Extending error function and related functions to complex arguments

Guillermo Navas–Palencia*

December 20, 2016

Abstract

In this short communication several extensions of the Faddeeva function are implemented using functions currently available in the NAG Library. These extensions allow the evaluation of error and related functions with complex arguments. Finally, two relevant applications employing these extensions are presented.

1 Introduction

Chapter S of the NAG Library[1] titled "Approximations of Special Functions", contains routines for evaluating Bessel functions, error functions, elliptic integrals, Fresnel integrals among others. In this report we extend the error functions available in the NAG Library for real argument to complex argument. These extensions will allow us to evaluate more difficult special functions related to the error functions, see [4].

The error functions appear in several mathematical and physics applications; in mathematics the complementary error function is extensively used in the development of uniform asymptotic expansions for evaluating more involved special functions such as incomplete gamma functions and appear in several exponentially-improved asymptotic expansions, for example in the generalized exponential integral. Furthermore, it plays a key role in providing a smooth interpretation of the Stokes phenomenon. Applications in statistics and probability theory are well known, for example the use of the error function in the normal distribution function and in the asymptotic of arbitrary probability density functions. The Faddeeva function, Fresnel integrals, and other related functions are present in several physics applications, from analysis of the diffraction of light to atomic physics and astrophysics. Voigt functions appear in the analysis of light absorption, plasma diagnostics, neutron diffraction, laser spectroscopy among others. Other closely related functions with important applications in physics will be explored in Section 3.

This short communication is organized as follows: In Section 2 we will present the extensions of the Faddeeva function. In Section 3 we will explore several applications of some related functions. Finally, Section 4 will include the necessary code to run the examples using the NAG Toolbox for *MATLAB*®.

2 Faddeeva extensions

Faddeeva function: All the extensions in this Section are based on the NAG routine `nag_specfun_erfc_complex (s15dd)`, which computes the function $w(z) = e^{-z^2}\operatorname{erfc}(-iz)$, the so-called *Faddeeva function* or *plasma dispersion function*.

*Numerical Algorithms Group (NAG) Ltd., Oxford, UK. Email: guillermo.navas@nag.co.uk

[1] See `http://www.nag.co.uk/numeric/cl/nagdoc_cl25/html/s/sconts.html`.

Complementary error function, erfc(z): The complementary error function can be expressed in terms of the Faddeeva function by means of the following connection formula

$$\mathrm{erfc}(z) = e^{-z^2} w(iz), \tag{1}$$

where the values of $w(z)$ in the lower half of the complex plane can be obtained from values in the upper half using the functional relation

$$w(-z) = 2e^{-z^2} - w(z). \tag{2}$$

For real argument z we use the routine `nag_specfun_erfc_real (s15ad)`.

Error function, erf(z): The error function can be defined in terms of the Faddeeva function as follows

$$\mathrm{erf}(z) = 1 - e^{-z^2} w(iz) \tag{3}$$

or simply using the well-known connection formula,

$$\mathrm{erf}(z) = 1 - \mathrm{erfc}(z). \tag{4}$$

For real argument z we use the routine `nag_specfun_erf_real (s15ae)`.

Imaginary error function, erfi(z): The imaginary function $\mathrm{erfi}(z)$ is an entire function defined by

$$\mathrm{erfi}(z) = -i\mathrm{erf}(iz). \tag{5}$$

This function is not currently implemented in the NAG Library, however it can be defined in terms of the Faddeeva function as follows

$$w(z) = e^{-z^2}(1 + i\mathrm{erfi}(z)). \tag{6}$$

Dawson's integral: Dawson's integral, also called Dawson's function, is the entire function given by the integral

$$F(z) = e^{-z^2} \int_0^z e^{t^2}\, dt = \frac{\sqrt{\pi}}{2} e^{-z^2} \mathrm{erfi}(z). \tag{7}$$

It can be defined in terms of the Faddeeva function using Equation (6)

$$F(z) = \frac{i\sqrt{\pi}}{2}(e^{-z^2} - w(z)). \tag{8}$$

For the real case we use the routine `nag_specfun_dawson (s15af)`.

