Evaluation of complex-valued error-like functions by the exponentially-convergent trapezoidal rule

Federico Maria Guercilena

November 27, 2025

Abstract

The exponentially convergent trapezoidal rule is applied to a suitable integral representation of the Faddeeva function to derive a simple formula for its evaluation. I describe its properties, strategies for maximising its efficiency, and its coupling with other evaluation methods (asymptotic expansions and Maclaurin series). From knowledge of the values of the Faddeeva function, all other complex-valued error-like functions such as erf and erfc can be easily obtained. The resulting algorithm has been implemented in a publicly-available C/C++ library named ERFLIKE in IEEE double precision arithmetic, and tested against more widespread evaluation methods based on Taylor series and continued fractions, as provided by the widely used FADDEEVA PACKAGE. It is found that the algorithm presented here and its implementation achieve better accuracy and a more regular behaviour of the relative error over vast regions of the complex plane. In terms of speed of evaluation the ERFLIKE library also outperforms the FADDEEVA PACKAGE for complex valued arguments, although not for real-valued ones.

1 Introduction

The error function is ubiquitously found in virtually every scientific field, with wide-ranging applications in pure mathematics as well as statistics, applied mathematics, number theory, physics, etc.. It is defined by the integral [3, 7.2.1]

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \mathrm{d}t \,, \tag{1}$$

where the argument z is an arbitrary complex number. From this definition several properties of erf can be established. In particular, the error function is entire (i.e. holomorphic over the whole complex plane), and it assumes real values when its argument is real. Being the integral of an exponential it is manifestly not an elementary function, but a transcendental one.

There exists a set of functions closely related to erf which share many of its properties, in particular each of them is an entire, transcendental function. These are [3,

7.2]

$$\begin{aligned}
\operatorname{erfc}(z) &= 1-\operatorname{erf}(z) && \text{Complementary error function},\\
\operatorname{erfcx}(z) &= e^{z^2}\operatorname{erfc}(z) && \text{Scaled complementary error function},\\
\operatorname{erfi}(z) &= -i\operatorname{erf}(iz) && \text{Imaginary error function},\\
\operatorname{Dawson}(z) &= \frac{\sqrt{\pi}}{2}e^{-z^2}\operatorname{erfi}(z) && \text{Dawson integral},\\
\operatorname{w}(z) &= \operatorname{erfcx}(-iz) && \text{Faddeeva function},
\end{aligned}$$

where despite its name the imaginary error function too assumes real values for real arguments. In the following, this group of functions is referred to as *error-like* functions. Other related functions are the Voigt profile and the Voigt functions [3, 7.19].

Error-like functions are non-trivial to evaluate numerically, even though several alternative representations are known. Being entire functions they equal their Taylor series at any point in the complex plane, and in particular their Maclaurin series are well known. Furthermore there exist asymptotic expansions, representations as continued fractions and several alternative integral representations as well. Algorithms developed for their evaluation usually rely on these series expansions, e.g. joining a Maclaurin series close to the origin with a continued fraction far from it. The most well known algorithm for this purpose seems to be the one by [5], later improved by [14]. A different algorithm has been proposed in [20], essentially based on writing the Gaussian in the definition of the error function as a sum of hyperbolic cosines.

These algorithms form the basis for the FADDEEVA PACKAGE [9], an open-source C/C++ code that provides functions to evaluate all error-like functions for arbitrary complex arguments, with accuracy close to IEEE double-precision $\varepsilon_{\mathrm{DP}} \approx 2.2 \cdot 10^{-16}$. Bindings to the FADDEEVA PACKAGE functions for many popular programming and scripting languages (in particular R and Matlab/Octave) exists. Furthermore the Python library scipy and the Julia language rely on the FADDEEVA PACKAGE to provide implementations of error-like functions. Therefore, the FADDEEVA PACKAGE can be see as a sort of *de facto* standard, and in this work I employ it as reference point to gauge the performance of the algorithm described here.

This work explores a different approach to the problem of evaluating error-like functions, based on the application of the composite trapezoidal rule to the numerical quadrature of an integral representation of the Faddeeva function. The rationale behind this choice is the observation that the integral in question satisfies the conditions necessary for the trapezoidal rule to converge exponentially [18], yielding a very cheap and yet accurate evaluation formula. Similar approaches have been considered in various past works [2, 13, 8, 11, 12, 19], but none have provided a practical implementation and a complete characterisation of the speed and accuracy of these schemes.

By applying the technique of [6], I derive a remarkably simple formula to evaluate the Faddeeva function. From knowledge of its value, all other error-like functions can be easily computed. I discuss the practical considerations involved in employing this evaluation strategy and maximising its efficiency. The proposed algorithm is then implemented, targeting double precision accuracy, in an open-source C/C++ library named ERFLIKE. The accuracy and computational performance of this implementation

are then measured and compared to the FADDEEVA PACKAGE.

The algorithm proposed here and its implementation efficiently achieve double precision accuracy, and in vast regions of the complex plane they outperform the FADDEEVA PACKAGE in terms of both accuracy (with a much more regular behaviour of the relative error as the evaluation argument varies) and evaluation speed. The FADDEEVA PACKAGE retains an advantage in terms of computational performance in the case of real arguments. The algorithm based on the trapezoidal rule has however the theoretical advantage that it does not rely on *ad hoc* fitted representations as the FADDEEVA PACKAGE does and can therefore be used also for arbitrary precision computations. Finally, complementing the trapezoidal-rule algorithm with asymptotic expansions and Maclaurin series where appropriate (i.e. for very large or very small arguments, respectively) yields further improvements in efficiency, making the ERFLIKE library a better choice overall than the FADDEEVA PACKAGE.

The rest of this paper is organised as follows. Sec. 2 details the derivation of the trapezoidal-rule based approximation formula for the Faddeeva function and its implementation; Sec. 3 reports the measurement of the performance and accuracy of the resulting code, comparing with the FADDEEVA PACKAGE; Sec. 4 is dedicated to discussion and conclusions. Finally, the theoretical framework behind the exponential convergence of the trapezoidal rule is summarised in Appendix A.

2 Applying the trapezoidal rule to the complex-valued Faddeeva function

In order to evaluate an error-like function by means of the trapezoidal rule, it is necessary to work with a suitable integral representation. As it turns out the Faddeeva function $\mathrm{w}(z)$ admits such a representation (see Eq. (5) below), so that the trapezoidal rule is immediately applicable. From the value $\mathrm{w}(z)$ at any point, the values of all other error-like functions are easily obtained by closed formulas. As such, the rest of this section is dedicated to developing an evaluation algorithm targeting the Faddeeva function. Note that this choice simplifies the comparison with the FADDEEVA PACKAGE, since it also treats the Faddeeva function as central.

In the following I make extensive use of the following asymptotic expansion of the Faddeeva function for large arguments [3, 7.12.1]:

$$\mathrm{w}(z) \sim \frac{i}{\sqrt{\pi}} \sum_{n=0}^{+\infty} \frac{\left(\frac{1}{2}\right)_n}{z^{2n+1}}\,, \tag{2}$$

where $\left(\frac{1}{2}\right)_n = \left(\frac{1}{2}\right)\left(\frac{1}{2}+1\right)\left(\frac{1}{2}+2\right)\ldots\left(\frac{1}{2}+n-1\right)$ is Pochammer's symbol (rising factorial). This expansion is valid in the angular sector $-\frac{1}{4}\pi < \arg(z) < \frac{5}{4}\pi$. Within this region, the expansion justifies the simple estimate

$$|\mathrm{w}(z)| \approx \frac{1}{\sqrt{\pi}|z|}\,, \tag{3}$$

accurate for large $|z|$.

