On the Computation of Confluent Hypergeometric Functions for Large Imaginary Part of Parameters b and z

Guillermo Navas-Palencia[1] and Argimiro Arratia[2]

[1] Numerical Algorithms Group Ltd, UK, and
Dept. Computer Science, Universitat Politècnica de Catalunya, Spain
guillermo.navas@nag.co.uk
[2] Dept. Computer Science, Universitat Politècnica de Catalunya, Spain
argimiro@cs.upc.edu

Abstract. We present an efficient algorithm for the confluent hypergeometric functions when the imaginary part of b and z is large. The algorithm is based on the steepest descent method, applied to a suitable representation of the confluent hypergeometric functions as a highly oscillatory integral, which is then integrated by using various quadrature methods. The performance of the algorithm is compared with open-source and commercial software solutions with arbitrary precision, and for many cases the algorithm achieves high accuracy in both the real and imaginary parts. Our motivation comes from the need for accurate computation of the characteristic function of the Arcsine distribution or the Beta distribution; the latter being required in several financial applications, for example, modeling the loss given default in the context of portfolio credit risk.

Keywords: confluent hypergeometric function, complex numbers, steepest descent

1 Introduction

The confluent hypergeometric function of the first kind ${}_1F_1(a;b;z)$ or Kummer's function $M(a,b,z)$ arises as one of the solutions of the limiting form of the hypergeometric differential equation, $z\dfrac{d^2w}{dz^2}+(b-z)\dfrac{dw}{dz}-aw=0$, for $b\notin\mathbb{Z}^-\cup\{0\}$, see [1, §13.2.1]. Another standard solution is $U(a,b,z)$, which is defined by the property $U(a,b,z)\sim z^{-a},\ z\to\infty$, $|\mathrm{ph}\, z|\leq(3/2)\pi-\delta$, where δ is an arbitrary small positive constant such that $0<\delta\ll 1$. Different methods have been devised for evaluating the confluent hypergeometric functions, although we are mainly interested in methods involving their integral representations. As stated in [1, §13.4.1, §13.4.4], the functions ${}_1F_1(a;b;z)$ and $U(a,b,z)$ have the following integral representations, respectively

$$ {}_1F_1(a;b;z)=\frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\int_0^1 e^{zt}t^{a-1}(1-t)^{b-a-1}\,dt,\quad \Re(b)>\Re(a)>0 \quad (1)$$

$$U(a,b,z) = \frac{1}{\Gamma(a)} \int_0^\infty e^{-zt} t^{a-1} (1+t)^{b-a-1} \, dt, \quad \Re(a) > 0, |\mathrm{ph} z| < \frac{1}{2}\pi \tag{2}$$

Furthermore, Kummer's transformations (cf. [1, §13.2(vii)]), can be applied in situations where the parameters are not valid for some methods, or when the regime of parameters causes numerical instability,

$${}_1F_1(a;b;z) = e^z \, {}_1F_1(b-a;b;-z), \quad U(a,b,z) = z^{1-b} U(a-b+1, 2-b, z) \tag{3}$$

In other cases, recurrences relations can be applied (cf. [1, §13.3]). A recent survey of numerical methods for computing the confluent hypergeometric function can be found in [9] and [10], where the authors provide *roadmaps* with recommendations for which methods should be used in each situation.

2 Algorithm

The presented method for the computation of the confluent hypergeometric functions is based on the application of suitable transformations to highly oscillatory integrals and posterior numerical evaluation by means of quadrature methods. Some direct methods for ${}_1F_1(a;b;iz)$ can be applied for moderate values of $|\Im(z)|$, however a more general approach is the use of the numerical steepest descent method, which turns out to be very effective for the regime of parameters of interest. First, we briefly explain the path of steepest descent. Subsequently, we introduce the steepest descent integrals for those cases where $|\Im(b)|$, $|\Im(z)| \to \infty$.