Scaled complement of the error function, erfcx(z): The scaled complement of the error function may be used to replace an expression of the form $e^{z^2}\mathrm{erfc}(z)$, in order to handle cases prone to underflow or overflow errors. It is directly related to the Faddeeva function

$$\mathrm{erfcx}(z) = w(iz). \tag{9}$$

For the real case, we use `nag_specfun_erfcx_real (s15ag)`.

Fresnel integrals: Fresnel integrals $S(z)$ and $C(z)$ can be defined in terms of the imaginary error function $\mathrm{erfi}(z)$. For the real case, two routines are available in the NAG Library, see `nag_specfun_fresnel_s (s20ac)` and `nag_specfun_fresnel_c (s20ad)`, respectively.

Fresnel integral $S(z)$

$$S(z) = \sqrt{\frac{\pi}{2}}\frac{1+i}{4}\left[\mathrm{erf}\left(\frac{1+i}{\sqrt{2}}z\right) - i\mathrm{erf}\left(\frac{1-i}{\sqrt{2}}z\right)\right] \tag{10}$$

$$= \frac{-1+i}{4}\left[\mathrm{erfi}\left(\frac{(1-i)\sqrt{\pi}z}{2}\right) + i\mathrm{erfi}\left(\frac{(1+i)\sqrt{\pi}z}{2}\right)\right] \tag{11}$$

Fresnel integral $C(z)$

$$C(z) = \sqrt{\frac{\pi}{2}}\frac{1-i}{4}\left[\mathrm{erf}\left(\frac{1+i}{\sqrt{2}}z\right) + i\mathrm{erf}\left(\frac{1-i}{\sqrt{2}}z\right)\right] \tag{12}$$

$$= \frac{1+i}{4}\left[\mathrm{erfi}\left(\frac{(1-i)\sqrt{\pi}z}{2}\right) - i\mathrm{erfi}\left(\frac{(1+i)\sqrt{\pi}z}{2}\right)\right] \tag{13}$$

3 Applications

3.1 Goodwin–Staton integral

These extended routines allow us to evaluate other special functions related to the error functions. An illustrative example is the computation of the Goodwin–Staton integral (see [2] and [1, §7.2(v)]), which is one of the generalizations of the complementary error function. The Goodwin–Staton integral representation is given by

$$G(z) = \int_0^\infty \frac{e^{-t^2}}{t+z}\,dt, \quad |\mathrm{ph} z| < \pi. \tag{14}$$

We consider the case $z = x$ and $x \in \mathbb{R}^+$. The Goodwin–Staton integral is defined as *elementary*, since it can be expressed in terms of the error function and the exponential integral as follows,

$$G(x) = -\frac{\pi}{2}ie^{-x^2}\mathrm{erf}(ix) - \frac{1}{2}e^{-x^2}E_i(x^2).$$

By using the well-known connection formulas $E_i(x) = -E_1(-x)$ and $\mathrm{erfi}(z) = -i\mathrm{erf}(iz)$, the previous expression can be rewritten in the following form

$$G(x) = \frac{e^{-x^2}}{2}(\pi\mathrm{erfi}(x) + E_1(-x^2)). \tag{15}$$

Equation (15) can be directly evaluated for small x (`nag_goodwin_staton`), however for large x it will quickly overflow. In order to handle these cases we propose some workarounds: The first term can be expressed in terms of Dawson's integral

$$\frac{\pi\mathrm{erfi}(x)}{2e^{x^2}} = \sqrt{\pi}F(x),$$

and the remaining term can be transformed into an asymptotic expansion, removing the explicit evaluation of e^{-x^2},

$$\frac{e^{-x^2}E_1(-x^2)}{2} \sim -\frac{1}{2x^2}\sum_{k=0}^{\infty}\frac{k!}{(x^2)^k}, \quad x \to \infty.$$

Hence,

$$G(x) \sim \sqrt{\pi}F(x) - \frac{1}{2x^2}\sum_{k=0}^{\infty}\frac{k!}{(x^2)^k}, \quad x \to \infty. \tag{16}$$

This method is implemented in `nag_goodwin_staton_3`. Alternatively, we might consider the following integral representation of the second term,

$$\frac{e^{-x^2}E_1(-x^2)}{2} = -\frac{1}{2}\int_0^1 \frac{dt}{x^2+\log(t)}.$$

Thus,

$$G(x) = \sqrt{\pi}F(x) - \frac{1}{2}\int_0^1 \frac{dt}{x^2+\log(t)}. \tag{17}$$

Finally, integral (14) can be evaluated using one of the available numerical quadrature routines, for instance `nag_quad_1d_inf (d01am)`. This is implemented in `nag_goodwin_staton_2`.