2.1 Conditioning of the Faddeeva function

Before devising an evaluation algorithm, it is useful to examine the conditioning of the target function, in order to identify regions that might pose difficulties. The relative condition number of a function is defined as the proportionality factor between its relative change and a given relative change of its argument, and can be estimated by linear approximation. It quantifies the amplification of errors in the output value due to errors in the input when evaluating the function in question.

In the case of the complex Faddeeva function, one obtains:

$$\frac{\mathrm{w}(z+\Delta z)-\mathrm{w}(z)}{\mathrm{w}(z)} = C\frac{\Delta z}{z} \quad \Rightarrow \quad C \approx \frac{z\mathrm{w}'(z)}{\mathrm{w}(z)} = \frac{2iz}{\sqrt{\pi}\mathrm{w}(z)} - 2z^2 \,. \tag{4}$$

where C denotes the condition number, Δz is an arbitrary displacement and $\mathrm{w}'(z)$ indicates the derivative of $\mathrm{w}(z)$. The last equality results from the identity $\mathrm{w}'(z) = 2i/\sqrt{\pi} - 2z\mathrm{w}(z)$ [3, 7.10.2]. Note that the Faddeeva function is entire, so this estimate is valid on the whole complex plane.

In the angular sector $-\frac{1}{4}\pi < \arg(z) < \frac{5}{4}\pi$, $|\mathrm{w}(z)|$ attains a maximum of 1 at $z = 0$, and otherwise decreases asymptotically as $\sim 1/\sqrt{\pi}|z|$ as noted above. As such $|C|$ is never larger than ~ 1.44 and tends towards 1 for large $|z|$. The exceptions to this behaviour occur at the zeros of $\mathrm{w}(z)$, where C diverges[1]. Outside this angular sector we have $\mathrm{w}(z) \sim 2e^{-z^2}$, resulting in $|C| \asymp 2|z|^2$.

Summarising, we can expect to be able to accurately evaluate $\mathrm{w}(z)$ without particular difficulties if $-\frac{1}{4}\pi < \arg(z) < \frac{5}{4}\pi$, except in the vicinity of a zero; while for $-\frac{3}{4}\pi \leq \arg(z) \leq -\frac{1}{4}\pi$ achieving the desired accuracy will become increasingly difficult as $|z|$ grows.

2.2 Trapezoidal rule formula

As mentioned, the Faddeeva function admits the following integral representation [3, 7.7.2]:

$$\mathrm{w}(z) = \frac{iz}{\pi}\int_{-\infty}^{+\infty} \frac{e^{-t^2}}{z^2-t^2}\mathrm{d}t \,. \tag{5}$$

This expression is valid for $\mathrm{Im}(z) > 0$, but also for real z by interpreting the integral as a Cauchy principal part. This representation has the same form as Eq. (18) with $K(t;z) = iz/[\pi(z^2-t^2)]$, so that formula Eq. (28) can be applied to estimate the integral via the composite trapezoidal rule (see Sec. A). This requires knowledge of the poles of K: in this case two simple poles located at $t = \pm z$. The resulting formula is singular for particular values of z, and to handle this complication it is useful to consider its staggered version too, resulting from Eq. (35). The residues at the poles of K necessary

[1] The zeros of $\mathrm{w}(z)$ asymptotically approach the lines $\arg z = -\frac{1}{4}\pi$ and $\arg z = \frac{5}{4}\pi$, i.e. they are just inside the region of validity of the asymptotic expansion Eq. (2) [4].

to apply the formulas are easily evaluated as:

$$\operatorname{Res}\left[\frac{ize^{-t^2}}{\pi(z^2-t^2)(1-e^{-2\pi it/h})}\right]_{t=\pm z} = \pm\frac{e^{-z^2}}{2\pi i(1-e^{\mp 2\pi iz/h})},$$
$$\operatorname{Res}\left[\frac{ize^{-t^2}}{\pi(z^2-t^2)(1+e^{-2\pi it/h})}\right]_{t=\pm z} = \pm\frac{e^{-z^2}}{2\pi i(1+e^{\mp 2\pi iz/h})},$$
$$\operatorname{Res}\left[\frac{ize^{-t^2}}{\pi(z^2-t^2)}\right]_{t=\pm z} = \mp\frac{e^{-z^2}}{2\pi i}.$$

Putting all together, the result is the following formula:

$$\mathrm{w}(z) = \begin{cases} \dfrac{ih}{\pi z} + \dfrac{2ihz}{\pi}\displaystyle\sum_{n=1}^{+\infty}\frac{e^{-n^2h^2}}{z^2-n^2h^2} + \frac{Pe^{-z^2}}{1-e^{-2\pi iz/h}} \\ \dfrac{2ihz}{\pi}\displaystyle\sum_{n=1}^{+\infty}\frac{e^{-(n-\frac{1}{2})^2h^2}}{z^2-(n-\frac{1}{2})^2h^2} + \frac{Pe^{-z^2}}{1+e^{-2\pi iz/h}} \end{cases} - E(h;z), \tag{6}$$

where h is the distance between the quadrature nodes and the constant P assumes the values:

$$P = \begin{cases} 2 & \text{if } \mathrm{Im}(z) < \dfrac{\pi}{h}, \\ 1 & \text{if } \mathrm{Im}(z) = \dfrac{\pi}{h}, \\ 0 & \text{otherwise}. \end{cases} \tag{7}$$

Both cases in Eq. (6) are singular for particular real values of z, different between the two. The formula on the second line is the staggered version of the first (see Appendix A). Following [8], the first line is employed if $0.25 \leq \mathrm{frac}[|\mathrm{Re}(z)|/h] \leq 0.75$ and the second otherwise, where $\mathrm{frac}(x) = x - \lfloor x \rfloor$ represents the fractional part. This ensures that the denominator of the sums in Eq. (6) never vanishes, while the absolute value of the denominator of the poles contribution, $1 \pm e^{-2\pi iz/h}$, never attains values lower than $\sqrt{2}$.

The error term $E(h;z)$ in Eq. (6) reads (see Eq. (32))

$$E(h;z) = 2e^{-\pi^2/h^2}\frac{iz}{\pi}\int_{-\infty}^{+\infty}\frac{e^{-y^2}}{z^2-(y-i\frac{\pi}{h})^2}\mathrm{d}y, \tag{8}$$

and clearly vanishes exponentially as h decreases.

As stated at the beginning of the section, this formulas applies in the upper part of the complex plane. To obtain $\mathrm{w}(z)$ when $\mathrm{Im}(z) < 0$, one leverages the following identity [3, 7.4.3]:

$$\mathrm{w}(-z) = 2e^{-z^2} - \mathrm{w}(z). \tag{9}$$

2.3 Double precision implementation

Once a target accuracy ε is chosen, the value of h that allows to evaluate Eq. (6) to this accuracy with minimal computational effort can be computed, along with the number

of terms after which the infinite series can be truncated. In the following I discuss these choices (and other considerations important to achieve good performance) targeting IEEE double precision, i.e. so that the relative error satisfies:

$$\delta_{\text{rel}} = \left| \frac{\mathrm{w}_{\text{exact}} - \mathrm{w}_{\text{trapezoidal}}}{\mathrm{w}_{\text{exact}}} \right| \leq \varepsilon_{\text{DP}} = 2^{-52} \approx 2.220446049250313 \cdot 10^{-16} . \tag{10}$$

The algorithm described in the following for the evaluation of the error-like functions has been implemented in a software library named ERFLIKE, which is publicly available at [7, DOI:10.5281/zenodo.11261631]. The main component of the library is written in C/C++ in a single-file, header-only fashion. It also provides bindings for the Python language written with the help of Cython [1, 15]. Finally, an unoptimised but fully functional arbitrary precision implementation, written with the help of the `mpmath` library [17] is included too.

