SUPPLEMENTARY MATERIALS: COMPUTATION OF THE COMPLEX ERROR FUNCTION USING MODIFIED TRAPEZOIDAL RULES*

MOHAMMAD AL AZAH† AND SIMON N. CHANDLER-WILDE‡

SM1. Matlab codes. In the main paper [SM3] we have proposed a method for computing the Faddeeva function $w(z)$, defined by [SM4, (7.2.3)]

$$w(z) := \mathrm{e}^{-z^2}\mathrm{erfc}(-\mathrm{i}\,z)$$

for all $z = x + \mathrm{i}y$ $(x, y \in \mathbb{R})$, where erfc is the complementary error function

$$\mathrm{erfc}(z) := \frac{2}{\sqrt{\pi}} \int_z^\infty \mathrm{e}^{-t^2}\, dt.$$

In this supplementary material we provide Matlab codes that implement the new method, namely[1]:

i) The Matlab function `wTrap.m` (Table SM1) for computing our approximation $w_N(z)$ to $w(z)$ for z in the first quadrant of the complex plane. This function was used to produce the numerical results in [SM3, §4].

ii) The Matlab function `wTrapWCP.m` (Table SM2) that makes use of `wTrap.m` and the symmetries (SM9) to extend the approximation $w_N(z)$ to the whole complex plane.

For $t \in \mathbb{R}$ let $\varphi(t) := t - \lfloor t \rfloor \in [0, 1)$ denote the fractional part of t. Then [SM3, (10)–(15)] our proposed approximation to $w(z)$ is the function $w_N(z)$ defined, for $x, y \geq 0$ and integer $N \geq 0$, by

$$w_N(z) := \begin{cases} w_N^{\mathrm{M}}(z), & \text{if } y \geq \max\left(x, \pi/h\right), \\ w_N^{\mathrm{MT}}(z), & \text{if } y < x \quad \text{and} \quad 1/4 \leq \varphi\left(x/h\right) \leq 3/4, \\ w_N^{\mathrm{MM}}(z), & \text{otherwise,} \end{cases} \tag{SM1}$$

where

$$h := \sqrt{\pi/(N+1)}\,, \tag{SM2}$$

$$w_N^{\mathrm{M}}(z) := \frac{2\mathrm{i}\,h\,z}{\pi} \sum_{k=0}^{N} \frac{\mathrm{e}^{-t_k^2}}{z^2 - t_k^2}, \tag{SM3}$$

$$w_N^{\mathrm{MM}}(z) := \frac{2\,\mathrm{e}^{-z^2}}{1 + \mathrm{e}^{-2\mathrm{i}\pi z/h}} + w_N^{\mathrm{M}}(z), \tag{SM4}$$

$$w_N^{\mathrm{MT}}(z) := \frac{2\,\mathrm{e}^{-z^2}}{1 - \mathrm{e}^{-2\mathrm{i}\pi z/h}} + \frac{\mathrm{i}\,h}{\pi z} + \frac{2\mathrm{i}\,h\,z}{\pi} \sum_{k=1}^{N} \frac{\mathrm{e}^{-\tau_k^2}}{z^2 - \tau_k^2}, \tag{SM5}$$

$$t_k := (k + 1/2)h, \quad \text{and} \quad \tau_k := kh. \tag{SM6}$$

*Supplementary material for SINUM MS#M137303.
https://doi.org/10.1137/20M1373037

†School of Social and Basic Sciences, Al Hussein Technical University (HTU), Amman, Jordan (mohammad.azah@htu.edu.jo).

‡Department of Mathematics and Statistics, University of Reading, Whiteknights, PO Box 220, Reading RG6 6AX, UK (S.N.Chandler-Wilde@reading.ac.uk).

[1]The codes in these supplementary materials are also available at github, see https://github.com/sms03snc/Faddeeva.