Table 1 shows CPU times and relative errors for several values of x using the three proposed methods. The first proposed method is generally faster than evaluating the integral, since the asympotitc expansion exhibits rapid convergence as x increases. Other approaches based on Chebyshev expansions are described in [3].

x	**nag_goodwin_staton**	**time**	**nag_goodwin_2**	**time**	**nag_goodwin_3**	**time**
0.5	3.73e−16	0.0282	0.0	0.0215	–	–
1	0.0	0.0276	1.83e−16	0.0201	–	–
5	0.0	0.0307	0.0	0.0157	–	–
10	1.16e−15	0.0275	1.65e−16	0.0215	3.3e−16	0.0172 (13)
20	0.0	0.0282	0.0	0.0202	0.0	0.0167 (9)
50	overflow	–	1.98e−16	0.0207	1.98e−16	0.0147 (7)
100	overflow	–	0.0	0.0142	0.0	0.0145 (6)
1000	overflow	–	1.22e−16	0.0158	2.45e−16	0.0158 (4)
10000	overflow	–	0.0	0.0164	3.06e−16	0.0154 (3)

Table 1: Relative errors obtained with each method. Number of iterations for asymptotic series within parentheses. Relative errors are computed using as a reference Wolfram Alpha with 50 digits of precision. Laptop: Intel® Core$^{\mathrm{TM}}$ i5-5300U CPU at 2.30GHz.

3.2 Voigt profile and functions

The Voigt profile is a line profile resulting from the convolution of the two broadening mechanisms, a Gaussian profile and a Lorentzian profile. Voigt profiles typically arise in many branches of spectroscopy and diffraction, and in particular, in the modelling and analysis of radiative transfer in the atmosphere. A closed form is given by

$$V(x;\sigma,\gamma) = \frac{\Re(w(z))}{\sigma\sqrt{2\pi}}, \quad z = \frac{x+i\gamma}{\sigma\sqrt{2}}, \tag{18}$$

where $\Re(w(z))$ indicates the real part of the Faddeeva function. The Voigt functions U, V and H (sometimes called the line broadening function) for $x \in \mathbb{R}$ and $t > 0$ are defined by

$$U(x,t) = \frac{1}{\sqrt{4\pi t}}\int_{-\infty}^{\infty}\frac{e^{-(x-y)^2/(4t)}}{1+y^2}\,dy\,. \tag{19}$$

$$V(x,t) = \frac{1}{\sqrt{4\pi t}}\int_{-\infty}^{\infty}\frac{ye^{-(x-y)^2/(4t)}}{1+y^2}\,dy\,. \tag{20}$$

Voigt function U and V are connected as follows

$$U(x,t) + iV(x,t) = \sqrt{\frac{\pi}{4t}} e^z \operatorname{erfc}(z) = \sqrt{\frac{\pi}{4t}} w(iz), \quad z = \frac{1-ix}{2\sqrt{t}}, \tag{21}$$

and the line broadening function is defined by

$$H(a,u) = \frac{1}{a\sqrt{\pi}} U\left(\frac{u}{a}, \frac{1}{4a^2}\right). \tag{22}$$

Voigt functions and Voigt profile have the following functional relation

$$V(x;\sigma,\gamma) = \frac{H(a,u)}{\sqrt{2\pi}\sigma}, \quad a = \frac{\gamma}{\sigma\sqrt{2}} \quad \text{and} \quad u = \frac{x}{\sigma\sqrt{2}}, \tag{23}$$

where γ is the half-width at half-maximum (HWHM) of the Lorentzian profile and σ is the standard deviation of the Gaussian profile, related to its HWHM α, $\alpha = \sigma\sqrt{2\log(2)}$. Figure 1 compares all three profiles for $\alpha = 0.5$ and $\gamma = 0.2$.

