Numerical Algorithms (2023) 94:669–679
https://doi.org/10.1007/s11075-023-01515-y

RESEARCH

Computation of the confluent hypergeometric function $U(a, b, x)$ and its derivative for positive arguments

Amparo Gil[1] · Diego Ruiz-Antolín[1] · Javier Segura[2] · Nico M. Temme[3,4]

Received: 6 November 2022 / Accepted: 7 February 2023 / Published online: 22 March 2023

Abstract
An algorithm and a MATLAB implementation for computing the Kummer function $U(a, b, x)$ and its derivative is given in this paper. The algorithm is efficient and accurate. Numerical tests show that the MATLAB algorithm allows the computation of the function with $\sim 10^{-14}$ relative accuracy in the parameter region $(a, b, x) \in (0, 500) \times (0, 500) \times (0, 1000)$ in double-precision floating point arithmetic.

Keywords Kummer function · Numerical computation · MATLAB software

Mathematics Subject Classification (2010) 33B15 · 33C15 · 65D20

1 Introduction

Many applications in physics and engineering involve the computation of confluent hypergeometric functions or some of their particular cases (Airy and Bessel functions, Laguerre polynomials, parabolic cylinder functions etc.) See [1, 2, 8, 12] or [11, §13.28] for some examples of applications in physics. As an additional example, in recent work we have obtained expansions for the relativistic Fermi-Dirac integral and its derivatives (of great importance in stellar astrophysics) where confluent hypergeometric functions play a key role [3, 5].

✉ Amparo Gil
amparo.gil@unican.es

1 Departamento de Matemática Aplicada y CC. de la Computación, Universidad de Cantabria, 39005, Santander, Spain

2 Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, 39005, Santander, Spain

3 IAA, 1825 BD 25, Alkmaar, The Netherlands

4 Centrum Wiskunde & Informatica (CWI), Science Park 123, 1098 XG, Amsterdam, The Netherlands

In spite of their importance, few algorithms are available in double-precision floating point arithmetic for the computation of any of the standard solutions of the Kummer's equation in the case of real or complex parameters. For the function $M(a, b, x)$, some algorithms can be found in the literature (see [9, 10]); however, for the second solution $U(a, b, x)$, no double-precision algorithms with good uniform accuracy seem to be available, as far as we know. For example, the SLATEC library [16] includes the Fortran 77 function `dchu.f` which implements the confluent Kummer function $U(a, b, x)$ for real parameters a, b and argument x. However, one has to be careful when using `dchu.f` to compute $U(a, b, x)$ when a or b are large in comparison to x because a significant loss of accuracy occurs in the computations. Another example of a double-precision implementation for the Kummer $U(a, b, x)$ function which is not free of accuracy problems is the function `hyperu` included in the SCIPY [7] module scipy.special. Having accurate efficient double-precision implementations is of interest in, for example, applications where a large number of function evaluations are needed (such as in the condensed matter or astrophysical calculations). On the other hand, extended precision implementations are given in Mathematica, Maple or Arb [6].

In this paper, we describe an efficient and accurate algorithm implemented in MATLAB for computing the Kummer $U(a, b, x)$ function and its derivative. The function $U(a, b, x)$ satisfies the following integral representation

$$U(a, b, x) = \frac{1}{\Gamma(a)} \int_0^{+\infty} e^{-zt} t^{a-1} (1+t)^{b-a-1}\, dt, \tag{1.1}$$

valid for $\Re a > 0$, $b \in \mathbb{C}$ and $x > 0$.

In the algorithm, we consider real positive values of the function arguments a, b, x. The algorithm uses series for small values of the parameters, asymptotic expansions for large values of the parameters and a double-precision implementation of the algorithm described in [14] in the rest of cases. Our numerical tests show that the resulting algorithm is both accurate and efficient.