We extend this approximation from the first quadrant to the full upper half-plane by defining [SM3, (16)]

$$w_N(z) := \overline{w_N(-\overline{z})}, \text{ if } y \geq 0 \text{ and } x < 0, \tag{SM7}$$

and then extend to the whole complex plane by defining [SM3, (16)]

$$w_N(z) := 2\mathrm{e}^{-z^2} - w_N(-z), \text{ if } y < 0. \tag{SM8}$$

These last two equations are just the symmetries that hold for $w(z)$ for all complex z, that [SM5, (3.1) and (3.2)]

$$w(-z) = 2\mathrm{e}^{-z^2} - w(z) \quad \text{and} \quad w(\overline{z}) = \overline{w(-z)}. \tag{SM9}$$

In the main paper [SM3, Theorem 1.1] we proved the following bounds on the absolute and relative errors, that, for $N \geq 0$:

$$\begin{aligned} &|w(z) - w_N(z)| \leq C_1\,\mathrm{e}^{-\pi N}, \quad \text{for all } x, y \in \mathbb{R}, \quad \text{and} \\ &\frac{|w(z) - w_N(z)|}{|w(z)|} \leq C_2\,\sqrt{N+1}\,\mathrm{e}^{-\pi N}, \quad \text{if } x \in \mathbb{R} \text{ and } y \geq 0, \end{aligned}$$

where $C_1 \approx 0.67$ and $C_2 \approx 3.97$ are given by [SM3, (75)–(76)]. The results in [SM3, §4], using the Matlab function `wTrap.m` that implements (SM1)–(SM6), confirm these bounds hold until machine precision takes effect, in particular that the absolute and relative errors in $w_N(z)$ are $< 2 \times 10^{-15}$ in the first quadrant with $N = 11$.

Table SM1 displays the Matlab function `wTrap.m` that implements (SM1)–(SM6), providing approximations to $w(z)$ in the first quadrant. (Note that the subfunctions `wM`, `wMM`, and `wMT` evaluate (SM3), (SM4), and (SM5), respectively.) The inputs are:

- `z` scalar, vector, or array of complex numbers with non-negative real and imaginary parts;
- `N` a positive integer;

and the output is

- `w` scalar, vector, or array of the same size as `z`, containing the corresponding values of $w_N(z)$, e.g. if `z` is a two-dimensional array then `w(i,j)` $= w_N$(`z(i,j)`).

Table SM2 displays the Matlab function `wTrapWCP.m` that implements (SM7) and (SM8), computing the approximation $w_N(z)$ to $w(z)$ in the whole complex plane, relying on `wTrap.m` to compute values in the first quadrant. The inputs and outputs are the same as for `wTrap.m` except that `z` is a scalar, vector, or array of complex numbers with no constraints on the signs of real or imaginary parts. The method used is to rewrite (SM7) and (SM8) as

$$w_N(z) = \begin{cases} \overline{w_N(-\overline{z})}, & \text{if } x < 0 \text{ and } y \geq 0, \\ 2\mathrm{e}^{-z^2} - w_N(-z), & \text{if } x \leq 0 \text{ and } y < 0, \\ 2\mathrm{e}^{-z^2} - \overline{w_N(\overline{z})}, & \text{if } x > 0 \text{ and } y < 0, \end{cases}$$

noting that the arguments of the function w_N on the right hand side are all in the first quadrant and so can be computed by `wTrap.m`.