2.1 Path of steepest descent

For this work we consider the ideal case for analytic integrand with no stationary points. We follow closely the theory developed in [3]. Let us consider the oscillatory integral

$$I = \int_\alpha^\beta f(x) e^{i\omega g(x)} \, dx \tag{4}$$

where $f(x)$ and $g(x)$ are smooth functions. By applying the steepest descent method, the interval of integration is substituted by a union of contours on the complex plane, such that along these contours the integrand is non-oscillatory and exponentially decaying. Given a point $x \in [\alpha, \beta]$, we define the path of steepest descent $h_x(p)$, parametrized by $p \in [0, \infty)$, such that the real part of the phase function $g(x)$ remains constant along the path. This is achieved by solving the equation $g(h_x(p)) = g(x) + ip$. If $g(x)$ is easily invertible, then $h_x(p) = g^{-1}(g(x) + ip)$, otherwise root-finding methods are employed, see [3, §5.2]. Along this path of steepest descent, integral (4) is transformed to

$$\begin{aligned} I[f; h_x] &= e^{i\omega g(x)} \int_0^\infty f(h_x(p)) h_x'(p) e^{-\omega p} \, dp \\ &= \frac{e^{i\omega g(x)}}{\omega} \int_0^\infty f\left(h_x\left(\frac{q}{\omega}\right)\right) h_x'\left(\frac{q}{\omega}\right) e^{-q} \, dq \end{aligned} \tag{5}$$

and $I = I[f; h_\alpha] - I[f; h_\beta]$ with both integrals well behaved. In the cases where $\beta = \infty$, this parametrization gives $I = I[f; h_\alpha] - 0$.

A particular case of interest is when $g(x) = x$. Then the path of steepest descent can be taken as $h_x(p) = x + ip$, and along this path (4) is written as

$$\int_\alpha^\beta f(x)e^{i\omega x}\,dx = \frac{ie^{i\omega\alpha}}{\omega}\int_0^\infty f\left(\alpha+i\frac{q}{\omega}\right)e^{-q}\,dq - \frac{ie^{i\omega\beta}}{\omega}\int_0^\infty f\left(\beta+i\frac{q}{\omega}\right)e^{-q}\,dq \quad (6)$$

2.2 $U(a, b, z)$, $\Im(z) \to \infty$

Integral representation (2) can be transformed into a highly oscillatory integral

$$U(a,b,z) = \frac{1}{\Gamma(a)}\int_0^\infty e^{-\Re(z)t}t^{a-1}(1+t)^{b-a-1}e^{-i\Im(z)t}\,dt \quad (7)$$

Taking $g(t) = t$, $g'(t) = 1 \neq 0$ and there are no stationary points. Therefore, in this case we only have one endpoint and the steepest descent integral obtained by (6) is reduced to a single line integral,

$$U(a,b,z) = \frac{i}{\omega\Gamma(a)}\int_0^\infty e^{-\Re(z)i\frac{q}{\omega}}\left(i\frac{q}{\omega}\right)^{a-1}\left(1+i\frac{q}{\omega}\right)^{b-a-1}e^{-q}\,dq \quad (8)$$

2.3 $U(a, b, z)$, $\Im(b) \to \infty$

In this case, the path of integration is modified to avoid a singularity at $t = 0$, as can be seen after performing the transformation to a highly oscillatory integral,

$$\begin{aligned} U(a,b,z) &= \frac{1}{\Gamma(a)}\int_0^\infty e^{-zt}t^{a-1}(1+t)^{b-a-1}\,dt \\ &= \frac{e^z}{\Gamma(a)}\int_1^\infty e^{-zt}(t-1)^{a-1}t^{\Re(b)-a-1}e^{i\Im(b)\log(t)}\,dt \end{aligned} \quad (9)$$

Now, we solve the path of steepest descent at $t = 1$ with $g(t) = \log(t)$, which in this case results trivial, $h_1(p) = e^{\log(1)+ip} = e^{ip}$ and $h_1'(p) = ie^{ip}$.

