A Single-Domain Implementation of the Voigt/Complex Error Function by Vectorized Interpolation

S. M. Abrarov[1,2], B. M. Quine[1,3], R. Siddiqui[1,2,3], R. K. Jagpal[2,3]

[1] Dept. Earth and Space Science and Engineering, York University, 4700 Keele St., M3J 1P3, Canada

[2] Epic College of Technology, 5670 McAdam Rd., Mississauga, L4Z 1T2, Canada

[3] Dept. Physics and Astronomy, York University, 4700 Keele St., Toronto, M3J 1P3, Canada

Correspondence: S. M. Abrarov, Dept. Earth and Space Science and Engineering, York University, 4700 Keele St., M3J 1P3, Canada.

Received: July 3, 2019 Accepted: August 3, 2019 Online Published: August 7, 2019

doi:10.5539/esr.v8n2p52 URL: https://doi.org/10.5539/esr.v8n2p52

Abstract

In this work we show how to perform a rapid computation of the Voigt/complex error over a single domain by vectorized interpolation. This approach enables us to cover the entire set of the parameters $x, y \in \mathbb{R}$ required for the HITRAN-based spectroscopic applications. The computational test reveals that within domains $x \in [0, 15] \cap y \in \left[10^{-8}, 15\right]$ and $x \in [0, 50000] \cap y \geq 10^{-8}$ our algorithmic implementation is faster in computation by factors of about 8 and 3, respectively, as compared to the fastest known C/C++ code for the Voigt/complex error function. A rapid MATLAB code is presented.

Keywords: complex error function, Faddeeva function, Voigt function, interpolation

1. Introduction

The complex error function, also commonly known as the Faddeeva function, can be defined as (Faddeeva & Terent'ev, 1961; Armstrong 1967; Gautschi, 1970; Abramowitz & Stegun, 1972)

$$w(z) = e^{-z^2}\left(1 + \frac{2i}{\sqrt{\pi}}\int\limits_0^z e^{t^2}\,dt\right), \tag{1}$$

where $z = x + iy$ is the complex argument. The real part of the complex error function (1), known as the Voigt function, can be represented as given by (Armstrong, 1967; Srivastava & Miller, 1987; Srivastava & Chen, 1992)

$$K(x, y) = \frac{1}{\sqrt{\pi}}\int\limits_0^\infty e^{-t^2/4}e^{-yt}\cos(xt)\,dt \tag{2a}$$

that is proportional to the Voigt profile describing the spectral broadening of atmospheric gas absorption or emission

$$g_V(\nu - \nu_0, \alpha_L, \alpha_D) = \frac{\sqrt{\ln 2/\pi}}{\alpha_D}K(x, y),$$

where ν is the frequency, ν_0 is the frequency of the line center, α_L and α_D are the Lorentz and Doppler half widths at half maximum (HWHM), respectively, and

$$x = \sqrt{\ln 2}\,\frac{\nu - \nu_0}{\alpha_D}, \qquad y = \sqrt{\ln 2}\,\frac{\alpha_L}{\alpha_D}.$$

The imaginary part of the complex error function (1) has no specific name. Historically, it is denoted by $L(x, y)$ and can be written in form (Srivastava & Miller, 1987; Srivastava & Chen, 1992)

$$L(x, y) = \frac{1}{\sqrt{\pi}}\int\limits_0^\infty e^{-t^2/4}e^{-yt}\sin(xt)\,dt. \tag{2b}$$

None of the integrals above are analytically integrable in closed form. Therefore, these equations must be solved numerically. There are two most important aspects that have to be taken into consideration for efficient algorithmic implementation of the integrals above in spectroscopic applications based on HITRAN database (Hill et al., 2016). Specifically, the latest versions of the HITRAN database provide spectroscopic data with 4 and more digits in floating format. In order to exclude the truncation errors that inevitably occur in any line-by-line atmospheric modeling (Berk & Hawes 2017; Pliutau & Roslyakov 2017; Siddiqui et el., 2015; Siddiqui et el. 2017, Jagpal et el., 2010; Quine & Abrarov, 2013), the algorithmic implementation of the Voigt/complex error function should provide accuracy at least as close as possible to 10^{-6}. This accuracy requirement is particularly relevant for the set of input numbers $\{x, y \in \mathbb{R}\colon\ |x| + y \le 15\}$ while it is not so much critical for the set $\{x, y \in \mathbb{R}\colon\ |x| + y > 15\}$ according to literature (Schreier, 2011; Schreier, 2018). Furthermore, the algorithm should be rapid as it may deal with many millions of input numbers z in computation for the various radiative transfer applications (Schreier, 2018; Grimm & Heng, 2015).