2 Computation for small values of the parameters

For $0 < a,\ b < \frac{1}{2}$ and $0 < x < 1$, we use the series given in [4] for the Kummer function $U(a, b, z)$ and its derivative

$$\begin{aligned} U(a, b, z) &= \frac{\Gamma(1-b)}{\Gamma(a-b+1)} + \frac{\pi b z}{\sin(\pi b)\Gamma(a)\Gamma(a-b+1)} \sum_{m=0}^{\infty} w_m \frac{z^m}{m!}, \\ U'(a, b, z) &= \frac{\pi b}{\sin(\pi b)\Gamma(a)\Gamma(a-b+1)} \sum_{m=0}^{\infty} w_m^d \frac{z^m}{m!}, \end{aligned} \tag{2.1}$$

where the coefficients w_m^d can be written in terms of the w_m coefficients

$$w_m^d = (m+1) w_m + z w_m'. \tag{2.2}$$

The coefficients w_m satisfy

$$w_m = \frac{u_m}{v_m}, \tag{2.3}$$

where

$$\begin{aligned} u_m &= (A_m - B_m)/b, \\ v_m &= \Gamma(m+2)\Gamma(b+m+1)\Gamma(2-b+m). \end{aligned} \tag{2.4}$$

with $A_m = \Gamma(m+1)\Gamma(2-b+m)\Gamma(a+m+1)$ and $B_m = z^{-b}\Gamma(a-b+1+m)\Gamma(b+1+m)\Gamma(m+2)$.

To compute w_m for $m = 1, 2, ...$ using (2.4), it is convenient to obtain a stable recursion for u_m when b is small. It is possible to verify that

$$\begin{aligned} u_{m+1} &= (a_m A_m - b_m B_m)/b = a_m u_m + d_m B_m, \\ d_m &= (a_m - b_m)/b, \end{aligned} \tag{2.5}$$

where

$$\begin{aligned} a_m &= c_m - \left(m^2 + (a+2)m + a + 1\right) b, \\ b_m &= c_m + (m+2)(a-b)b, \\ c_m &= (m+1)(m+2)(m+a+1), \\ d_m &= -\left(m^2 + 2m(a+1) + 3a + 1\right) + (m+2)b. \end{aligned} \tag{2.6}$$

The starting u_0 value of the recursion given in (2.5) is computed as $u_0 = v_0 w_0$, where $v_0 = \frac{(1-b)\pi b}{\sin(\pi b)}$ and for calculating w_0, we use

$$w_0 = \frac{g_0}{r g_4}\left(1 + r g_2 - g_3 - r g_0 g_5 g_1 + \tilde{z}_b g_6\right), \tag{2.7}$$

where $r = b - 1$, $\tilde{z}_b = -e^q \log z \frac{\sinh q}{q}$ with $q = -\frac{b}{2}\log z$ and

$$\begin{aligned} &g_0 = \Gamma(a+1),\ \ g_1 = G(a,-b),\ \ g_2 = G(0,b), \\ &g_3 = G(0,-b),\ \ g_4 = 1 - b g_0 g_1,\ \ g_5 = 1 + b g_2,\ \ g_6 = 1 - b g_3. \end{aligned} \tag{2.8}$$

with

$$G(a,b) = \frac{1}{b}\left(\frac{1}{\Gamma(a+1+b)} - \frac{1}{\Gamma(a+1)}\right). \tag{2.9}$$

For computing $G(a,b)$, we use the series

$$G(a,b) = \sum_{k=2}^{\infty} c(k)d(k), \quad d(k) = \frac{1}{b}\left((a+b)^{k-1} - a^{k-1}\right), \tag{2.10}$$

where the first three $c(k)$ coefficients are $c(2) = 0.57721566490153286$, $c(3) = -0.65587807152025388$ and $c(4) = -0.04200263503409523$. More $c(k)$ coefficients are given in Table 1 of [4].

3 Algorithm based on the use of recurrences

One of the key ingredients in the algorithm described in [14] to evaluate $U(a,b,x)$ is the use of three-term recurrence relations satisfied by the function with respect to the parameters a and b.

With respect to the a parameter, we have

$$U(a-1,b,x)+(b-2a-x)U(a,b,x)+a(1+a-b)U(a+1,b,x)=0, \quad (3.1)$$

and with respect to the b parameter,

$$(b-a-1)U(a,b-1,x)+(1-b-x)U(a,b,x)+xU(a,b+1,x)=0. \quad (3.2)$$

As discussed for example in [13], the Kummer function $U(a,b,x)$ is the minimal solution of the three-term recurrence relation (3.1). Therefore, the forward computation of the recursion is ill-conditioned and backward recursion should be applied. On the contrary, $U(a,b,x)$ is a dominant solution of (3.2) and forward recursion is possible.