2.3.1 Choice of h and truncation of infinite sums

Since by definition $|E(h;z)|$ is the absolute error of Eq. (6), in order to determine a suitable value for h it is necessary to require $\delta_{\text{rel}} = |E(h;z)|/|w(z)| \leq \varepsilon_{\text{DP}}$. This inequality must hold over the upper complex plane, where Eq. (6) is to be applied. Clearly h is most strongly constrained when $|\mathrm{w}(z)|$ is small, i.e. for large $|z|$. Therefore one can employ Eq. (3) to estimate the magnitude of $\mathrm{w}(z)$, while the expression for $E(h;z)$ can be simplified according to Eq. (33) (note that $K(t;z)$ monotonically decreases for large t, so this approximation is very close to the correct value). One arrives at the condition

$$\left| \frac{E(h;z)}{\mathrm{w}(z)} \right| \approx 2e^{-\pi^2/h^2} \frac{|z|^2}{|z^2 + \pi^2/h^2|} \approx 2e^{-\pi^2/h^2} \leq \varepsilon_{\text{DP}} \quad \Rightarrow \quad h \leq \frac{\pi}{\sqrt{\log\left(\frac{2}{\varepsilon_{\text{DP}}}\right)}} , \tag{11}$$

where the second approximate equality results from the assumption of large $|z|$. The approximations introduced to find this formula tend to underestimate the correct value of h, especially for a low target accuracy. An empirically determined correction, replacing the value of h obtained from Eq. (11) by $h \to h(1 - 0.06h)$, solves this issue. Applying the correction, the value of h necessary to target double precision and adopted in the following is $h \approx 0.5022$.

Having determined h, the infinite sums in Eq. (6) need to be truncated at some finite number N of terms. When considering the first line of Eq. (6), the requirement $\delta_{\text{rel}} \leq \varepsilon_{\text{DP}}$ translates to

$$\frac{1}{|\mathrm{w}(z)|} \left| \frac{2izh}{\pi} \frac{e^{-N^2h^2}}{z^2 - N^2h^2} \right| \approx \frac{2he^{-N^2h^2}}{\sqrt{\pi}} \frac{|z|^2}{|z^2 - N^2h^2|} \approx \frac{2he^{-N^2h^2}}{\sqrt{\pi}} \leq \varepsilon_{\text{DP}} , \tag{12}$$

where again the approximate equalities stem from using Eq. (3) for large $|z|$, since the small $|z|$ region constrains N less severely. This results in

$$N = \left\lceil \left[\frac{1}{h^2} \log\left(\frac{2h}{\sqrt{\pi}\varepsilon_{\text{DP}}} \right) \right]^{\frac{1}{2}} \right\rceil = 12 , \tag{13}$$

Applying this line of reasoning to the staggered version of Eq. (6) one finds that N or $N+1$ terms should be retained, depending on the value of h and the target precision. However in all tests conducted, using the value of N found for the unstaggered case in the staggered case as well allowed recovering the desired double precision accuracy. Accordingly, in the following the value of N equals 12 in both cases.

2.3.2 Partition of the complex plane

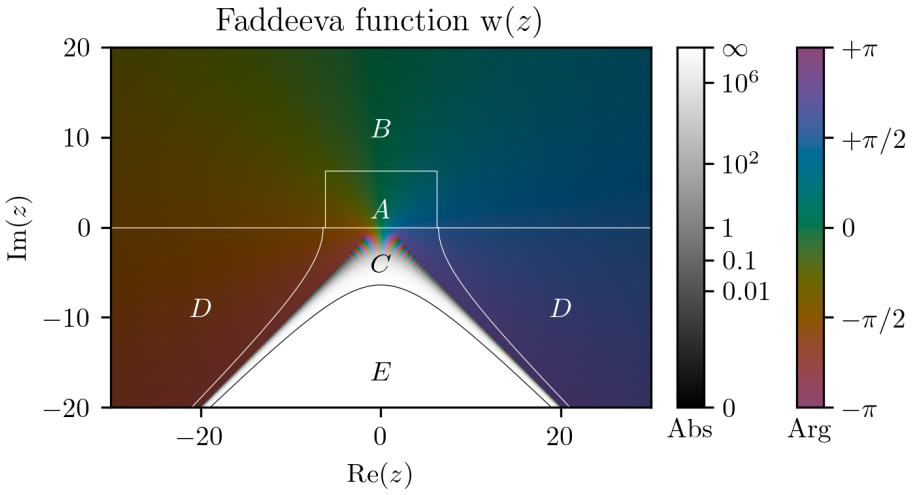

Figure 1: Domain colouring of the Faddeeva function on the complex plane. Superimposed in white and black are the boundaries of the regions of Sec. 2.3, themselves indicated by capital letters.

While formulas Eq. (6) and Eq. (9) are applicable for any argument z in the complex plane, depending on its value not all terms in those equations are necessary to reach the desired accuracy. To achieve maximum efficiency is therefore beneficial to partition the complex plane in non-overlapping regions, see Fig. 1, and apply different strategies of evaluation in each.

In the upper part of the complex plane, two regions emerge naturally depending on the possibility to neglect the poles contribution in Eq. (6). Region A is bounded by $0 \leq \mathrm{Im}(z) \leq \pi/h$ and $|\mathrm{Re}(z)| \leq 6.17$. In this region no term in Eq. (6) can be neglected without sacrificing accuracy. Region B is defined by $\mathrm{Im}(z) \geq 0$, excluding the points contained in region A. Here the poles contribution in Eq. (6) either vanishes (see Eq. (7)) or the real part of z is sufficiently large to neglect it without losing accuracy. It is sufficient to check the real part of z because $e^{-z^2}/(1 \pm e^{-2\pi i z/h})$ decreases rapidly as $\mathrm{Im}(z)$ increases from 0 towards π/h. The condition for neglecting this term is

$$\frac{2e^{-\mathrm{Re}(z)^2}}{\sqrt{2}\,|\mathrm{w}[\mathrm{Re}(z)]|} \leq \varepsilon_{\mathrm{DP}} \quad \Rightarrow \quad \mathrm{Re}(z)^2 \geq \log\left(\frac{\sqrt{\pi}\,|\mathrm{Re}(z)|}{\sqrt{2}\varepsilon_{\mathrm{DP}}}\right), \tag{14}$$

where again Eq. (3) has been employed, and the factor of $\sqrt{2}$ in the denominator is the minimum value attained by $|1 \pm e^{-2\pi i z/h}|$, see Sec. 2.2. In the case of double precision, this is equivalent to $|\mathrm{Re}(z)| > 6.17$, which justifies the definition of region A above.

If instead $\mathrm{Im}(z)$ is negative, Eq. (9) is used to evaluate $\mathrm{w}(z)$ from the value of $\mathrm{w}(-z)$. The Gaussian term in Eq. (9) allows to divide this portion of the complex plane in three regions, in which the Gaussian is dominant, negligible or neither. These regions are bounded by the level curves of the complex Gaussian function $\mathrm{Re}(-z^2) = \mathrm{Im}(z)^2 - \mathrm{Re}(z)^2 = \pm g$, corresponding to $|e^{-z^2}| = g^{\pm 1}$ for some $g > 0$, and are labeled in Fig. 1 as Region D ($\mathrm{Re}(-z^2) < g$), Region E ($\mathrm{Re}(-z^2) > g$) and Region C (in between the previous two). The value of the constant g can be determined e.g. by requiring the relative error in neglecting the Gaussian term in Eq. (9) to be smaller than $\varepsilon_{\mathrm{DP}}$, and specialising this condition to $\mathrm{Im}(z) = 0$. The condition can then be written as

$$2\sqrt{\pi} e^{-\mathrm{Re}(z)^2} |z| < \varepsilon_{\mathrm{DP}} \quad \Rightarrow \quad \mathrm{Re}(z)^2 \geq \log\left(\frac{2\sqrt{\pi}\,|\mathrm{Re}(z)|}{\varepsilon_{\mathrm{DP}}}\right), \tag{15}$$

from which follows that when targeting double precision the appropriate value for g is ≈ 41.024025.