In order to satisfy these two criteria in computation of the integrals above, the complex plane is segmented into several domains. For example, Humlíček proposed a rapid algorithm for computation of the Voigt function (2a) based on rational functions by segmenting the complex plane into several domains such that each of them is computed by corresponding rational approximation (Humlíček, 1982). Kuntz proposed some modifications and showed efficiency of the Humlíček algorithm with 4 domains (Kuntz, 1997). The Humlíček's algorithm is rapid and provides accuracy 10^{-4}. Later, Wells developed a FORTRAN code where he succeeded to improve accuracy by an order of the magnitude by making some modifications to the original Humlíček's algorithm (Wells, 1999). Another interesting variation of the Humlíček's algorithm with improved accuracy was presented by Imai et al., 2010.

It is very desirable to reduce wherever possible the number of the domains to accelerate an algorithm since each area segmentation of the complex plane leads to run-time increase due to additional logical operations and subsequent sorting of the input numbers x and y.

In our earlier publication (Abrarov et al., 2009) we have shown how to reduce the number of domains to 2 by interpolation. In particular, we suggested two-domain scheme for rapid computation of the Voigt function that can be represented as

$$K(x, y) \approx \begin{cases} \text{interpolation}, & \dfrac{x^2}{27^2} + \dfrac{y^2}{15^2} \leqslant 1 \\ \dfrac{a_1 + b_1 x^2}{a_2 + b_2 x^2 + x^4}, & \dfrac{x^2}{27^2} + \dfrac{y^2}{15^2} > 1, \end{cases} \tag{3}$$

where (see Appendix in (Kuntz, 1997))

$$\begin{aligned} a_1 &= y/(2\sqrt{\pi}) + y^3/\sqrt{\pi} \approx 0.2820948y + 0.5641896y^3 \\ b_1 &= y/\sqrt{\pi} \approx 0.5641896y \\ a_2 &= 0.25 + y^2 + y^4 \\ b_2 &= -1 + 2y^2 \end{aligned}$$

and

$$\frac{a_1 + b_1 x^2}{a_2 + b_2 x^2 + x^4} = \operatorname{Re}\left\{\frac{iz/\sqrt{\pi}}{z^2 - 1/2}\right\}.$$

Thus, the complex plane is segmented into two domains by an ellipse centered at the origin with semi-major and semi-minor axises equal to 27 and 15, respectively (see inset in Fig. 2 from our paper (Abrarov et al., 2009)). Inside the ellipse (computationally difficult internal domain) we apply interpolation while outside the ellipse (computationally simple external domain) we apply a simple rational approximation of low order. This scheme shows high efficiency particularly when $\mathbf{x} = \{x_1, x_2, x_3, \ldots\}$ is a vector and y is a scalar. In fact, vectorized $\mathbf{x} = \{x_1, x_2, x_3, \ldots\}$ and scalar y is a quite common technique in radiative transfer applications (Lynas-Gray, 1993; Letchworth & Benner, 2007; Schreier & Kohlert, 2008).

Recently Schreier reported a two-domain scheme (see equation (12) in (Schreier, 2018))

$$w(z) = K(x, y) + iL(x, y) \approx \begin{cases} \dfrac{\sum_{k=0}^{n-1} \alpha_k z^k}{\sum_{\ell=0}^{n} \beta_\ell z^\ell}, & |x| + y \le 15 \\ \dfrac{iz/\sqrt{\pi}}{z^2 - 1/2}, & |x| + y > 15, \end{cases} \tag{4}$$

where n is assumed to be an even integer, α_k and β_ℓ are the expansion coefficients that can be readily generated by Computer Algebra System (CAS) supporting symbolic programming. It is suggested that $n = 20$ in the single quotient is sufficient for line-by-line calculations in atmospheric modeling (Schreier, 2018).