Briefly, the algorithm described in [14] uses a sophisticated version of the backward a-recursion to compute $U(a,b,x)$ for $0<b<1$. This recursion is computed using starting values given by asymptotic expansions in terms of Bessel functions (for small x) or Miller algorithm's when x is not small. For $b>1$, backward recursion (3.1) is first applied to a value $\tilde{b}=b-[b]$ in the interval $[0,\ 1]$; then, the forward b-recursion (3.2) is applied. Technical details can be seen in [14]. This algorithm works very well in extended precision; however, in double-precision floating point arithmetic, some loss of accuracy could appear when using the recursions for large values of the parameter a. This is the reason why it is better to use an alternative method of computation for large values of the parameters, such as the expansions described in the next section.

4 Asymptotic expansions for large values of the parameters

We consider the recent expansion given in [15]

$$U(a,b+1,z)\sim e^{z\mathcal{A}}z^{-a}p_0(\mu)\sum_{n=0}^{\infty}(-1)^n\frac{\widetilde{p}_n(\mu)}{z^n},\ \ z\to\infty, \quad (4.1)$$

where

$$\begin{aligned} p_0(\mu) &= \frac{1-\mu\tau}{\sqrt{\beta\mu\tau^2-2\mu\tau+1}}, \qquad \mathcal{A}=\mu\,(\tau-\ln\tau-1)-\alpha\ln(1-\mu\tau), \\ t_0 &= \frac{2\mu}{\beta+1+\sqrt{(\beta+1)^2-4\mu}}, \end{aligned} \quad (4.2)$$

with $\tau=t_0/\mu$, $\beta=b/z$ and $\mu=(b-a)/z$.

The first coefficients of the expansion (4.1) are

$$
\begin{aligned}
\widetilde{p}_0(\mu) &= 1,\\
\widetilde{p}_1(\mu) &= \frac{\mu\tau^2(1-\tau)\left(\mu^2\tau^4-\mu\tau^3+8\mu\tau^2-9\tau+1\right)}{12(\mu\tau^2-1)^3(\mu\tau-1)},\\
\widetilde{p}_2(\mu) &= \frac{\mu\tau^4(\tau-1)}{288(\mu\tau^2-1)^6(\mu\tau-1)^2}\Big(\mu^5\tau^9-\mu^5\tau^8-2\mu^4\tau^8+18\mu^4\tau^7-304\mu^4\tau^6\\
&\quad +\mu^3\tau^7-35\mu^3\tau^6+1396\mu^3\tau^5-930\mu^3\tau^4+18\mu^2\tau^5-1892\mu^2\tau^4\\
&\quad +2610\mu^2\tau^3-304\mu^2\tau^2+801\mu\tau^3-2403\mu\tau^2+595\mu\tau-\mu\\
&\quad +720\tau-288\Big).
\end{aligned}
\tag{4.3}
$$

4.1 Expansion for $U'(a,b+1,z)$

For obtaining the asymptotic expansion of $U'(a,b+1,z)$ we consider the relation

$$U'(a,b,z)=-aU(a+1,b+1,z), \tag{4.4}$$

and use the integral (see (4.3) of [15])

$$U'(a,b+1,z)=-a\frac{z^{-b-1}\Gamma(\lambda+1)}{2\pi i}\int_{-\infty}^{(0+)}e^{zt}t^{-\lambda-1}(1-t)^{-a-1}\,dt, \tag{4.5}$$

where $\lambda=b-a$. We write this in the form

$$U'(a,b+1,z)=-a\frac{z^{-b-1}\Gamma(\lambda+1)}{2\pi i}\int_{-\infty}^{(0+)}e^{z\phi(t)}\,\frac{dt}{t(1-t)}, \tag{4.6}$$

where, with $\mu=\lambda/z$,

$$\phi(t)=t-\alpha\ln(1-t)-\mu\ln t,\quad \phi'(t)=-\frac{t^2-(\beta+1)t+\mu}{t(1-t)}. \tag{4.7}$$