Table SM3 displays the Matlab script `test_wTrapWCP.m` that tests the implementation of `wTrapWCP.m` by checking the accuracy of $w_{11}(z)$, for $|\mathrm{Re}(z)|, |\mathrm{Im}(z)| \leq 1/2$, as calculated by `wTrapWCP.m`, against approximation with a truncated power series

```
function w = wTrap(z,N)
% Computes Faddeeva function w(z) for z = x+iy with x,y >=0.
% Inputs: z complex (scalar, vector, or array).
%         N positive integer (N+1 is the number of quadrature points used);
%           the choice N = 11 is recommended for absolute and relative
%           errors < 2e-15
% Output: w complex array of same dimensions as z containing
%           estimates for values of w(z)
h = sqrt(pi/(N+1)); H = pi/h;
rz = real(z); rzh = rz/h; iz = imag(z); buff = abs(rzh-floor(rzh)-0.5);
select1 = iz >= max(rz,H); select2 = (iz < rz) & (buff <= 0.25);
select3 = ~(select1|select2);
w = zeros(size(z)); w(select1) = wM(z(select1),N);
w(select2) = wMT(z(select2),N); w(select3) = wMM(z(select3),N);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Ih = wM(z,N)
% Midpoint rule approximation
z2 = z.*z; az = (2i/H)*z;
t = h*((N:-1:1)+0.5); t2 = t.^2; et2 = exp(-t2); h0 = 0.5*h;
Sum = exp(-h0^2)./(z2-h0^2);
for n = 1 : N
    Sum = Sum + et2(n)./(z2-t2(n));
end
Ih = az.*Sum;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Ihstar = wMM(z,N)
% Modified midpoint rule approximation
z2 = z.*z; az = (2i/H)*z;
t = h*((N:-1:1)+0.5); t2 = t.^2; et2 = exp(-t2); h0 = 0.5*h;
Sum = exp(-h0.^2)./(z2-h0.^2);
for n = 1 : N
    Sum = Sum + et2(n)./(z2-t2(n));
end
Ch = 2./(exp(z2).*(1+exp((-2i*H)*z))); % This is the modification
Ihstar = az.*Sum + Ch;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Ihstar = wMT(z,N)
% Modified trapezium rule approximation
z2 = z.*z; az = (2i/H)*z;
tau = h*(N:-1:1); tau2 = tau.^2; et2 = exp(-tau2);
Sum = et2(1)./(z2-tau2(1));
for n = 2 : N
    Sum = Sum + et2(n)./(z2-tau2(n));
end
Ch2 = 2./(exp(z2).*(1-exp((-2i*H)*z))); % This is the modification
Ihstar = (1i/H)./z + az.*Sum + Ch2;
end
end
```

TABLE SM1
Matlab function to evaluate $w_N(z)$ for complex z in the first quadrant.

```
function w = wTrapWCP(z,N)
% Computes Faddeeva function w(z) for arbitrary complex z.
% See 'Computation of the complex error function using modified trapezoidal
% rules', M Al Azah and S N Chandler-Wilde, 2020, for details of method.
% Inputs: z complex (scalar, vector, or array).
%         N positive integer (N+1 is the number of quadrature points used);
%           the choice N = 11 is recommended for absolute errors (and
%           relative errors in the upper half-plane) < 2e-15
% Output: w complex array of same dimensions as z containing
%           estimates for values of w(z)
selectXneg = real(z) < 0; selectYneg = imag(z) < 0;
selectNotBoth = xor(selectXneg,selectYneg);
zYneg = z(selectYneg);
z(selectXneg) = -z(selectXneg); z(selectNotBoth) = conj(z(selectNotBoth));
w = wTrap(z,N);
w(selectNotBoth) = conj(w(selectNotBoth));
w(selectYneg) = 2*exp(-zYneg.*zYneg) - w(selectYneg);
end
```

TABLE SM2
Matlab function to evaluate $w_N(z)$ for complex z in the whole complex plane.

```
% Test of accuracy and correct implementation of wTrapWCP.m by
% comparing against truncated power series for w(z) given in (2.16) of
% Poppe and Wijers, ACM TOMS (1990), taking N = 12
points = (-200:200)/400; % Uniform grid on [-1/2,1/2] including 0
[x,y] = meshgrid(points);
% Now set z to be matrix of complex numbers covering {z:x,y\in [-1/2,1/2]},
% including points on the positive and negative real and imaginary axes
z = complex(x,y);
wN = wTrapWCP(z,11); % Computing w_N(z) with N = 11
q = z.^2;
w = ((((((((((q/11975040000+1/918086400).*q+1/76204800).*q+ ...
    1/6894720).*q+1/685440).*q+1/75600).*q+1/9360).*q+1/1320).*q+ ...
    1/216).*q+1/42).*q+1/10).*q+1/3).*q+1;
% Now compute w(z) by Poppe and Wijers (2.16) with N = 12
w = exp(-q).*(1+(2i/sqrt(pi))*z.*w);
ErrorPS = max(max(abs(wN-w))) % Maximum of |w_N(z)-w(z)|
```

TABLE SM3
Matlab script file, testing against [SM5, (2.16)] with $N = 12$.