Likewise, no stationary points besides $t = \infty$ are present, and therefore there are no further contributions. The steepest descent integral is given by

$$U(a,b,z) = \frac{ie^z}{\omega\Gamma(a)}\int_0^\infty e^{\phi(q,\omega)}(\mu(q,\omega)-1)^{a-1}\mu(q,\omega)^{\Re(b)-a-1}e^{-q}\,dq \quad (10)$$

where $\mu(q,\omega) = i\frac{q}{\omega}$ and $\phi(q,\omega) = -ze^{\mu(q,\omega)} + \mu(q,\omega)$

2.4 ${}_1F_1(a, b, z)$, $\Im(z) \to \infty$

Similarly, we transform integral (1) into a highly oscillatory integral

$${}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\int_0^1 e^{\Re(z)t}t^{a-1}(1-t)^{b-a-1}e^{i\Im(z)t}\,dt \quad (11)$$

Again with $g(t) = t$ and the transformation stated in (6), we obtain, after some calculations, the steepest descent integrals given by

$$
\begin{aligned}
{}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\frac{i}{\omega}\Bigg[& \int_0^\infty e^{\Re(z)i\frac{q}{\omega}}\left(i\frac{q}{\omega}\right)^{a-1}\left(1-i\frac{q}{\omega}\right)^{b-a-1}e^{-q}\,dq \\
& -e^{i\omega}\int_0^\infty e^{\Re(z)(1+i\frac{q}{\omega})}\left(1+i\frac{q}{\omega}\right)^{a-1}\left(-i\frac{q}{\omega}\right)^{b-a-1}e^{-q}\,dq\Bigg] \qquad (12)
\end{aligned}
$$

2.5 ${}_1F_1(a,b,z)$, $\Im(b) \to \infty$

For this case we can use the following connection formula [1, §13.2.41], valid for all $z \neq 0$,

$$
\frac{1}{\Gamma(b)}\,{}_1F_1(a;b;z) = \frac{e^{\mp\pi i a}}{\Gamma(b-a)}U(a,b,z) + \frac{e^{\pm\pi i(b-a)}}{\Gamma(a)}e^z U(b-a,b,ze^{\pm\pi i}) \qquad (13)
$$

2.6 Numerical quadrature schemes

Adaptive quadrature for oscillatory integrals. The integrand in (11) can be rewritten in terms of its real and imaginary parts to obtain two separate integrals with trigonometric weight functions, the oscillatory factor, given the property,

$$
\int_0^1 f(t)e^{i\omega t}\,dt = \int_0^1 f(t)\cos(\omega t)\,dt + i\int_0^1 \sin(\omega t)\,dt \qquad (14)
$$

Thus, we obtain the following integral representation for ${}_1F_1(a;b;z)$ when $|\Im(z)| \to \infty$,

$$
\begin{aligned}
{}_1F_1(a;b;z) = \frac{\Gamma(b)}{\Gamma(a)\Gamma(b-a)}\Bigg[& \int_0^1 e^{\Re(z)t}t^{a-1}(1-t)^{b-a-1}\cos(\Im(z)t)\,dt \\
& + i\int_0^1 e^{\Re(z)t}t^{a-1}(1-t)^{b-a-1}\sin(\Im(z)t)\,dt\Bigg] \qquad (15)
\end{aligned}
$$

These type of integrals can be solved using specialized adaptive routines, such as the routine `gsl_integration_qawo` from the GNU Scientific Library [2]. This routine combines Clenshaw-Curtis quadrature with Gauss-Kronrod integration. Numerical examples can be found in [8], which show that this method works reasonably well for moderate values of $|\Im(z)|$. Unfortunately, this method cannot be directly applied to $U(a,b,z)$, and Kummer's transformation [1, §13.2.42], valid for $b \notin \mathbb{Z}$, is needed.

Gauss-Laguerre quadrature. An efficient approach for infinite integrals with an exponentially decaying integrand is classical Gauss-Laguerre quadrature. Laguerre polynomials are orthogonal with respect to e^{-x} on $[0,\infty)$. Hence, using n-point quadrature yields an approximation,

$$
I[f;h_x] \approx Q[f;h_x] := \frac{e^{i\omega g(x)}}{\omega}\sum_{k=1}^{n} w_k f\left(h_x\left(\frac{x_k}{\omega}\right)\right)h_x'\left(\frac{x_k}{\omega}\right) \qquad (16)
$$

