 - 10^{-6})$	1.179141830652952E-02	$1 \cdot 10^{-1}$	1.3160047470407800 6E-02	R	$6 \cdot 10^{-16}$
$\mu = 1, y = 17, p = 17$					
$x = d(17 - 10^{0})$	3.725839564800000E+12	$8 \cdot 10^{-1}$	2.0551230250735383 3E+12	R	$5 \cdot 10^{-16}$
$x = d(17 - 10^{-1})$	2.998156984320000E+11	$5 \cdot 10^{-1}$	2.0202925544705441 3E+11	R	$1 \cdot 10^{-15}$
$x = d(17 - 10^{-3})$	2.941651456000000E+09	$5 \cdot 10^{-1}$	2.0146022707112157 3E+09	R	$4 \cdot 10^{-17}$
$x = d(17 - 10^{-4})$	2.928078720000000E+08	$5 \cdot 10^{-1}$	2.0145489618187716 6E+08	R	$8 \cdot 10^{-16}$
$\mu = -1, y = 21, p = 10$					
$x = d(21 - 10^{0})$	5.859836137154984E+20	$5 \cdot 10^{-2}$	5.5623377927217197 9E+20	D	$4 \cdot 10^{-15}$
$x = d(21 - 10^{-1})$	1.025911814748488E+20	$5 \cdot 10^{-2}$	9.7609411144076853 2E+19	R	$1 \cdot 10^{-16}$
$x = d(21 - 10^{-3})$	1.099215584270221E+18	$5 \cdot 10^{-2}$	1.0467611548967834 9E+18	R	$5 \cdot 10^{-16}$
$x = d(21 - 10^{-5})$	1.045801880623513E+16	$2 \cdot 10^{-3}$	1.0475015408053944 0E+16	R	$7 \cdot 10^{-16}$

Table 6: Comparison between Fullerton's algorithm and Algorithm 5, for the computation of $I_{x,y}^{\mu,p}$ when $x \approx y$. In this last experiment, we compute $I_{x,y}^{\mu,p}$ in the case $x \approx y$. The notation $d(s)$ used in the left column denotes the double-precision floating-point number that is closest to s. The fifth row (Method) indicate the computation method that was used in Algorithm 5 to compute the $I_{x,y}^{\mu,p}$ integral (R for Romberg approximation and D for differences). We can see that the relative error reached by Algorithm 435 gets worse as x and y get close to each other, and as already remarked before, Algorithm 435 is very inaccurate when $\mu x < p < \mu y$. In contrast, the relative errors observed with Algorithm 5 never exceed $8 \cdot 10^{-16}$ (which corresponds to less than one digit of precision), thanks to the use of Romberg's numerical integration method which takes over to avoid cancellations when x and y are very close to each other.

6 Application to image denoising

This work was initially motivated by an image processing application presented in Abergel et al. [2015], which aims at reconstructing a gray-level image given the measurement of its intensity values corrupted by a Poisson noise. This kind of image denoising model is typically interesting for restoring images acquired in low-light conditions, for example astronomical and medical images. The restoration model derived in Abergel et al. [2015] results in an iterative scheme which requires the computation, at each iteration and for each pixel of the image, of a ratio of generalized incomplete gamma functions. More precisely, from an initial (noisy) image $u_0 : \Omega \to \mathbb{R}$ (where $\Omega \subset \mathbb{Z}^2$ is the discrete image domain and $u_0(x)$ represents the intensity of u_0 at the pixel $x \in \Omega$), we build a sequence of images u_n such that