The function $\phi(t)$ is the same as used in [15, §3], and the saddle point t_0 and the coefficients are the same as used in that section, up to a factor $(-1)^n$. We obtain the expansion

$$U'(a,b+1,z)\sim -ae^{z\mathcal{A}}z^{-a-1}f_0(\mu)\sum_{n=0}^{\infty}(-1)^n\frac{\widetilde{f}_n(\mu)}{z^n},\quad z\to\infty, \tag{4.8}$$

where

$$\mathcal{A}=\phi(t_0)-\mu+\mu\ln\mu,\qquad \widetilde{f}_n(\mu)=\frac{f_n(\mu)}{f_0(\mu)}, \tag{4.9}$$

and

$$f_0(\mu)=\sqrt{\frac{\mu}{\beta t_0^2-2\mu t_0+\mu}}=\sqrt{\frac{\mu}{(1-t_0)\left(\mu-t_0^2\right)}}. \tag{4.10}$$

The saddle point t_0 is given by

$$t_0=\tfrac12(\beta+1)-\tfrac12\sqrt{(\beta+1)^2-4\mu}\ =\frac{2\mu}{\beta+1+\sqrt{(\beta+1)^2-4\mu}}, \tag{4.11}$$

with expansion

$$t_0 = \frac{\mu}{\beta+1} + \frac{\mu^2}{(\beta+1)^3} + \mathcal{O}\left(\mu^3\right), \quad \mu \to 0. \tag{4.12}$$

The first few $\widetilde{f}_n$ coefficients are $\widetilde{f}_0 = 1$ and

$$\begin{aligned} \widetilde{f}_1 &= -\frac{1}{12}\mu\tau^2 \frac{\left(\mu^2\tau^5 - 13\mu^2\tau^4 - \mu\tau^4 + 21\mu\tau^3 + 4\mu\tau^2 - 9\tau^2 - 2\tau - 1\right)}{(\mu\tau^2-1)^3(\mu\tau-1)}, \\ \widetilde{f}_2 &= \frac{1}{288}\frac{\mu\tau^4}{(\mu\tau^2-1)^6(\mu\tau-1)^2}\left(\mu^5\tau^{10} - 26\mu^5\tau^9 + 313\mu^5\tau^8 - 2\mu^4\tau^9\right. \\ &\quad +68\mu^4\tau^8 - 1690\mu^4\tau^7 + mu^3\tau^8 + 1048\mu^4\tau^6 - 60\mu^3\tau^7 \\ &\quad +3255\mu^3\tau^6 - 3958\mu^3\tau^5 + 18\mu^2\tau^6 + 186\mu^3\tau^4 - 2678\mu^2\tau^5 \\ &\quad +5462\mu^2\tau^4 - 490\mu^2\tau^3 + 801\mu\tau^4 - 8\mu^2\tau^2 - 3276\mu\tau^3 \\ &\quad \left.+454\mu\tau^2 + 4\mu\tau + 720\tau^2 + \mu - 144\tau\right). \end{aligned} \tag{4.13}$$

5 Numerical testing and algorithm

The set of MATLAB functions implementing the methods used are:

1. `Uabxsmall(a,b,x)`. Implementation of the method given in Section 2.
2. `Uabxrec(a,b,x)`. Implementation of the method given in Section 3.
3. `Uabxexpan(a,b,x)`. Implementation of the method given in Section 4.

For testing the computation of $U(a,b,x)$ and $U'(a,b,x)$ using the different methods, we consider the recurrence relations

$$\begin{aligned} \frac{aU(a+1,b,x) + U(a,b-1,x)}{U(a,b,x)} &= 1, \\ \frac{(a-1)U'(a,b-1,x) + U'(a-1,b-2,x)}{U'(a-1,b-1,x)} &= 1. \end{aligned} \tag{5.1}$$

Also, for testing the method in Section 2, we consider the following alternative relation

$$\frac{(a-b+x)U(a,b,x) - xU'(a,b,x)}{U(a-1,b,x)} = 1. \tag{5.2}$$

When using this relation, values $a \sim b$ should not be included to avoid the loss of significant digits by cancellation. We have randomly generated 10^6 values in the parameter region $(a, b, x) \in (0, 0.5) \times (0, 0.5) \times (0, 1)$ for testing (5.2) using `Uabxsmall(a,b,x)`. The results obtained show that for more of the $\sim$ 98% of the points, the obtained accuracy was better than 10^{-14}. Also, for the same parameter region, this method is more than a 20% faster than the method implemented in `Uabxrec(a,b,x)`.