As stated in [3], the approximation error by the quadrature rule behaves asymptotically as $\mathcal{O}(\omega^{-2n-1})$ as $\omega \to \infty$. As an illustrative example, let us consider the asymptotic expansion for $U(a,b,z)$ when $|z| \to \infty$, which can be deduced by applying Watson's lemma [11] to (8),

$$U(a,b,z) \sim z^{-a} \sum_{n=0}^{\infty} \frac{(a)_n (a-b+1)_n}{n!(-z)^n}, \quad |\mathrm{ph}\, z| \leq \frac{3}{2}\pi - \delta \tag{17}$$

The error behaves asymptotically as $\mathcal{O}(z^{-n-1})$, as notice by truncating the asymptotic expansion after n terms. Therefore, the asymptotic order of the Gauss-Laguerre quadrature is practically double using the same number of terms. A formula for the error of the n-point quadrature approximation (16) is

$$E = \frac{(n!)^2}{(2n)!} f^{2n}(\zeta), \quad 0 < \zeta < \infty \tag{18}$$

According to this formula and under the general assumption that $a, b \in \mathbb{R} \setminus \mathbb{N}$, f is infinitely differentiable on $[0, \infty)$, we can use the general Leibniz rule for the higher derivatives of a product of m factors to obtain the derivative of order $2n$,

$$((f \cdot g) \cdot h)^{(2n)} = \sum_{j=0}^{2n} \sum_{k=0}^{2n-j} \frac{(2n)! \cdot f^{(j)} g^{(k)} h^{(2n-k-j)}}{j!k!(2n-k-j)!} \tag{19}$$

where

$$f(x) = e^{-\Re(z) i x/\omega}, \quad g(x) = \left(1 + i\frac{x}{\omega}\right)^{b-a-1}, \quad h(x) = \left(i\frac{x}{\omega}\right)^{a-1} \tag{20}$$

and the $2n$ derivatives are given by

$$\sum_{j=0}^{2n} \sum_{k=0}^{2n-j} \frac{(2n)!(-1)^j}{j!k!(2n-k-j)!} \left(\frac{\Re(z) i}{\omega}\right)^j e^{-\Re(z) i x/\omega} \frac{\left(\frac{i}{\omega}\right)^k}{(b-a-1)_{-k}} \left(1 + i\frac{x}{\omega}\right)^{b-a-1-k} \times \frac{\left(\frac{i}{\omega}\right)^{2n-k-j}}{(a-1)_{-2n+k+j}} x^{a-1-2n+k+j} \tag{21}$$

where $(a)_n$ is the Pochhammer symbol or rising factorial. An error bound in terms of a, b and z might be obtained from (21). Ideally, the error bound shall be tight enough without increasing the total computation time excessively. However, as can be seen below, numerical experiments indicate that the number of terms n rarely exceeds 50 for moderate values of the remaining parameters, typically if $|a|, |b| \cdot 10 < |\omega|$, for the case $U(a,b,iz)$ or ${}_1F_1(a,b,iz)$. Finally, for large parameters we apply logarithmic properties to the integrand in order to avoid overflow or underflow.

2.7 Numerical examples

In this section, we compare our algorithm (NSD) with other routines in double precision floating-point arithmetic in terms of accuracy and computation time[3]. Note that just a few packages in double precision allow the evaluation of the confluent hypergeometric function with complex argument. For this study we use Algorithm 707: CONHYP, described in [6, 7] and Zhang and Jin implementation (ZJ) in [12]. Both codes are written in Fortran 90 and were compiled using `gfortran 4.9.3` without optimization flags. We implemented a simple prototype of the described methods using `Python 3.5.1` and the package `SciPy` [5], therefore there is plenty of room for improvement, and is part of ongoing work. Nevertheless, as shown in Table 1, our algorithm clearly outperforms aforementioned codes, being more noticeable as z increases. In order to test the accuracy, we use `mpmath` [4] with 20 digits of precision to compute the relative errors.