Finally, note that for purely imaginary arguments $z = ix$ with x real, the Faddeeva function equals the real-valued scaled complementary error function $\mathrm{erfcx}(x)$. For purely real arguments instead, $\mathrm{w}(x)$ is in general complex, but its real part is simply e^{-x^2}. It is therefore beneficial to specialise the implementation on the axes of the complex plane to two real-valued functions of real argument, namely $\mathrm{erfcx}(x)$ and $\mathrm{Im}(\mathrm{w})(x)$. This allows to employ faster real floating point operations, with computational advantages that propagate to the computation of other error-like functions as described in the next section.

2.4 erf and other error like-functions

Once the value of the Faddeeva function is known for some argument z, the value of the error function itself or any of the error-like functions is computed from the following

closed relations:

$$
\begin{aligned}
\mathrm{erfcx}(z) &= \mathrm{w}(iz) \\
\mathrm{erfc}(z) &= \begin{cases} e^{-z^2}\mathrm{w}(iz) & \text{if } \mathrm{Re}(z) \geq 0 \\ 2 - e^{-z^2}\mathrm{w}(-iz) & \text{if } \mathrm{Re}(z) < 0 \end{cases} \\
\mathrm{erf}(z) &= \begin{cases} 1 - e^{-z^2}\mathrm{w}(iz) & \text{if } \mathrm{Re}(z) \geq 0 \\ e^{-z^2}\mathrm{w}(-iz) - 1 & \text{if } \mathrm{Re}(z) < 0 \end{cases} \\
\mathrm{erfi}(z) &= \begin{cases} -i + ie^{z^2}\mathrm{w}(-z) & \text{if } \mathrm{Im}(z) \leq 0 \\ -ie^{z^2}\mathrm{w}(z) + i & \text{if } \mathrm{Im}(z) > 0 \end{cases} \\
\mathrm{Dawson}(z) &= \begin{cases} i\dfrac{\sqrt{\pi}}{2}\left[e^{-z^2} - \mathrm{w}(z)\right] & \text{if } \mathrm{Re}(z) \geq 0 \\ i\dfrac{\sqrt{\pi}}{2}\left[\mathrm{w}(-z) - e^{-z^2}\right] & \text{if } \mathrm{Re}(z) < 0 \end{cases}
\end{aligned}
\tag{16}
$$

Employing different formulas depending on the sign of the real part of z (or the imaginary part in the case of erfi) allows to avoid loss of accuracy due to multiplying or adding very large and small numbers. Note that the FADDEEVA PACKAGE employs the same technique.

3 Accuracy and performance results

This section collects and presents the measurements of the accuracy and speed of the algorithms presented above and implemented in the ERFLIKE library. As basis of comparison, the corresponding measurements relative to the FADDEEVA PACKAGE are employed.

3.1 Accuracy

Fig. 2 compares the accuracy of the implementation of the algorithm presented here against the ones implemented in the FADDEEVA PACKAGE over a wide region of the complex plane centered around the origin. The insets in both panels magnify the square region $(-6,6) \times (-6,6)i$. The plotted quantity is the relative error δ_{rel} (Eq. (10)), expressed in units of the IEEE double precision epsilon, $\varepsilon_{\mathrm{DP}} = 2^{-52} \approx 2.220446049250313 \cdot 10^{-16}$. Here and in the following, the reference values of $\mathrm{w}(z)$ have been obtained by evaluating the expression $\mathrm{w}(z) = e^{-z^2}\mathrm{erfc}(-iz)$ in 236-bit arithmetic (≈ 70 decimal digits) using the arbitrary-precision Python `mpmath` library [17].

Firstly one can notice that the results summarized in Fig. 2 match the behaviour expected by examining the condition number of the Faddeeva function (see Sec. 2.1): namely, double precision accuracy is attained in the angular sector $-\frac{1}{4}\pi < \arg(z) < \frac{5}{4}\pi$, while outside of it the relative error tends to be higher and grows rather rapidly as $|z|$ increases. This behaviour can be observed for both the ERFLIKE library and the FADDEEVA PACKAGE. Note that in the region $\mathrm{Im}(z) < 0 \cap |\mathrm{Im}(z) - \mathrm{Re}(z)| \gtrapprox 26$ the

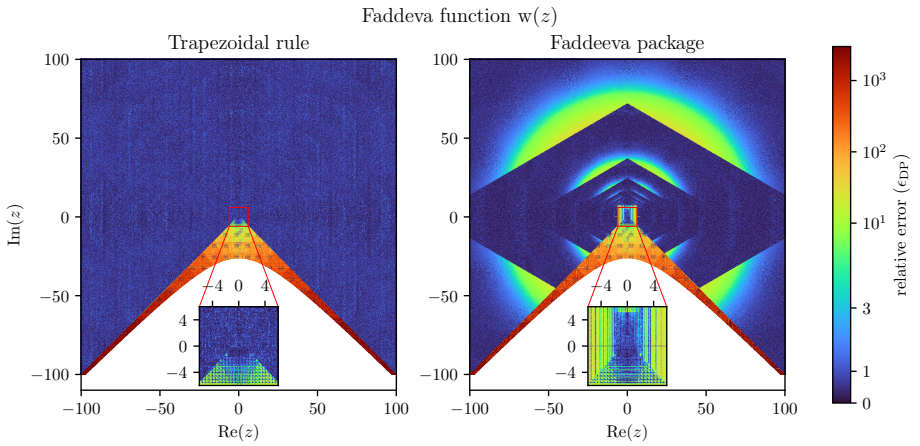

Figure 2: Relative error in the computation of the Faddeeva function $\mathrm{w}(z)$ over the complex plane in units of the double precision epsilon $\varepsilon_{\mathrm{DP}}$. The colour scale is linear up to $5\varepsilon_{\mathrm{DP}}$ and logarithmic beyond.

absolute value of the Faaddeva function overflows and the result cannot be represented in IEEE double precision, so the relative error is undefined.

Besides these general considerations, it is apparent that where the Faddeeva function is well-conditioned the relative error of the trapezoidal rule is consistently closer to $\varepsilon_{\mathrm{DP}}$ than the FADDEEVA PACKAGE. The right panel of Fig. 2 shows overlapping circle- and rhombus-shaped regions of growing size, where the accuracy of the FADDEEVA PACKAGE oscillates between $\sim 1\,\varepsilon_{\mathrm{DP}}$ and $\sim 10^2\,\varepsilon_{\mathrm{DP}}$. This behaviour seems to be a manifestation of the variable number of terms that is retained in the continued fraction representation of $w(z)$ for large $|z|$ [14], so that when the algorithm switches to a lower-order representation as $|z|$ attains larger values, significant loss of accuracy occurs. The ERFLIKE library instead essentially employs a single representation throughout the domain, and is therefore completely free from these artefacts.

Rather surprisingly, the accuracy of the FADDEEVA PACKAGE degrades significantly also in the square-shaped region centred on the origin and shown in the insets of Fig. 2. Here the FADDEEVA PACKAGE relative error is small very close to the origin, but increases to $\sim 10\,\varepsilon_{\mathrm{DP}}$ when $|\mathrm{Re}(z)| \gtrapprox 2$ or $\mathrm{Im}(z) \gtrapprox 5$. Since in many applications the region where error-like functions are evaluated most often is around the origin, relying on the FADDEEVA PACKAGE might result in a noticeable loss of accuracy. The reason for this behaviour is unclear, but the formulas employed by the FADDEEVA PACKAGE in this region appear to be more complex and have more terms [20] than the ones resulting from applying the trapezoidal rule. This might increase the occurrence of cancellation error. On the other hand the potential for cancellation error to occur seems to be much lower in the case of the ERFLIKE library, which shows a much more regular and satisfactory behaviour. Its relative error does not differ significantly

from $1\varepsilon_{\mathrm{DP}}$ in this region too, save where the Faddeeva function itself is ill-conditioned. These observation are confirmed at a more quantitative level by comparing the average relative error (computed via a simple arithmetic mean) of the two implementations over the region shown in the insets: for the ERFLIKE library $\langle\delta_{\mathrm{rel}}\rangle \approx 1.84\,\varepsilon_{\mathrm{DP}}$, while for the FADDEEVA PACKAGE $\langle\delta_{\mathrm{rel}}\rangle \approx 8.91\,\varepsilon_{\mathrm{DP}}$.