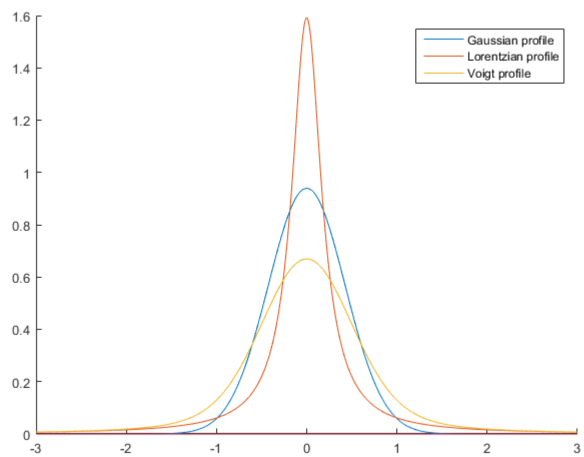

Figure 1: Voigt profile vs Gaussiand profile vs Lorentzian profile.

References

[1] NIST Digital Library of Mathematical Functions. `http://dlmf.nist.gov/`, Release 1.0.10 of 2015-08-07. Online companion to [5].

[2] Table of $\int_0^\infty \frac{e^{-u^2}}{u+x}\, du$. The Quarterly Journal of Mechanics & Applied Mathematics, Volume 1 (1), pp. 319-326, (1948).

[3] Allan Mcleod. *Algorithm 757, MISCFUN: A software package to compute uncommon special functions.* ACM Transactions on Mathematical Software, Volume 22, Number 3, pp. 288-301, (1996).

[4] Mofresh R. Zaghloul and Ahmed N. Ali, *Algorithm 916: Computing the Faaddeyeva and Voigt Functions.* ACMS Trans. Math. Soft. 38(2), 15, (2011). `http://arxiv.org/ftp/arxiv/papers/1106/1106.0151.pdf`

[5] 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 [1].

4 Code

Listing 1: Complementary error function for complex z

```
function [result, ifail] = nag_erfc_complex(z)
% Compute complementary error function for complex z.
% erfc(z) = exp(-z^2) * w(z * 1j)

if (imag(z) == 0)
    [result, ifail] = nag_specfun_erfc_real(z);
else
    [w_z, ifail] = nag_specfun_erfc_complex(complex(z * 1j));
    if (ifail == 0)
        result = exp(-z^2) * w_z;
    else
        error('Error occurred.')
    end
end
```

Listing 2: Error function for complex z

```
function [result, ifail] = nag_erf_complex(z)
% Error function for complex z.
% erf(z) = 1 - erfc(z)

if (imag(z) == 0)
    [result, ifail] = nag_specfun_erf_real(z);
else
    [erfc, ifail] = nag_erfc_complex(z);
    if (ifail == 0)
        result = 1 - erfc;
    else
        error('Error occurred.')
    end
end
```

Listing 3: Imaginary error function

```
function [result, ifail] = nag_erfi_complex(z)
% Imaginary error function for complex z

[erf, ifail] = nag_erf_complex(z * 1j);
if (ifail ~= 0)
    error('Error occurred.')
else
    result = -1j * erf;
end
```

Listing 4: Dawson function for complex z

```
function [result, ifail] = nag_dawson_complex(z)
% Dawson function for complex z.
% Dawson(z) = sqrt(pi) / 2 * exp(-z^2) * erfi(z)

if (imag(z) == 0)
    [result, ifail] = nag_specfun_dawson(z);
else
    [erfi, ifail] = nag_erfi_complex(z);
    if (ifail ~= 0)
        error('Error occurred.')
    else
        result = sqrt(pi)* 0.5 * exp(-z^2) * erfi;
    end
end
```

Listing 5: Scaled complement of error function for complex z

```
function [result, ifail] = nag_erfcx_complex(z)
% Scaled complement of error function, erfcx(z)

if (imag(z) == 0)
    [result, ifail] = nag_specfun_erfcx_real(z);
else
    [result, ifail] = s15dd(complex(z * 1j));
    if (ifail ~= 0)
        error('Error occurred.')
    end
end
```