${}_1F_1(a,b,z)$	CONHYP	ZJ	NSD	N
$(1, 4, 50i)$	3.96e−13/4.29e−18i	1.50e−15/4.28e−18i	1.15e−16/1.11e−16i	2
$(3, 10, 30 + 100i)$	1.27e−13/1.28e−13i	6.83e−17/1.07e−14i	2.48e−17/1.24e−14i	25
$(15, 20, 200i)$	9.20e−13/9.20e−13i	E	8.43e−16/7.93e−16i	25
$(400, 450, 1000i)$	8.32e−12/1.00e−11i	–	1.37e−12/1.02e−13i	50
$(2, 20, 50 - 2500i)$	1.35e−11/1.35e−11i	7.30e−11/2.10e−09i	4.75e−16/6.41e−16i	20
$(500, 510, 100 - 1000i)$	4.10e−13/3.68e−12i	–	4.71e−13/3.11e−16i	50
$(2, 20, -20000i)$	–	5.79e−10/7.99e−07i	5.92e−16/3.62e−14i	10
$(900, 930, -10^{10}i)$	–	–	6.78e−13/6.77e−13i	20
$(4000, 4200, 50000i)^*$	–	–	6.04e−12/5.99e−12i	80

Table 1. Relative errors for routines computing the confluent hypergeometric function for complex argument. N: number of Gauss-Laguerre quadratures. $(*)$: precision in `mpmath` increased to 30 digits. (E): convergence to incorrect value. $(-)$: overflow.

Table 2 and Figure 1 summarize the testing results and general performance of the algorithm for $U(a, b, z)$. As can be observed, 13-14 digits of precision in real and imaginary part are typically achieved. A similar precision for ${}_1F_1(a; ib; z)$ is expected. In terms of computational time, we compare our implementation in Python with MATLAB R2013a. As shown in Table 3, the MATLAB routine `hypergeom` is significantly slow for large imaginary parameters.

Function	Min	Max	Mean
$U(a, b, iz)$	1.97e−18/2.04e−17i	9.97e−13/2.50e−11i	1.34e−14/6.94e−14i
$U(a, ib, z)$	6.57e−18/6.17e−18i	1.49e−11/8.55e−12i	1.38e−13/1.43e−13i

Table 2. Error statistics for $U(a, b, iz)$ and $U(a, ib, z)$ using $N = 100$ quadratures.

3 Applications

Besides the necessity of accurate and reliable methods for the regime of parameters and argument considered, confluent hypergeometric functions can be

[3] Intel(R) Core(TM) i5-3317U CPU at 1.70GHz.

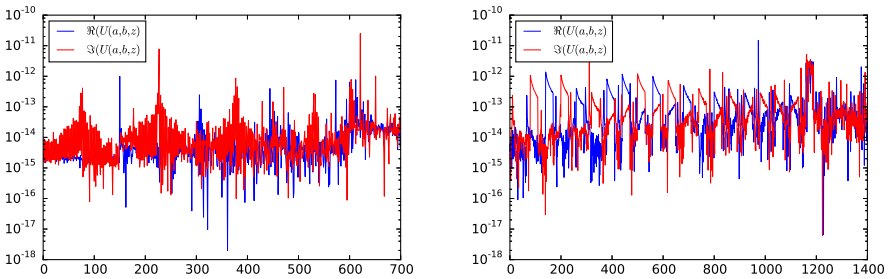

Fig. 1. Relative error in computing $U(a,b,z)$. Error in $U(a,b,iz)$ for $a \in [2,400], b \in [-500,500], z \in [10^3,10^6]$ (left) and $U(a,ib,z)$ for $a \in [10,100], b \in [10^3,10^4], z \in [10,100]$ (right). 700 and 1400 tests, respectively.

${}_1F_1(a;b;z)$	**MATLAB**	**NSD**
$(2,20,-20000i)$	1.509 (0.068)	0.033
$(900,930,-10^{10}i)$	5.594 (0.739)	0.035
$(4000,4200,50000i)$	488.384(18.127)	0.043

Table 3. Comparison in terms of cpu time. MATLAB second evaluation in parenthesis.

encountered in several scientific applications. In this paper, we focus on applications in statistics, more precisely on the evaluation of characteristic functions, which can be defined in terms of confluent hypergeometric functions. Characteristic functions appear in many financial econometric models, for example modelling a beta-distributed loss given default in portfolio credit risk models (see [8, §4.4.2]). Let us consider three statistical distributions:

– Characteristic function of the Beta distribution.

$$\phi_X(t) = {}_1F_1(\alpha;\alpha+\beta;it) \tag{22}$$

where $\alpha,\beta > 0$. Thereby, the regime of parameters holds for the integral representation in (12).