In Figs. 1 and 2, we show the accuracy obtained when using the expansion (4.1) and (4.8) implemented in the MATLAB function `Uabxexpan(a,b,x)` to approximate the values of $U(a,b,x)$ and $U'(a,b,x)$. We take $n = 5$ terms in both

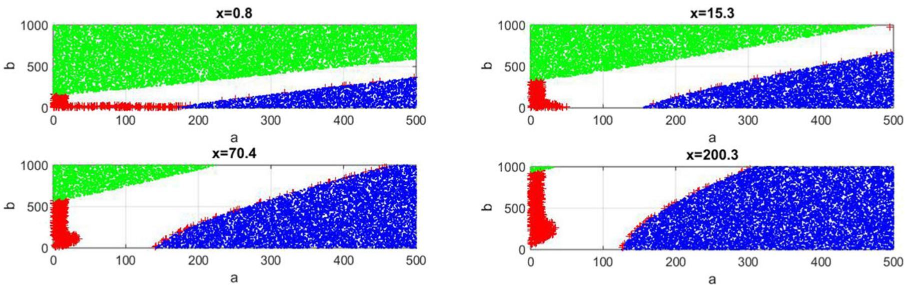

Fig. 1 Test for the expansion (4.1). The blue (green) points indicate the region where the function values are close or under (above) the underflow (overflow) limit in double-precision floating point arithmetics. The red crosses indicate points where the error obtained when testing the first of the recurrence relations in (5.1) is greater than 10^{-12}

expansions. We check the recurrence relations (5.1) for few fixed values of x in the region $(a, b) = (0, 500) \times (0, 1000)$ with 20,000 random points. The red crosses indicate points where the error obtained when testing the recurrence relations (5.1) is greater than 10^{-12}. The blue points correspond to function values very close or under the underflow limit in double precision. The green points correspond to function values very close or above the overflow limit in double precision. As can be seen, the expansions are not accurate, in general for a small or when x is small, for all values of a when the b-values are not large. As we have shown, the function `Uabxsmall(a,b,x)` works very well $(a, b, x) \in (0, 0.5) \times (0, 0.5) \times (0, 1)$. For other small parameter values, the algorithm implemented in `Uabxrec(a,b,x)` can be used. For x small, we have made a slight modification in the original algorithm and we use a scaled version of the a-recursion in order to avoid the loss of significant digits for large values of a when computed in double-precision floating point arithmetics.

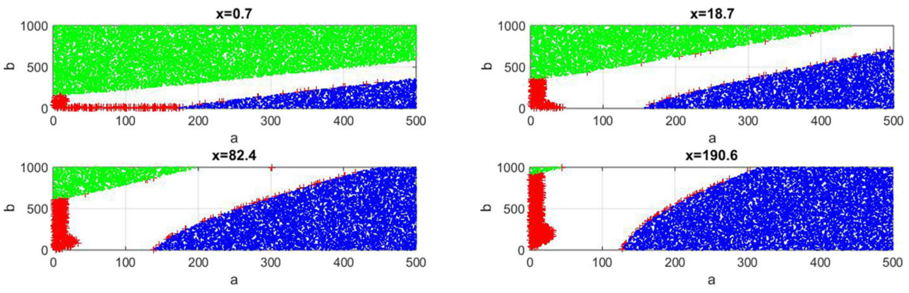

Fig. 2 Test for the expansion (4.8). The blue (green) points indicate the region where the function values are close or under (above) the underflow (overflow) limit in double-precision floating point arithmetics. The red crosses indicate points where the error obtained when testing the second of the recurrence relations in (5.1) is greater than 10^{-12}

Taking into account the results obtained in the different tests of the methods, the resulting computational scheme is given in Algorithm 1. This algorithm is implemented in the MATLAB function `Uabxcomp`.[1] The front factors in the expansions (4.1) and (4.8) are used in the full algorithm as an estimation of the function value to avoid possible overflow/underflow problems.