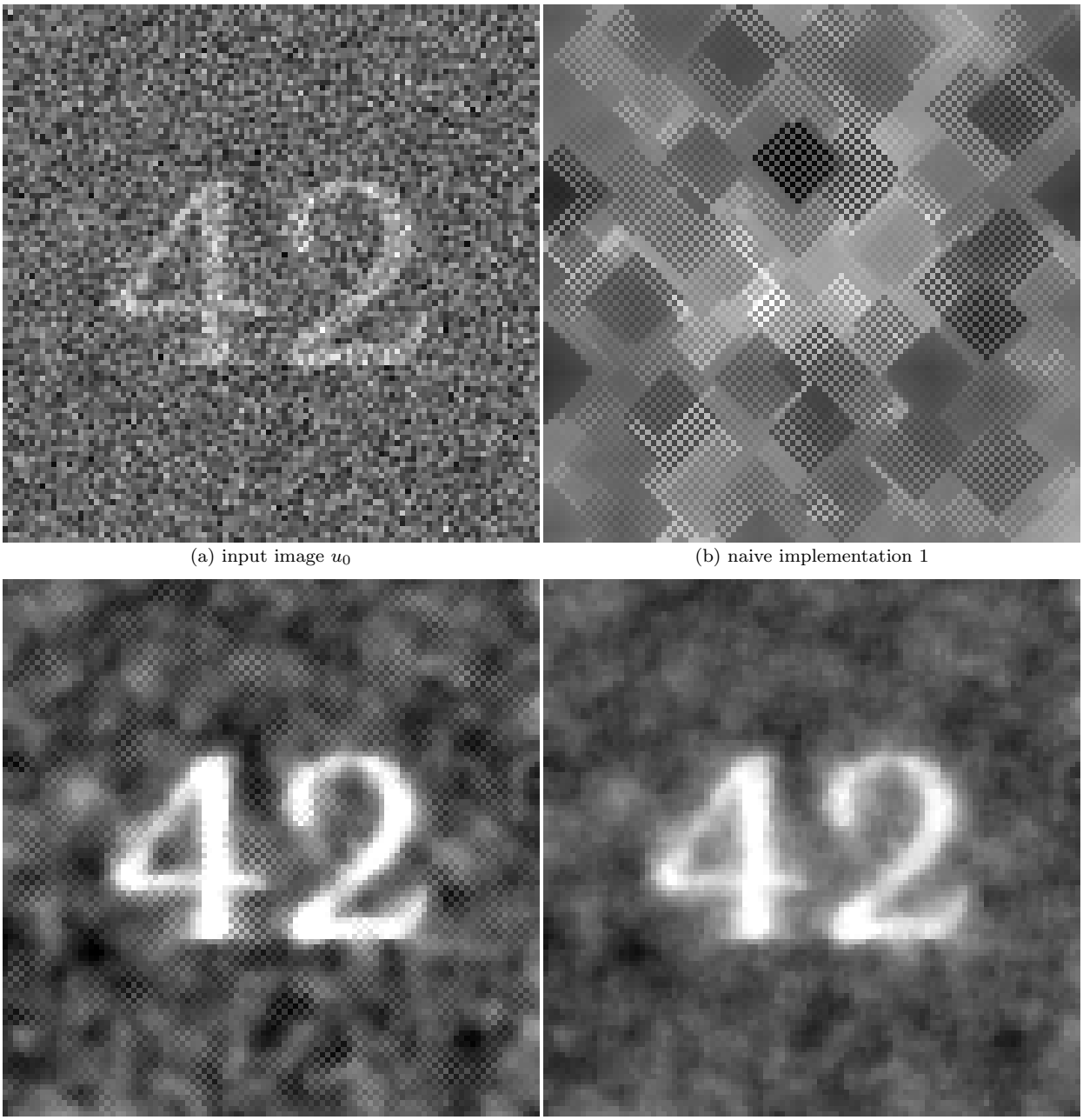
$$\forall n \geq 0, \forall x \in \Omega, \quad u_{n+1}(x) = R(x) := \frac{\displaystyle\sum_{k=1}^{5} c_k \, I_{a_{k-1},a_k}^{\mu_k, u_0(x)+2}}{\displaystyle\sum_{k=1}^{5} c_k \, I_{a_{k-1},a_k}^{\mu_k, u_0(x)+1}}, \tag{42}$$

where $\{a_k, c_k\}_{1 \leq k \leq 5}$ are some positive coefficients that explicitly depend on the image at the previous iteration (u_n) and $\{\mu_k\}_{1 \leq k \leq 5}$ are nonzero real numbers (see Abergel et al. [2015] for more details concerning theses coefficients). Notice that this scheme involves evaluating a huge number of generalized incomplete gamma functions: if u_0 is a 1000×1000 image, 10^9 evaluations of $I_{x,y}^{\mu,p}$ integrals are required to perform 100 iterations of (42).