([SM5, (2.16)] with $N = 12$). Running this in Matlab (R2017b) produces the output `ErrorPS = 1.132209773400735e-15`, indicating that the maximum difference is $< 1.2 \times 10^{-15}$ between $w_{11}(z)$ and [SM5, (2.16)] with $N = 12$ for this range of z.

Tables SM5 and SM6 display `test_accuracy_Q.m` and `test_time.m`, the Matlab script files that we used to produce the results in [SM3, Table 1]. These files call our Matlab function `wTrap.m`, displayed in Table SM1. They also call the Matlab functions published by other authors that are discussed below [SM3, Table 1], namely `cef.m` taken from [SM6, Table 1], `fadsamp.m` taken from [SM2, Appendix A], and `Faddeyeva_v2.m` discussed in [SM7]. The script file `test_accuracy_Q.m` uses the ADVANPIX Multiprecision Computing Toolbox for Matlab [SM1] to calculate high precision benchmark values of $w(z)$ to use as the true values when computing errors.

```
function w = wTrap_Q(z,N)
% Computes Faddeeva function w(z) for z = x+iy with x,y >=0.
% This version uses ADVANPIX, the Multiprecision Computing Toolbox.
% See 'Computation of the complex error function using modified trapezoidal
% rules', M Al Azah and S N Chandler-Wilde, 2021, for details of method.
% Inputs: z complex (scalar, vector, or array).
%         N positive integer (N+1 is the number of quadrature points used);
%           the choice N = 11 is recommended for absolute and relative
%           errors < 2e-15
% Output: w complex array of same dimensions as z containing
%           estimates for values of w(z)
piQ = mp('pi'); % Multiprecision computation of pi, using ADVANPIX mp
zQ = mp(z); % Converting input z to multiprecision (if it isn't already)
h = sqrt(piQ/(N+1)); H = piQ/h;
rz = real(zQ); rzh = rz/h; iz = imag(zQ); buff = abs(rzh-floor(rzh)-0.5);
select1 = iz >= max(rz,H); select2 = (iz < rz) & (buff <= 0.25);
select3 = ~(select1|select2);
w = mp(zeros(size(zQ))); w(select1) = wM(zQ(select1),N);
w(select2) = wMT(zQ(select2),N); w(select3) = wMM(zQ(select3),N);
```

TABLE SM4

Matlab function to evaluate $w_N(z)$ for complex z in the first quadrant using the multiprecision toolbox ADVANPIX. Shown is just the main function. The subfunctions are as in Table SM1.

These high precision values are computed using ADVANPIX in its default quadruple precision mode (compliant with IEEE 754-2008). (This is the default but can also be set by the command `mp.Digits(34)`.) The script file calls `wTrap_Q(z,N)` displayed in Table SM4, a high precision version of `wTrap`, to calculate these accurate values, with $N = 20$. As discussed in the main paper [SM3, §4], we expect that these benchmark values have absolute and relative errors no larger than 3.5×10^{-28} and 9.4×10^{-27}, respectively.

The results produced by the script file `test_accuracy_Q.m` in Table SM5 are those shown in the maximum absolute and relative error columns of [SM3, Table 1]. Figure SM1 shows typical output from running the script file `test_time.m` of Table SM6 in Matlab version 9.3.0.713579 (R2017b) on a laptop with a single Intel64 Family 6 Model 78 2.40 GHz processor, 8,124 MB physical memory, running under the Microsoft Windows 10 Enterprise operating system, version 10.0.17134 N/A Build 17134 (and with `maxNumCompThreads` set to its default value of 2). We tabulate these results in Table SM7. These results are very similar but not identical to those in [SM3, Table 1]. Timing results, as is clear from the standard deviation column in Table SM7, are not completely reproducible, even on the same computer with the same configuration.