While the analysis above is focused on the Faddeeva function, the same qualitative behaviour has been observed for each of the error-like functions. This is to be expected, since both the ERFLIKE implementation and the FADDEEVA PACKAGE compute them by starting from the corresponding value of the Faddeeva function and then applying the same identities (Eqs. (16)).

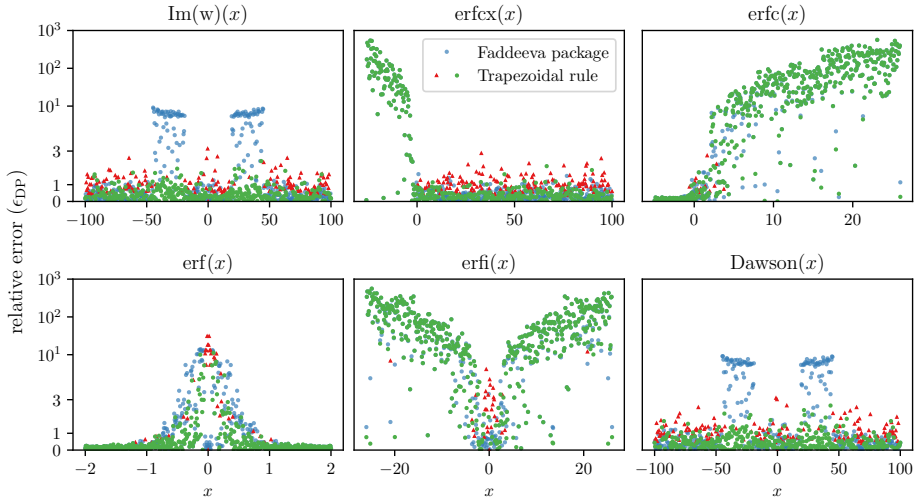

Figure 3: Relative error in the computation of the error-like functions over the real axis in units of the double precision epsilon $\varepsilon_{\mathrm{DP}}$. The ordinate scale is linear up to 5ε and logarithmic beyond. Values relative to the FADDEEVA PACKAGE are marked by blue dots. Values relative to the ERFLIKE library are denoted by green dots if their accuracy is no more than twice worse than that of the FADDEEVA PACKAGE, by red upwards-pointing triangles otherwise.

Since error-like functions are often evaluated along the real line, it is important to investigate the accuracy of the present implementation in this particular case too. This is shown in Fig. 3, in which all error-like functions are considered. Note that Im(w) and Dawson integral are proportional to each other and as such their plots are extremely similar. The latter is included only for completeness. It can be seen that in most cases the ERFLIKE library and the FADDEEVA PACKAGE behave very similarly, and although at any given point and for any given function any of the two could be more accurate than the other, there is no clear overall trend.

Two exceptions to this general observation are visible in Fig. 3. First of all, around $|x| \sim 30$ for Dawson and Im(w), the relative error of the FADDEEVA PACKAGE grows

noticeably, reaching $\sim 10\,\varepsilon_{\mathrm{DP}}$. This might be an effect related to switching from a continued fraction representation for large $|x|$ and a Chebyshev interpolation closer to the origin. Interestingly, this does not happen in the case of erfcx or erfc, which seems to suggest this is a simple oversight in the implementation rather than an intrinsic property of the algorithm. The relative error of the ERFLIKE library implementation instead remains consistently below $\sim 3\,\varepsilon_{\mathrm{DP}}$.

Secondly, in the case of the odd functions Im(w), erf, erfi and Dawson, the trapezoidal rule implementation suffers from a slight loss of accuracy close to the origin. This is not surprising since the origin is a zero of these functions, which makes them ill-conditioned in its vicinity. An effective way to mitigate accuracy loss in this cases is simply to evaluate the relevant Maclaurin series. This is indeed the algorithm implemented in the FADDEEVA PACKAGE, resulting in a better accuracy as seen in Fig. 3. While the results shown here focus on characterising the behaviour of the trapezoidal rule algorithm, the ERFLIKE library offers to the user the possibility of switching from evaluating Eq. (6) to evaluating the Maclaurin series for all real-valued error-like functions at compile time. With this precaution the relative error drops to the same values observed for the FADDEEVA PACKAGE.

3.2 Speed of evaluation

The evaluation speed of the ERFLIKE library and the FADDEEVA PACKAGE has been measured with the help of the `nanobench` library [10]. The code has been compiled using the `gcc` compiler [16], version 14.1.1, with flags `-Ofast -march native`. All tests have been run on an Intel Core i7-7700HQ CPU running at a frequency of 2.80 GHz.

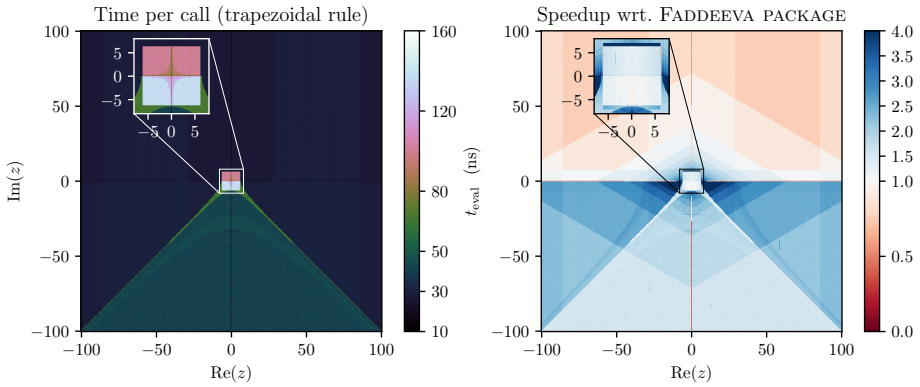

Figure 4: Left panel: time of evaluation of the Faddeeva function w(z) over the complex plane for the ERFLIKE implementation, in nanoseconds. Right panel: ratio of evaluation speed of the ERFLIKE library with respect to the FADDEEVA PACKAGE.

The performance in terms of speed of evaluation of the Faddeeva function for the ERFLIKE library over the complex plane is summarised in Fig. 4. Its left panel shows the evaluation time in nanoseconds while the right panel displays the relative speedup (or slowdown) with respect to the FADDEEVA PACKAGE. The left panel in particular shows that the behaviour of the evaluation speed of the ERFLIKE library divides the complex plane in five regions, essentially characterised by how many exponentials need to be computed for each of them. In the angular sector $-\frac{1}{4}\pi < \arg(z) < \frac{5}{4}\pi$ and far from the origin no exponential term is needed and the time of evaluation is around $\sim 30\,\text{ns}$. In regions E and C defined in Sec. 2.3.2 one exponential term appears, raising evaluation times to $\sim 50\,\text{ns}$ and $\sim 70\,\text{ns}$ respectively (recall however that in most of region E $\mathrm{w}(z)$ overflows, so high evaluation times would not be particularly concerning). In region A the presence of the pole contribution in Eq. (6), needing the evaluation of two complex exponentials, pushes evaluation times up to about $\sim 100\,\text{ns}$. Finally, the region symmetric to region A across the real axis requires one further exponential evaluation, reaching the highest evaluation times of about $\sim 150\,\text{ns}$. Note that the selection of the staggered or unstaggered version of formula Eq. (6) depending on the value of the real part of z influences the evaluation time only to a negligible level.