– The standard Arcsine distribution is a special case of the Beta distribution with $\alpha = \beta = 1/2$, therefore we obtain a similar characteristic function, which can be identically computed.

$$\phi_X(t) = {}_1F_1\left(\frac{1}{2};1;it\right) \tag{23}$$

– The characteristic function for the F–distribution is defined in terms of the confluent hypergeometric function of the second kind,

$$\phi_X(t) = \frac{\Gamma((p+q)/2)}{\Gamma(q/2)} U\left(\frac{p}{2},1-\frac{q}{2},-\frac{q}{p}it\right) \tag{24}$$

where $p,q > 0$, are the degrees of freedom. In this case we can use the integral representation in (8).

4 Conclusions

We have presented an efficient algorithm for computing the confluent hypergeometric functions with large imaginary parameter and argument, which emerges as an alternative to asymptotic expansions. The numerical experiments show promising results and fast convergence as the imaginary part increases. Throughout this paper we have been considering real values for the remaining parameters, otherwise the function f becomes oscillatory. The numerical steepest descent method is not insensitive to oscillations in f, although in some cases this can be treated by applying other transformations. In cases where that is not possible, other methods have to be considered. Finally, a suitable integral representation for $|\Im(a)| \to \infty$ carry more complications and is part of future work.

References

1. NIST Digital Library of Mathematical Functions. `http://dlmf.nist.gov/`, Release 1.0.10 of 2015-08-07. Online companion to: F. W. J. Olver et al (editors) *NIST Handbook of Mathematical Functions*. Cambridge University Press, NY, (2010).
2. M. Galassi et al. *GNU Scientific Library Reference Manual (3rd Ed.)*, ISBN 0954612078. `http://www.gnu.org/software/gsl/`
3. D. Huybrechs and S. Vandewalle. *On the evaluation of highly oscillatory integrals by analytic continuation*. SIAM J. Numer. Anal., 44(3), pp. 1026-1048, (2006).
4. F. Johansson and others, *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.19)*, (2013). `http://mpmath.org/`.
5. E. Jones, E. Oliphant, P. Peterson, et al. *SciPy: Open Source Scientific Tools for Python*, (2001). `http://www.scipy.org/`.
6. M. Nardin, W. F. Perger, and A. Bhalla. *Algorithm 707. CONHYP : A numerical evaluator of the confluent hypergeometric function for complex arguments of large magnitudes*. ACM Trans. Math. Software 18, 345–349, (1992).
7. M. Nardin, W. F. Perger, and A. Bhalla. *Numerical evaluation of the confluent hypergeometric function for complex arguments of large magnitudes*. J. Comput. Appl. Math. 39, 193–200, (1992).
8. G. Navas-Palencia. *Portfolio Credit Risk: Models and Numerical Methods*. MSc in Statistics and Operations Research Dissertation, Universitat Politècnica de Catalunya, (2016). `http://upcommons.upc.edu/bitstream/handle/2117/82265/memoria.pdf`.
9. J. W. Pearson *Computation of Hypergeometric Functions*. MSc in Mathematical Modelling and Scientific Computing Dissertation, University of Oxford, (2009).`http://people.maths.ox.ac.uk/~porterm/research/pearson_final.pdf`.
10. J. W. Pearson, S. Olver, M. A. Porter. *Numerical Methods for the Computation of the Confluent and Gauss Hypergeometric Functions*, arXiv:1407.7786, (2015).
11. G. N. Watson. *The Harmonic Functions Associated with the Parabolic Cylinder*. Proc. of the London Mathematical Society, 2, pp. 116–148, (1918).
12. S. Zhang and J. Jin. *Computation of special functions*. John Wiley & Sons Inc., New York, (1996).