For testing the accuracy of the full algorithm for computing $U(a, b, x)$, we have considered 10^8 random values in the parameter region $(a, b, x) \in (0, 500) \times (0, 500) \times (0, 1000)$. Of all the randomly generated points, 2.64×10^7 were points not in the underflow or overflow regions. The results show that at approximately 54% of these points, the accuracy is better than 10^{-14}, while at around 43% of the points the accuracy is between 10^{-14} and 10^{-13}. The maximum error obtained was $\sim 10^{-11}$, at a point for which the function value was close to the underflow limit. See also Fig. 3. Similar results were obtained for the derivative.

```
Require: $x > 0, a > 0, b > 0$
Ensure: $U(a, b, x)$, $U'(a, b, x)$ and $iov$ (overflow flag)
  Set huge= $10^{+308}$ and dwarf= $10^{-310}$.
  if $a < 0.5$ & $b < 0.5$ & $x < 1$ then
      Use the function Uabxsmall(a,b,x)
  else if $x < 1.4$ & $b < 30$ then
      Use the function Uabxrec(a,b,x)
  else
      Compute $F = e^{x\mathcal{A}}x^{-a}$ in Section 4 to decide whether ($iov = 1$) or not ($iov =$
  0) the function value surpass the overflow limit.
      if $iov = 1$ then
          Return $U(a, b, x) =$ huge and $U'(a, b, x) =$ huge.
      else
          Compute $F = e^{x\mathcal{A}}x^{-a}$ for $b = 10^{-10}$.
          Set logical $p = (a > 50)\&(b > 50)$.
          if $F >$ dwarf & not $p$ then
              Use the function Uabxrec(a,b,x)
          else
              Use the function Uabxexpan(a,b,x)
          end if
      end if
  end if
```

Algorithm 1 Computation of the Kummer function $U(a, b, x)$ and its derivative.

As an additional test, we have compared our function `Uabxcomp` against the intrinsic MATLAB function to evaluate the Kummer $U(a.b, x)$ function which is called `kummerU`. This function, included in the symbolic math toolbox, provides

[1]This function is available at http://personales.unican.es/gila/KummerUeval.zip

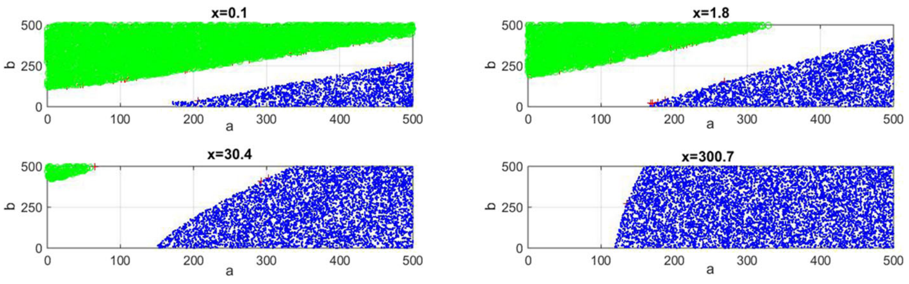

Fig. 3 Test for the combined algorithm. The blue (green) points indicate the region where the function values are close or under (above) the underflow (overflow) limit in double-precision floating point arithmetics. The red crosses indicate points where the error obtained when testing the first of the recurrence relations in (5.1) is greater than 10^{-12}

also floating-point results for numeric arguments. When checking the accuracy obtained in comparison to our function, the tests reveal that for some parameters, the computations obtained with `kummerU` are not accurate. For example, in Fig. 4, we show that the calculation of $U(a, b, x)$ using `kummerU` in MATLAB R2022b gives a negative number for certain values of the parameters ($U(a, b, x)$ is non-negative for $x > 0$, $a \geq 0$, $b \in \Re$). With respect to computational efficiency, the computations with `kummerU` take much more time than those obtained with our algorithm, but this is expected since `kummerU` is part of a symbolic package. As an example, the computation of 500 function values with random arguments in the parameter region $(a, b, x) \in (0, 500) \times (0, 500) \times (0, 1000)$ took $9.4\,10^{-2}\,s$ using `Uabxcomp` and $199\,s$ using `kummerU`. The results have been obtained using MATLAB R2022b in a PC with 8 GB RAM and Intel Core i5-4310U processor.