The main difficulty encountered when computing (42) is that the ratio $R(x)$ can exhibit a non-representable numerator and denominator (due to underflow or overflow), although the actual value of the ratio is representable in the floating-point arithmetic. In practice, this yields dramatic errors when one tries to evaluate the numerator and the denominator separately in double precision before computing the ratio, as illustrated in Fig. 6 (a). This issue cannot be solved by applying the standard normalization (division by the complete gamma function) to the numerator and the denominator, as we showed that such a normalization may produce severe underflow issues. In contrast, by evaluating all integrals $I_{x,y}^{\mu,p}$ using a mantissa-exponent representation $I_{x,y}^{\mu,p} = \rho \cdot e^{\sigma}$ as in Algorithm 5, we are able to avoid this undesirable behavior (see more details in [Abergel, 2016, Chap. 4]). Similarly, the systematic use of differences of incomplete gamma functions to evaluate the integrals $I_{x,y}^{\mu,p}$ produces instabilities in the algorithms (Fig. 6 (b)), due to cancellations as discussed in Section 4.3. In contrast, the association with Romberg's method proposed in Algorithm 5 leads to accurate $I_{x,y}^{\mu,p}$ estimates and results in stable iterations of the scheme (42) (Fig. 6 (c)).

Acknowledgments

The authors would like to thank the GDS Mathrice 2754 as well as MathStic and LAGA (Laboratoire Analyse, Géométrie et Applications) at Université Paris 13, for having kindly provided us an access to the computing server GAIA. The authors are also very grateful to the anonymous reviewers and the Editor-in-Chief for their careful reading of this paper, as well as for their useful suggestions that helped to improve this paper and the associated algorithm.



(a) input image u_0 (b) naive implementation 1

(c) naive implementation 2 (d) Algorithm 5

Fig. 6. Improper incomplete gamma estimates lead to numerical instabilities. The image u_0 displayed in **(a)** corresponds to the simulated low-light observation of a synthetic image. We applied to u_0 the denoising scheme proposed in Abergel et al. [2015], which amounts to iterate a few hundred times the recursion (42). We implemented this algorithm in three different ways. **(b):** straightforward implementation of (42), where each integral $I_{x,y}^{\mu,p}$ is computed as the difference of incomplete gamma functions evaluated with the algorithm proposed in [Press et al., 1992], and the ratio is evaluated directly; **(c):** similar implementation, except that a mantissa-exponent representation is used for each integral to avoid underflow/overflow issues before computing the ratio; **(d):** Algorithm 5 is used to compute each integral and estimate the ratio. This requires a huge number of evaluations of the generalized incomplete gamma function which, in turn, needs the careful handling of underflow/overflow and cancellation issues as provided by Algorithm 5. Failure to do this leads to the appearance of undesired checkerboard patterns as seen in (b) and (c) which are caused by numerical instabilities.

References

R. Abergel. *Several mathematical models and fast algorithms for image processing*. PhD thesis, Université Paris Descartes, 2016.

R. Abergel, C. Louchet, L. Moisan, and T. Zeng. Total variation restoration of images corrupted by poisson noise with iterated conditional expectations. In *Scale Space and Variational Methods in Computer Vision*, pages 178–190. Springer, 2015.

M. Abramowitz and I. A. Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*. Number 55. Courier Corporation, 1964.

G. P. Bhattacharjee. Algorithm as 32: The incomplete gamma integral. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 19(3):285–287, 1970.

M. Bleicher and P. Nicolini. Large extra dimensions and small black holes at the lhc. In *Journal of Physics: Conference Series*, volume 237, page 012008. IOP Publishing, 2010.

C. J. Cannon and I. M. Vardavas. The effect of redistribution on the emission peaks from chromospheric-type stellar atmospheres. *Astronomy and Astrophysics*, 32:85, 1974.

B. W. Char. On stieltjes's continued fraction for the gamma function. *Mathematics of Computation*, 34(150): 547–551, 1980.

M. A. Chaudhry and S. M. Zubair. *On a class of incomplete gamma functions with applications*. CRC press, 2001.

G. W. Collins. *Fundamentals of Stellar Astrophysics*. W. H. Freeman and Co., New York, NY, 1989.

A. Cuyt, F. Backeljauw, and C. Bonan-Hamada. *Handbook of continued fractions for special functions*. Springer Science & Business Media, 2008.

DLMF. NIST Digital Library of Mathematical Functions. http://dlmf.nist.gov/, Release 1.0.10, 2015. Online companion to Olver et al. [2010].

W. Fullerton. Algorithm 435: Modified incomplete gamma function [s14]. *Commun. ACM*, 15(11):993–995, November 1972. ISSN 0001-0782. doi: 10.1145/355606.361891.

W. Gautschi. A computational procedure for incomplete gamma functions. *ACM Trans. Math. Software*, 5(4): 466–481, December 1979. ISSN 0098-3500.