Listing 6: Fresnel integral S for complex z

```
function [result, ifail] = nag_fresnel_s_complex(z)
% Fresnel integral S(z) for complex z.

if (imag(z) == 0)
    [result, ifail] = nag_specfun_fresnel_s(z);
else
    [erfi1,ifail] = nag_erfi_complex((1 - 1j) * 0.5 * sqrt(pi) * z);
    if (ifail ~= 0)
        error('Error occurred.')
    end
    [erfi2,ifail] = nag_erfi_complex((1 + 1j) * 0.5 * sqrt(pi) * z);
    if (ifail ~= 0)
        error('Error occurred.')
    end

    term1 = (-1 + 1j) * 0.25;
    term2 = erfi1 + 1j * erfi2;

    result = term1 * term2;
end
```

Listing 7: Fresnel integral C for complex z

```
function [result, ifail] = nag_fresnel_c_complex(z)
% Fresnel integral C(z) for complex z.

if (imag(z) == 0)
    [result, ifail] = nag_specfun_fresnel_c(z);
else
    [erfi1,ifail] = nag_erfi_complex((1 - 1j) * 0.5 * sqrt(pi) * z);
    if (ifail ~= 0)
        error('Error occurred.')
    end
    [erfi2,ifail] = nag_erfi_complex((1 + 1j) * 0.5 * sqrt(pi) * z);
    if (ifail ~= 0)
        error('Error occurred.')
    end

    term1 = (1 + 1j) * 0.25;
    term2 = erfi1 - 1j * erfi2;
    result = term1 * term2;
end
```

Listing 8: Goodwin–Staton integral for real positive x

```
function result = nag_goodwin_staton(x)
% GS(x) = e^(-x^2) / 2 (pi * erfi(x) + E1(-x^2))

t1 = nag_erfi_complex(x) * pi;
t2 = nag_specfun_integral_exp(-x^2);

result = (t1 + t2) * exp(-x^2) * 0.5;
```

Listing 9: Goodwin–Staton numerical integration

```
function result = nag_goodwin_staton2(z)
% GS(x) solved using numerical integration method

bound = 0;
inf = int64(1);
epsabs = 0;
epsrel = 1e-10;
f = @(x) exp(-x^2) / (x + z);

result = nag_quad_1d_inf(f, bound, inf, epsabs, epsrel);
```

Listing 10: Goodwin–Staton asymptotic

```
function [result, k] = nag_goodwin_staton3(z)
% compute asymptotic series E_1
maxiter = 100;
acc = false;
tol = 10 * x02aj;

s = 1;
sp = 1;
n = 1;
d = 1;
z2 = z * z;

for k=1:maxiter
    n = n * k;
    d = d * z2;
    s = s + n / d;
    if (abs((s - sp) / sp) < tol)
        if (acc) % two consecutive iterations
            break
        else
            acc = true;
        end
    else
        sp = s;
    end
end

result = -s / z2;

% Dawson's integral
result2 = sqrt(pi) * nag_specfun_dawson(z);
result = 0.5 * result + result2;
```

Listing 11: Voigt profile

```
function v_profile = voigt_profile(x, alpha, gamma)
% compute Voigt profile using Faddeeva function

sigma = alpha / sqrt(2 * log(2));
q = (sigma * sqrt(2 * pi));
z = complex(x, gamma) / (sigma * sqrt(2));

v_profile = real(nag_specfun_erfc_complex(z)) / q;
```

Listing 12: Lorentzian profile

```
function l_profile = lorentzian_profile(x, gamma)

% Return Lorentzian line shape at x with HWHM gamma
l_profile = gamma ./ (pi * (x.^2 + gamma^2));
```

Listing 13: Gaussian profile

```
function g_profile = gaussian_profile(x, alpha)

% Return Gaussian line shape at x with HWHM alpha
aux = exp(-(x / alpha).^2 * log(2));
g_profile = sqrt(log(2) / pi) / alpha * aux;
```

Listing 14: Example Voigt profile

```
x = linspace(-3,3,2000);
alpha = 0.5;
gamma = 0.2;

mesh = size(x, 2);
vp = zeros(mesh);

for i=1:mesh
    vp(i) = voigt_profile(x(i), alpha, gamma);
end

hold on
plot(x, gaussian_profile(x, alpha))
plot(x, lorentzian_profile(x,gamma))
plot(x, vp)
hold off

legend('Gaussian profile', 'Lorentzian profile', 'Voigt profile')
```