A MATLAB™ program for the computation of the confluent hypergeometric function $\Phi_2$


Abstract

We here present a sample MATLAB™ program for the numerical evaluation of the confluent hypergeometric function $\Phi_2$. This program is based on the calculation of the inverse Laplace transform using the algorithm suggested by Simon and Alouini in their reference textbook [1].

I. THE MULTIVARIATE $\Phi_2$ FUNCTION

The following confluent form of the generalized Lauricella series is defined in [2, eq. 7.2, pp. 446]

$$\Phi_2^{(n)} (b_1, \ldots, b_n; c; x_1, \ldots, x_n) \triangleq \sum_{m_1,\ldots,m_n} \frac{(b_1)_{m_1} \cdots (b_n)_{m_n}}{(c)_{m_1+\cdots+m_n} m_1! \cdots m_n!} x_1^{m_1} \cdots x_n^{m_n}$$  \hspace{1cm} (1)

where $(\cdot)_m$ denotes the Pochhammer symbol. This function, also regarded as confluent hypergeometric function of $n$ variables [3], makes appearances in numerous problems in communication theory [4–8], either in a bivariate form ($n = 2$) or in a multivariate fashion.

Because it is defined as an $n$-fold infinite summation, its numerical evaluation poses some challenges from a computational point of view. However, the Laplace transform of the $\Phi_2$ function has a comparatively simpler form, in terms of a finite productory of elementary functions. Specifically, in [9, 4.24.5], the following Laplace transform pair is listed:

$$\mathcal{L} \left\{ t^{c-1} \Phi_2^{(n)} (b_1, \ldots, b_n; c; x_1 t, \ldots, x_n t) ; t, s \right\} = \frac{\Gamma(c)}{s^c} \left( 1 - \frac{x_1}{s} \right)^{-b_1} \cdots \left( 1 - \frac{x_n}{s} \right)^{-b_n}$$  \hspace{1cm} (2)

which is valid for $\Re(c) > 0$, $\Re(s) > 0$, $b_i \in \mathbb{R}$, $i = 1 \ldots n$.

Thus, we have that

$$\Phi_2^{(n)} (b_1, \ldots, b_n; c; x_1, \ldots, x_n) = t^c \Phi_2^{(n)} (b_1, \ldots, b_n; c; x_1, \ldots, x_n t) \big|_{t=1} = \mathcal{L}^{-1} \left\{ \frac{\Gamma(c)}{s^c} \left( 1 - \frac{x_1}{s} \right)^{-b_1} \cdots \left( 1 - \frac{x_n}{s} \right)^{-b_n} ; s, t \right\}$$  \hspace{1cm} (3)
Therefore, the $\Phi_2$ function can be evaluated by means of an inverse Laplace transform. Due to numerous requests, we here provide a MATLAB™ sample code for the evaluation of the $\Phi_2$ function. This code implements an Euler summation-based technique described in detail in the appendix 9B of the reference textbook by Simon and Alouini [1], inspired in [10].

II. MATLAB™ CODE

```matlab
function y=Phi2(bvector,c,xvector,N)

% bvector : [b1 ... bn]
% c : scalar parameter Re(c)>0
% xvector : [x1 ... xn]
% N : Truncation of infinite summation

% Heuristically adjusted so that the discretization error term
% abs{E(A)}<exp(-A)
A=15;

% Inverse Laplace Transform
K=exp(A/2);
alphainv=[0.5, ones(1,N-1)];
n=0:N-1;
y1=(-1).^n.*alphainv.*real(LaplacePhi2((A+2*pi*1i*n)/2,c,xvector,bvector));
y=K.*sum(y1);

function y=LaplacePhi2(s,c,x,b)
P=ones(1,length(s));
for k=1:length(b)
    P=P.*((1-x(k)*(s.^(-1))).^(-b(k)));
end;
y=gamma(c)*(s.^(-c)).*P;
```
REFERENCES