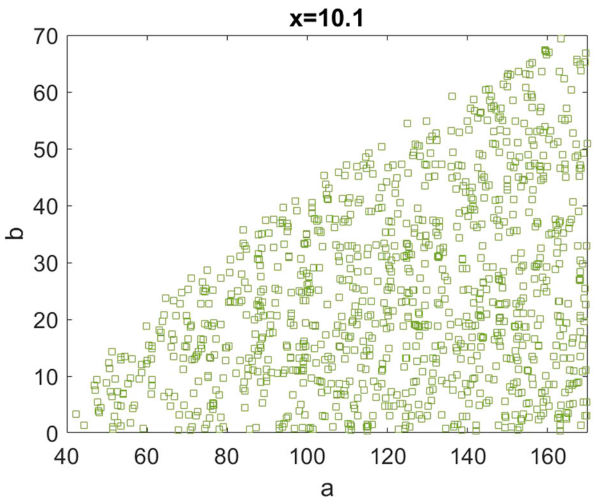

Fig. 4 Example of points in the (a, b)-plane where the calculation of $U(a, b, x)$ using the intrinsic function in MATLAB R2022b gives a negative number

Acknowledgements The authors thank the reviewer for useful comments and suggestions. NMT thanks CWI Amsterdam for scientific support.

Author Contributions The authors contributed equally to the paper.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. The authors acknowledge financial support from project PGC2018-098279-B-I00 funded by MCIN/ AEI /10.13039/501100011033/ FEDER “Una manera de hacer Europa” and PID2021-127252NB-I00 funded by MCIN/AEI/10.13039/501100011033/ FEDER, UE.

Declarations

Competing interests The authors declare no competing interests.

References

1. Felsen, L.B., Marcuvitz, N.: Radiation and scattering of waves. Wiley-IEEE Press (1994)
2. Gasaneo, G., Ancarani, L.U.: Two-body Coulomb problems with sources. Phys. Rev. A **82**, 042706 (2010)
3. Gil, A., Odrzywołek, A., Segura, J., Temme, N.M.: Evaluation of the generalized Fermi-Dirac integral and its derivatives for moderate/large values of the parameters. Comput. Phys. Commun. **238**, 108563 (2023)
4. Gil, A., Segura, J., Temme, N.M.: Computing the Kummer function $U(a, b, z)$ for small values of the arguments. Appl. Math. Comput. **271**, 532–539 (2015)
5. Gil, A., Segura, J., Temme, N.M.: Complete asymptotic expansions for the relativistic Fermi-Dirac integral. Appl. Math. Comput. **412**, 126618 (2022)
6. Johansson, F.: Computing hypergeometric functions rigorously. ACM Trans. Math. Softw. **45**(3), Art. 30, 26 (2019)
7. Jones, E., Oliphant, T., Peterson, P.: SciPy: Open Source Scientific Tools for Python (2001)
8. Mathews, W.N. Jr, Esrick, M.A., Teoh, Z.Y., Freericks, J.K.: A physicist’s guide to the solution of Kummer’s equation and confluent hypergeometric functions. Condensed Matter Phys. **25**, 33203:,1–23 (2022)
9. Muller, K.E.: Computing the confluent hypergeometric function, $M(a, b, x)$. Numer. Math. **90**(1), 179–196 (2001)
10. Nardin, M., Perger, W.F., Bhalla, A.: Algorithm 707: CONHYP: A numerical evaluator of the confluent hypergeometric function for complex arguments of large magnitudes. ACM Trans. Math. Soft. **18**, 345–349 (1992)
11. Olde Daalhuis, A.B.: Chapter 13, Confluent hypergeometric functions. In: NIST Handbook of Mathematical Functions, pp. 321–349. Cambridge University Press, Cambridge (2010a). http://dlmf.nist.gov/13
12. Schweizer, W.: Confluent hypergeometric function. In: Special Functions in Physics with MATLAB, pp. 91–99. Springer International Publishing (2021)
13. Segura, J., Temme, N.M.: Numerically satisfactory solutions of Kummer recurrence relations. Numer. Math. **111**, 109–119 (2008)

14. Temme, N.M.: The numerical computation of the confluent hypergeometric function $U(a, b, z)$. Numer. Math. **41**(1), 63–82 (1983)
15. Temme, N.M., Veling, E.J.M.: Asymptotic expansions of Kummer hypergeometric functions with three asymptotic parameters a, b and z. Indag. Math. **33**(6), 121–1235 (2022)
16. Vandevender, W.H., Haskell, K.H.: The SLATEC mathematical subprogram library. ACM SIGNUM Newsl. **17**, 16–21 (1982)