The right panel of Fig. 4 reveals that over large portions of the complex plane the ERFLIKE implementation offers clear advantages over the FADDEEVA PACKAGE. Firstly a significant speedup is observed for $\mathrm{Im}(z) < 0$, where the speedup can be up to ~ 4 times close to the origin. The FADDEEVA PACKAGE is clearly penalised by its reliance on a continued fraction representation with a variable number of terms, resulting in rhombus-like structures as in Fig. 2. The ERFLIKE library instead computes the value $\mathrm{w}(-z)$ and reflects it to the lower part of the complex plane by means of Eq. (9), with a clear improvement in performance. This advantage in performance is present (although not as striking) in the upper part of the complex plane too, at least for $|z| \lessapprox 25$.

Closer to the origin, in the region $(6.17, -6.17) \times (\pi/h, -\pi/h)i$, the speed of the ERFLIKE library is marginally higher than that of the FADDEEVA PACKAGE, as clearly seen in the insets of Fig. 4). In this region both algorithms require the evaluation of complex exponentials, resulting in comparable evaluation times. Finally, in the upper part of the complex plane, the FADDEEVA PACKAGE continued fraction representation edges over the ERFLIKE library in terms of speed, although generally by less than a factor of 2.

Fig. 5 focuses on the evaluation speed in the case of real arguments, where both the FADDEEVA PACKAGE and the trapezoidal rule implementations switch from real to complex arithmetic. As can be seen the FADDEEVA PACKAGE clearly performs better, being often more than twice as fast. This is particularly evident around the origin for any of the error-like functions, where the poles contribution to the trapezoidal rule require the evaluation of exponentials which severely slows down the code, up to $\sim 50\,\text{ns}$ per call in the case of erfi.

The FADDEEVA PACKAGE instead achieves evaluation times smaller than 10 ns per call for every function and most values of the argument, rising slightly close to the origin. It achieves this performance by employing only a few terms of the continued fraction representation of these functions far from the origin. Closer to the origin instead the continued fraction converges very slowly, and the FADDEEVA PACKAGE switches to

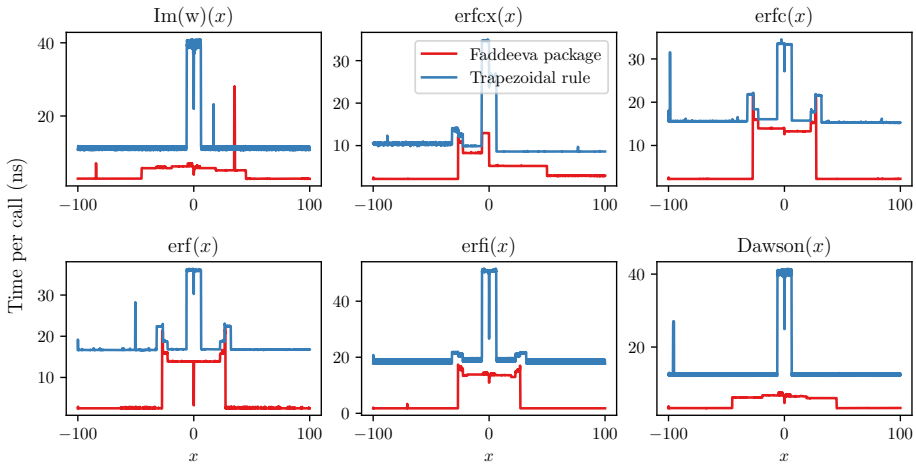

Figure 5: Evaluation time of the real-valued error-like functions, expressed in nanoseconds, for both the FADDEEVA PACKAGE and ERFLIKE library.

a piecewise fit in terms of Chebyshev polynomials, thus reducing the operation count and sidestepping the evaluation of quotients and exponentials, which instead appear prominently in Eq. (6). This explains the difference in performance.

Note however that the Chebyshev representation employed by the FADDEEVA PACKAGE has been computed *ad hoc* to maximise speed in the case of double precision, and its coefficients cannot be determined from closed formulas in more general cases. Eq. (6) on which the ERFLIKE library is based is instead applicable to any required level of accuracy, as detailed in Sec. 2.3.

Finally, the same compile-time flag mentioned at the end of Sec. 3.1 allows the user of the ERFLIKE library to switch to the asymptotic expansion Eq. (2), truncated to six terms, for both complex- and real-valued functions when the absolute value of the argument is larger than 30. This choices preserves the accuracy summarised in Fig. 2 and Fig. 3, but greatly improves evaluation times far from the origin (especially for the complex-valued functions), surpassing the performance of the FADDEEVA PACKAGE.

4 Conclusions

This work is concerned with the accurate and efficient evaluation of the complex-valued error function and related functions such as erfc, erfi and the Faddeeva function. The evaluation strategy focuses on the application of the trapezoidal rule, including pole contributions, to a suitable integral representation of the Faddeeva function. Exploiting the exponential convergence of the trapezoidal rule to the integral in question, I have derived a remarkably compact evaluation formula.

Next, considerations regarding the efficient use and implementation of this formula have been discussed, namely the optimal number of terms of the formula to retain and

the possibility to neglect some parts of it in specific regions of the complex plane without compromising accuracy. Moreover, the application of the formula to the evaluation to all error-like functions besides the Faddeeva function has been described.

I have implemented this algorithm in a C/C++ library named ERFLIKE targeting an accuracy of evaluation equal to IEEE double precision, and measured both its relative error against reference values of the error-like functions and its evaluation speed. As a basis of comparison to characterise these results and place them in the existing literature, the implementation provided by the FADDEEVA PACKAGE, which can be seen as a *de facto* standard for the evaluation of complex error functions, has been employed.

The comparison has revealed that the trapezoidal-rule based algorithm and its implementation is accurate, displaying a relative error very close to IEEE double precision level over all regions of the complex plane where the target functions are well conditioned. This behaviour is much more regular than the one of the FADDEEVA PACKAGE, which suffers from significant loss of accuracy in particular regions. In terms of speed of evaluation too the comparison is positive towards the trapezoidal-rule algorithm, which is faster than the FADDEEVA PACKAGE over most of the complex plane, although slower for real-valued arguments. Coupling this algorithm with an asymptotic expansion and Maclaurin series where appropriate further enhances its performance, making it generally the preferable choice.

In conclusion, the application of the exponentially convergent trapezoidal rule yields an evaluation algorithm for the error-like function which is superior to the methods widely employed. The ERFLIKE library which implements this algorithm is publicly available at [7, DOI:10.5281/zenodo.11261631], making it possible to reproduce the results presented here. Finally, being readily available, self-contained and similar in design to the FADDEEVA PACKAGE, it may also be widely adopted.

Conflict of interest

The author declares that he has no conflict of interest.

A Composite trapezoidal rule formula

The composite trapezoidal rule formula for the evaluation of definite integrals on the real line can be written as

$$\int_{-\infty}^{+\infty} f(x)\mathrm{d}x = h \sum_{n=-\infty}^{+\infty} f(nh) + E(h;z)\,, \tag{17}$$

where $E(h;z)$ is an error term. For generic functions the formula converges to the value of the integral to second order in h, i.e. $|E(h;z)| = O(h^2)$. It is well known however that in particular cases the error term convergences exponentially. For integrals on the real line, the conditions for exponential convergence can be stated in terms of fast decay of the integrand as $|x| \to \infty$ and analyticity in a strip of the complex plane surrounding the real line [18]. Even if $f(x)$ is not analytic but meromorphic in this strip, exponential convergence can still be recovered by taking into account the poles' contribution. [6]

has provided a clear and succinct way of deriving trapezoidal-rule formulas for a particular class of integrals and estimate the magnitude of $E(h;z)$. I reproduce his treatment here in the interest of a self-contained discussion, including the contribution from the possible poles of the integrand.