W. Gautschi. The incomplete gamma functions since tricomi. In *Tricomi's Ideas and Contemporary Applied Mathematics, Atti dei Convegni Lincei, Accademia Nazionale dei Lincei*, volume 147, pages 203–237, 1998.

A. Gil, J. Segura, and N. M. Temme. Efficient and accurate algorithms for the computation and inversion of the incomplete gamma function ratios. *SIAM Journal on Scientific Computing*, 34(6):A2965–A2981, 2012.

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 Transactions on Mathematical Software*, 43(3), 2016.

I. I. Guseinov and B. A. Mamedov. Evaluation of Incomplete Gamma Functions Using Downward Recursion and Analytical Relations. *Journal of Mathematical Chemistry*, 36(4):341–346, August 2004.

J. G. Hills. Effect of binary stars on the dynamical evolution of stellar clusters. ii-analytic evolutionary models. *The Astronomical Journal*, 80:1075–1080, 1975.

W. B. Jones and W. J. Thron. *Continued fractions: analytic theory and applications*. Number 11 in Encyclopedia of mathematics and its applications. Addison-Wesley Pub. Co, Reading, Mass, 1980. ISBN 978-0-201-13510-7.

L. Kissel, R. H. Pratt, and S. C. Roy. Rayleigh scattering by neutral atoms, 100 ev to 10 mev. *Phys. Rev. A*, 22:1970–2004, Nov 1980. doi: 10.1103/PhysRevA.22.1970. URL http://link.aps.org/doi/10.1103/PhysRevA.22.1970.

C. Lanczos. A precision approximation of the gamma function. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 1(1):86–96, 1964.

W. J. Lentz. Generating bessel functions in mie scattering calculations using continued fractions. *Applied Optics*, 15(3):668–671, 1976.

V. Linetsky. Pricing equity derivatives subject to bankruptcy. *Mathematical Finance*, 16(2):255–282, 2006.

Y. Moreno, R. Pastor-Satorras, and A. Vespignani. Epidemic outbreaks in complex heterogeneous networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 26(4):521–529, 2002.

F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark, editors. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, 2010. Print companion to DLMF.

Anne Philippe. Simulation of right and left truncated gamma distributions by mixtures. *Statistics and Computing*, 7(3):173–181, 1997.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge ; New York, 2nd edition, 1992. ISBN 978-0-521-43108-8 978-0-521-43720-2.

G. R. Pugh. *An analysis of the Lanczos gamma approximation*. PhD thesis, University of British Columbia, 2004.

A. Robin, L. Moisan, and S. Le Hégarat-Mascle. An a-contrario approach for sub-pixel change detection in satellite imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):1977–1993, 2010.

W. Romberg. Vereinfachte numerische integration. *Det Kongelige Norske Videnskabers Selskab Forhandlinger*, 28(7):30–36, 1955.

A. Y. Schoene. Remark on "algorithm 435: Modified incomplete gamma function [s14]". *ACM Trans. Math. Softw.*, 4(3):296–304, September 1978. ISSN 0098-3500. doi: 10.1145/355791.355803.

I. Thompson. Algorithm 926: Incomplete gamma functions with negative arguments. *ACM Trans. Math. Software*, 39(2):14:1–14:9, February 2013. ISSN 0098-3500. doi: 10.1145/2427023.2427031.

I. J. Thompson and A. R. Barnett. Coulomb and bessel functions of complex arguments and order. *Journal of Computational Physics*, 64(2):490–509, 1986.

F. G. Tricomi. Sulla funzione gamma incompleta. *Annali di Matematica Pura ed Applicata*, 31(1):263–279, 1950.

Roel Verbelen, Lan Gong, Katrien Antonio, Andrei Badescu, and Sheldon Lin. Fitting mixtures of erlangs to censored and truncated data using the em algorithm. *ASTIN Bulletin: The Journal of the IAA*, 45(3): 729–758, 2015.

S. Winitzki. Computing the incomplete gamma function to arbitrary precision. In *Proceedings of the 2003 International Conference on Computational Science and Its Applications: Part I*, ICCSA'03, pages 790–798, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3-540-40155-5.

Wolfram Research Inc. Generalized incomplete gamma function, 1988. URL http://reference.wolfram.com/language/ref/Gamma.html. (documentation page).

Wolfram Research Inc. Generalized incomplete gamma function, 1998. URL http://functions.wolfram.com/webMathematica/FunctionEvaluation.jsp?name=Gamma3. (online evaluation page).