REFERENCES

[SM1] *ADVANPIX Multiprecision Computing Toolbox for MATLAB*, 2021, https://www.advanpix.com/. Downloaded 8 June 2021.

[SM2] S. M. Abrarov, B. M. Quine, and R. K. Jagpal, *A sampling-based approximation of the complex error function and its implementation without poles*, Appl. Numer. Math., 129 (2018), pp. 181–191.

[SM3] M. Al Azah and S. N. Chandler-Wilde, *Computation of the complex error function using modified trapezoidal rules*, SIAM J. Numer. Anal., 59 (2021), pp. 2346–2367.

[SM4] *NIST Digital Library of Mathematical Functions.* Release 1.0.28 of 2020-09-15, http://dlmf.

```
% Set 34 digits, so IEEE quadruple precision, in the ADVANPIX
% Multiprecision Computing Toolbox (advanpix.com)
mp.Digits(34)
% Create a 2001X801 array containing the 1,602,801 complex numbers
% z = 10^p exp(i theta), with p = -6(0.0006)6, theta = 0(pi/1600)pi/2
p = -6:0.0006:6;
theta = (0:0.1125:90)*pi/180;
z = zeros(length(p),length(theta));
for i = 1: length(p)
    for j = 1: length(theta)
        z(i,j) = 10^p(i)*exp(1i*theta(j));
    end
end
% Calculate accurate values for w(z), by the call wTrap_Q(z,N)
% which uses our new method with N = 20 evaluated in quadruple
% precision via the ADVANPIX toolbox.
truew = double(wTrap_Q(z,20)); wabs = abs(truew);
% Compute maximum absolute and relative errors over these z values for ...
% ... our new method with N = 11
errors = abs(truew-wTrap(z,11));
a1 = max(max(errors)); r1 = max(max(errors./wabs));
% ... Weideman's approximation with N = 40
errors = abs(truew-cef(z,40));
a2 = max(max(errors)); r2 = max(max(errors./wabs));
% ... Abrarov et al's approximation
errors = abs(truew-fadsamp(z));
a3 = max(max(errors)); r3 = max(max(errors./wabs));
% ... Zaghloul and Ali's approximation
errors = abs(truew-Faddeyeva_v2(z,13));
a4 = max(max(errors)); r4 = max(max(errors./wabs));
% Display these errors
format SHORTG
Absolute = [a1;a2;a3;a4]
Relative = [r1;r2;r3;r4]
```

TABLE SM5

The Matlab script file used to produce the maximum absolute and relative error columns in [SM3, Table 1].

nist.gov/. F. W. J. Olver et al., eds.

[SM5] G. POPPE AND C. WIJERS, *More efficient computation of the complex error function*, ACM Trans. Math. Software, 16 (1990), pp. 38–46.

[SM6] J. A. C. WEIDEMAN, *Computation of the complex error function*, SIAM J. Numer. Anal., 31 (1994), pp. 1497–1518.

[SM7] M. R. ZAGHLOUL, *Remark on "Algorithm 916: Computing the Faddeyeva and Voigt functions": efficiency improvements and Fortran translation*, ACM Trans. Math. Software, 42 (2016). Article 26 (9 pages).

```
% Create a 4001X4001 array containing the 16,008,001 complex numbers
% z = x+iy, with x = 0(0.0025)10, y = 0(0.0025)10
points = linspace(0,10,4001);
[x,y] = meshgrid(points);
z = complex(x,y);
% Setting up function handles for exp(-z.*z), wTrap(z,11), cef(z,40),
% fadsamp(z), Faddeyeva_v2(z,13), preparatory to timing their execution
% with the Matlab function timeit (type "help timeit" in Matlab for
% more explanation)
f0 = @() exp(-z.*z);
f1 = @() wTrap(z,11);
f2 = @() cef(z,40);
f3 = @() fadsamp(z);
f4 = @() Faddeyeva_v2(z,13);
% Now time each of these function calls NN = 25 times using timeit,
% storing the execution times (in seconds) in the vectors T0,...,T4.
NN = 25
T0 = zeros(1,NN); T1 = T0; T2 = T0; T3 = T0; T4 = T0;
for n = 1:NN
    n
    T0(n) = timeit(f0)
    T1(n) = timeit(f1)
    T2(n) = timeit(f2)
    T3(n) = timeit(f3)
    T4(n) = timeit(f4)
end
% Display statistics of the results, namely, for each function execution:
% number of timeit measurements, and mean, median, standard deviation of
% the NN measurements
T0stats = [length(T0),mean(T0),median(T0),std(T0)]
T1stats = [length(T1),mean(T1),median(T1),std(T1)]
T2stats = [length(T2),mean(T2),median(T2),std(T2)]
T3stats = [length(T3),mean(T3),median(T3),std(T3)]
T4stats = [length(T4),mean(T4),median(T4),std(T4)]
```