Consider a function (in general a complex-valued function of complex variable) denoted by $f(z)$, which can be represented by an integral of the form

$$f(z) = \int_{-\infty}^{+\infty} K(t;z)\, e^{-t^2}\, \mathrm{d}t\,, \tag{18}$$

where the integration is over the real axis and the function K is even with respect to t (a non-zero odd part of K would not contribute to the integral). Let h be a positive real number and assume that K is meromorphic over the strip $|\mathrm{Im}(t)| < \pi/h$.

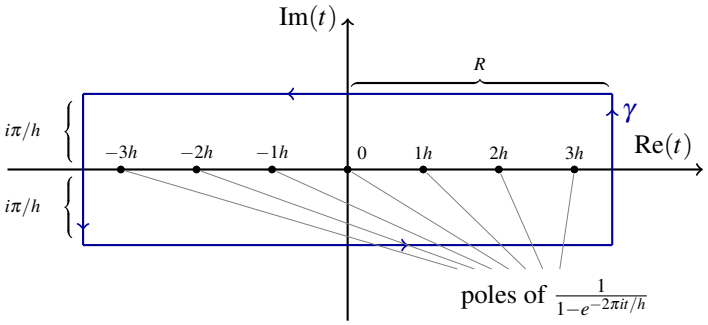

Figure 6: Contour γ, enclosing the poles of $1/(1-e^{-2\pi i t/h})$ on the real axis.

Consider the following contour integral over the rectangle γ with corners $-R-i\pi/h$, $R-i\pi/h$, $R+i\pi/h$ and $-R+i\pi/h$ (see Fig. 6)

$$C = \int_{\gamma} \frac{K(t;z)\, e^{-t^2}\, \mathrm{d}t}{1-e^{-2\pi i t/h}}\,, \tag{19}$$

where R is real and positive. The function $1/(1-e^{-2\pi i t/h})$ has simple poles at all integer multiples of h on the real axis, each with residue $h/2\pi i$. Introducing it allows for the trapezoidal rule's formula to emerge naturally from contour integration. Taking the limit $R \to +\infty$, the value of C can be computed by invoking the residue theorem. Let z_k, $k \in \mathbb{N}$ denote the poles of $K(t;z)$ lying inside the contour. Then the result can be written as

$$C = h \sum_{n=-\infty}^{+\infty} K(nh;z)\, e^{-n^2h^2} + 2\pi i \sum_k \mathrm{Res}\left[\frac{K(t;z)\, e^{-t^2}}{1-e^{-2\pi i t/h}}\right]_{t=z_k}. \tag{20}$$

The first term of Eq. (20) takes into account the contribution from the nodes of $1/(1-e^{-2\pi i t/h})$. The second term corresponds to the residues of the integrand at the poles z_k of $K(t;z)$.

On the other hand, the integral can also be expressed as

$$C = \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} \frac{K(t;z)\,e^{-t^2}\,\mathrm{d}t}{1-e^{-2\pi i t/h}} + \int_{+\infty+i\pi/h}^{-\infty+i\pi/h} \frac{K(t;z)\,e^{-t^2}\,\mathrm{d}t}{1-e^{-2\pi i t/h}}\,, \tag{21}$$

where the contribution of the vertical segments located at $\mathrm{Re}(t) = \pm R$ vanishes for $R \to +\infty$ due to the fast decay of the Gaussian term. The first term on the right hand side can be rearranged to yield:

$$\begin{aligned} C + \int_{-\infty+i\pi/h}^{+\infty+i\pi/h} \frac{K(t;z)\,e^{-t^2}\,\mathrm{d}t}{1-e^{-2\pi i t/h}} &= \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} K(t;z)\,e^{-t^2}\,\mathrm{d}t \\ &\quad + \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} \frac{K(t;z)\,e^{-t^2-2\pi i t/h}\,\mathrm{d}t}{1-e^{-2\pi i t/h}}\,. \end{aligned} \tag{22}$$

Note that this manipulation is valid as long as the denominator does not vanish, which is guaranteed over the line $\mathrm{Im}(t) = -\pi/h$ since the zeros of the denominator are located on the real axis.

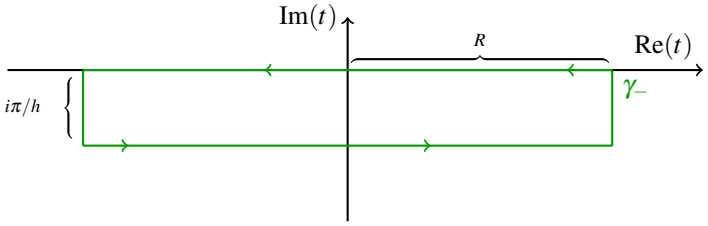

Figure 7: Contour γ_-, overlapping with the real axis.

Now consider a second contour integral of the form

$$\begin{aligned} C_- = \int_{\gamma_-} K(t;z)\,e^{-t^2}\,\mathrm{d}t &= \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} K(t;z)\,e^{-t^2}\,\mathrm{d}t \\ &\quad + \int_{+\infty}^{-\infty} K(t;z)\,e^{-t^2}\,\mathrm{d}t\,, \end{aligned} \tag{23}$$

where γ_- has corners $-\infty - i\pi/h$, $+\infty - i\pi/h$, $+\infty + i\pi/h$ and $-\infty + i\pi/h$ (see Fig. 7, where again $R \to +\infty$), and in this case too the vertical segments do not contribute to the final value. Note that the last term of this expression equals $-f(z)$ by definition. According to the residue theorem, the value C_- can also be computed as

$$C_- = 2\pi i \sum_j \mathrm{Res}\left[K(t;z))\,e^{-t^2}\right]_{t=z_j}\,, \tag{24}$$

where now z_j are the poles of $K(t;z)$ lying in the region $-\pi/h < \mathrm{Im}(t) < 0$.

Combining Eq. (23) and Eq. (24), one finds that

$$f(z) = -2\pi i \sum_{z_j} \mathrm{Res}\left[K(t;z)\,e^{-t^2}\right]_{t=z_j} + \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} K(t;z)\,e^{-t^2}\,\mathrm{d}t\,. \tag{25}$$

At this point, it can be noticed that the integral term in the right hand side of Eq. (25) is the same as the first term in the right hand side of Eq. (22), allowing $f(z)$ to be expressed as:

$$f(z) = -2\pi i \sum_{z_j} \mathrm{Res}\left[K(t;z))\, e^{-t^2}\right]_{t=z_j} + C \tag{26}$$
$$+ \int_{-\infty+i\pi/h}^{+\infty+i\pi/h} \frac{K(t;z)\, e^{-t^2}\,\mathrm{d}t}{1-e^{-2\pi i t/h}} - \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} \frac{K(t;z)\, e^{-t^2-2\pi i t/h}\,\mathrm{d}t}{1-e^{-2\pi i t/h}}\,.$$

Finally, we plug in the value of C from Eq. (20). The expression to evaluate $f(z)$ using the trapezoidal rule therefore reads

$$f(z) = h\sum_{n=-\infty}^{+\infty} K(nh;z)\, e^{-n^2t^2} + 2\pi i \sum_{z_i} \mathrm{Res}\left[\frac{K(t;z)\, e^{-t^2}}{1-e^{-2\pi i t/h}}\right]_{t=z_i} \tag{27}$$
$$-2\pi i \sum_{z_j} \mathrm{Res}\left[K(t;z)\, e^{-t^2}\right]_{t=z_j} - E(h;z)\,,$$

where the error term $E(h;z)$ is given by the two last terms of Eq. (26). Exploiting the evenness of $K(t;z)$ and e^{-t^2} with respect to t this expression can be simplified. The final formula reads

$$f(z) = hK(0;z) + 2h\sum_{n=1}^{+\infty} K(nh;z)\, e^{-n^2t^2} + 2\pi i \sum_{z_i} \mathrm{Res}\left[\frac{K(t;z)\, e^{-t^2}}{1-e^{-2\pi i t/h}}\right]_{t=z_i} \tag{28}$$
$$-2\pi i \sum_{z_j} \mathrm{Res}\left[K(t;z)\, e^{-t^2}\right]_{t=z_j} - E(h;z)\,.$$

Note that in the derivation above I have assumed for simplicity of notation that $K(t;z)$ has no poles falling on the lines $\mathrm{Im}(t) = \pm i\pi/h$ or $\mathrm{Im}(t) = 0$. Should this happen instead, and assuming these poles to be of order 1 (which is the case for the present work, and possibly the most relevant in general), their contribution would be halved with respect to the formulas stated above. Note that if such poles occur on the real axis, Eq. (18) should be interpreted as a Cauchy principal part.

The error term $E(h;z)$ of Eq. (28) reads

$$E(h;z) = + \int_{-\infty-i\pi/h}^{+\infty-i\pi/h} \frac{K(t;z)\, e^{-t^2-2\pi i t/h}\,\mathrm{d}t}{1-e^{-2\pi i t/h}} - \int_{-\infty+i\pi/h}^{+\infty+i\pi/h} \frac{K(t;z)\, e^{-t^2}\,\mathrm{d}t}{1-e^{-2\pi i t/h}}\,. \tag{29}$$

Written in this way it is not especially illuminating or useful. However it can be manipulated into a much clearer form. First, change the integration variable as $t \to -t$ in the second term (which leaves K and the Gaussian term unaffected since they are even) to find

$$E(h;z) = 2\int_{-\infty-i\pi/h}^{+\infty-i\pi/h} \frac{K(t;z)\, e^{-t^2-2\pi i t/h}}{1-e^{-2\pi i t/h}}\,\mathrm{d}t\,. \tag{30}$$

By the substitution $t = y - i\pi/h$ this is then written as

$$E(h;z) = 2\int_{-\infty}^{+\infty} \frac{K(y-i\pi/h;z)\, e^{-y^2}\, e^{-\pi^2/h^2}}{1-e^{-2\pi i y/h}\, e^{-2\pi^2/h^2}}\,\mathrm{d}y\,, \tag{31}$$

where the integration is now on the real axis. The term $e^{-2\pi^2/h^2}$ in the denominator will be negligible with respect to 1 in all cases of practical interest, so the expression simplifies to

$$E(h;z) = 2e^{-\pi^2/h^2} \int_{-\infty}^{+\infty} K(y - i\pi/h; z)\, e^{-y^2}\, \mathrm{d}y\,. \tag{32}$$

As noted by [6], as long as $K(t;z)$ does not grow too fast as $|t| \to +\infty$, the error can be approximated satisfactorily by setting $K(t;z) \approx K(0;z)$ in Eq. (32), which results in the simple but useful estimate:

$$E(h;z) = 2\sqrt{\pi} e^{-\pi^2/h^2} K(i\pi/h; z)\,. \tag{33}$$

In many applications, such as the present work, it can be beneficial to have the nodes of the trapezoidal rule on the real axis staggered with respect to 0, i.e. located at $h/2$, $3h/2$, $5h/2$, ... instead of 0, h, $2h$, ... [8]. This becomes necessary if some pole of $K(t;z)$ coincides with one of the poles of $1/(1-e^{-2\pi i t/h})$, since in this case expression Eq. (28) will become singular. This is easily achieved by considering instead of Eq. (19) the integral

$$C_{\text{staggered}} = \int_\gamma \frac{K(t;z)\, e^{-t^2}\, \mathrm{d}t}{1 + e^{-2\pi i t/h}}\,, \tag{34}$$

where the contour γ is the same as above, and the function $1/(1+e^{-2\pi i t/h})$ has simple poles at $t = (n-1/2)h$ for $n \in \mathbb{Z}$, each with residue $h/2\pi i$. It is easy to see that the calculations above can be carried on in the same fashion, resulting in the final formula

$$\begin{aligned} f(z) =& 2h \sum_{n=1}^{+\infty} K\left[(n-1/2)\,h; z\right] e^{-(n-1/2)^2 t^2} + 2\pi i \sum_{z_i} \mathrm{Res}\left[\frac{K(t;z)\, e^{-t^2}}{1+e^{-2\pi i t/h}}\right]_{t=z_i} \\ & - 2\pi i \sum_{z_j} \mathrm{Res}\left[K(t;z)\, e^{-t^2}\right]_{t=z_j} + E(h;z)\,, \end{aligned} \tag{35}$$

where the error term is still given by Eq. (32).

References

[1] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011.

[2] C. Chiarella and A. Reichel. On the evaluation of integrals related to the error function. *Math. Comp.*, 22(101):137–143, 1968.

[3] DLMF. *NIST Digital Library of Mathematical Functions*. https://dlmf.nist.gov/, Release 1.2.0 of 2024-03-15, 2024. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

[4] Henry E. Fettis, James C. Caslin, and Kenneth R. Cramer. Complex zeros of the error function and of the complementary error function. *Math. Comp.*, 27(122):401–407, 1973.

[5] Walter Gautschi. Efficient Computation of the Complex Error Function. *SIAM J. Numer. Anal.*, 7(1):187–198, March 1970.

[6] E. T. Goodwin. The evaluation of integrals of the form $\int_{-\infty}^{+\infty} f(x)e^{-x^2}dx$. *Math. Proc. Camb. Phil. Soc.*, 45(2):241–245, April 1949.

[7] F. M. Guercilena. erflike - evaluation of complex-valued error-like functions via exponentially-convergent trapezoidal rule, 2024.

[8] D. B. Hunter and T. Regan. A note on the evaluation of the complementary error function. *Math. Comp.*, 26(118):539–541, 1972.

[9] Steven G. Johnson and Massachusetts Institute of Technology. Faddeeva package, 2012.

[10] Leitner-Ankerl M. *nanobench*.

[11] Y.L. Luke. *The Special Functions and Their Approximations*. ISSN. Academic Press, New York, 1969.

[12] Y.L. Luke. *Mathematical Functions and Their Approximations*. Academic Press rapid manuscript reproduction. Academic Press, New York, 1975.

[13] F. Matta and A. Reichel. Uniform computation of the error function and other related functions. *Math. Comp.*, 25(114):339–344, 1971.

[14] G. P. M. Poppe and C. M. J. Wijers. More efficient computation of the complex error function. *ACM Trans. Math. Softw.*, 16(1):38–46, March 1990.

[15] R. Bradshaw, S. Behnel, D. S. Seljebotn, G. Ewing, et al. *The Cython compiler*.

[16] The gcc development team. *GCC, the GNU Compiler Collection*.

[17] The mpmath development team. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.3.0)*, 2023.

[18] Lloyd N. Trefethen and J. A. C. Weideman. The Exponentially Convergent Trapezoidal Rule. *SIAM Review*, 56(3):385–458, January 2014.

[19] C.G. van der Laan, N.M. Temme, and Netherlands) Centrum voor Wiskunde en Informatica (Amsterdam. *Calculation of Special Functions: The Gamma Function, the Exponential Integrals and Error-like Functions*. CWI Tract - Centrum voor Wiskunde en Informatica. CWI, Amsterdam, 1984.

[20] Mofreh R. Zaghloul and Ahmed N. Ali. Algorithm 916: Computing the Faddeyeva and Voigt Functions. *ACM Trans. Math. Softw.*, 38(2):1–22, December 2011.