TABLE SM6

The Matlab script file used to produce the computing time column in [SM3, Table 1].

Function call	Computing time (seconds)		
	Mean	Median	s.d.
`exp(-z.*z)`	1.03	1.03	0.02
`wTrap(z,11)`	4.22	4.28	0.12
`cef(z,40)`	4.18	4.16	0.15
`fadsamp(z)`	5.84	5.79	0.15
`Faddeyeva_v2(z,13)`	10.89	10.87	0.09

TABLE SM7

Tabulation of the computing time results shown in Figure SM1 produced by running the script file in Table SM6. Shown are the mean, median, and standard deviation (s.d.) of 25 measurements of the computing times in seconds of the function calls indicated.

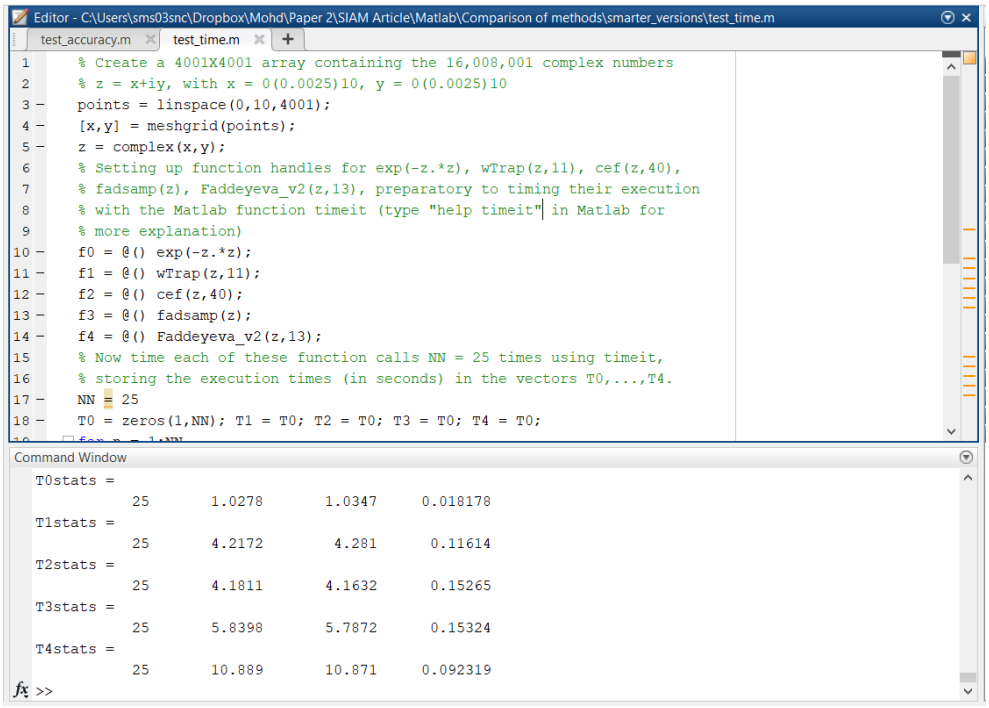

FIG. SM1. *Example output from the Matlab script file in Table SM6.*