In general, for arbitrary n (even or odd) representation of the Humlíček's approximation as a single quotient can be made by using notation $\lceil \ldots \rceil$ for the ceiling function as

$$w(z) \approx \frac{1}{2} \sum_{k=1}^{n} \left(\frac{\gamma_k + i\theta_k}{z - x_k + i\delta} - \frac{\gamma_k - i\theta_k}{z + x_k + i\delta} \right) = \frac{\sum_{k=0}^{2\lceil n/2 \rceil - 1} \alpha_k z^k}{\sum_{\ell=0}^{2\lceil n/2 \rceil} \beta_\ell z^\ell}, \tag{5}$$

where

$$\gamma_k = -\frac{1}{\pi}\omega_k e^{\delta^2}\sin(2x_k\delta),$$
$$\theta_k = \frac{1}{\pi}\omega_k e^{\delta^2}\cos(2x_k\delta),$$

x_k are roots of the Hermite polynomial $H_n(x)$ of degree n that can be defined by recurrence relations (Weisstein, 2019a)

$$H_{n+1} = 2xH_n(x) - 2nH_{n-1}(x), \qquad H_0(x) = 1, \qquad H_1(x) = 2x,$$

δ is a fitting parameter that at $n = 20$ can be taken as 1.55 (Schreier, 2018) and

$$\omega_k = \frac{2^{n-1}n!\sqrt{\pi}}{n^2H_{n-1}^2(x_k)}$$

are weights of the Hermite polynomial $H_n(x)$ of degree n (Weisstein, 2019b).

It is interesting to note that if integer n is even and roots are given in ascending order such that $x_{k-1} < x_k$, then number of the summation terms in the Humlíček's approximation can be reduced by a factor of two (compare equations (27a) and (26b) from (Berk & Hawes, 2017))

$$w(z) \approx \sum_{k=1}^{n/2}\left(\frac{\gamma_k + i\theta_k}{z - x_k + i\delta} - \frac{\gamma_k - i\theta_k}{z + x_k + i\delta}\right).$$

The single quotient reformulation shown in equations (4) and (5) is interesting. However, by performing computational test we found empirically that deterioration of accuracy with decreasing y is especially inherent to the single quotient reformulation of the Humlíček's approximation (5) (deterioration of accuracy with decreasing y is a common problem in computation of the Voigt/complex error function (Armstrong, 1967; Amamou et al., 2013; Abrarov & Quine, 2011)). Although a multiple precision arithmetic may be used to resolve this problem, it needs a special package (see for example (Tsarapkina, 2014)) that may affect computational speed and makes the MATLAB code inconvenient in practical applications.

In plasma physics of rarefied gases or at low atmospheric pressure that takes place in stratosphere, mesosphere and thermoshpere of the Earth, where the Doppler broadening considerably predominates over the Lorentz broadening, the value of y dependent on the pressure and temperature may be relatively close to zero. This is particularly important as the latest versions of the HITRAN supply parameters for high temperatures almost reaching 10000 K and there is a tendency that it will be increased in future. Therefore, it would be very desirable to develop a rapid algorithm that can sustain the required accuracy for input parameter $y \geq 10^{-8}$ (Schreier, 2018).

One of the possible ways to overcome these problems is to segment the complex plane as a narrow band along x-axis and to use appropriate approximation for smaller $\mathrm{Im}[z] = y$ (see for example C/C++ code (Johnson, 2017)). However, this decelerates computation as a result of additional segmentation.

Consider a complete version of the two-domain scheme (3) that includes both, the real and imaginary parts, as follows

$$w(z) = K(x,y) + iL(x,y) \approx \begin{cases} \text{interpolation}, & \dfrac{x^2}{27^2} + \dfrac{y^2}{15^2} \leqslant 1 \\ \dfrac{iz/\sqrt{\pi}}{z^2 - 1/2}, & \dfrac{x^2}{27^2} + \dfrac{y^2}{15^2} > 1. \end{cases} \tag{6}$$

The run-time test we performed shows that for the set of input numbers $\{x, y \in \mathbb{R}\colon\ |x| + y \leq 15\}$ this two-domain scheme is faster in performance by a factor about 2 as compared to that of reported in (Schreier, 2018) (we used the MATLAB codes built on approximations (4) and (6)). This is possible to achieve since interpolation utilizes a simple cubic spline instead of a rational function of high order. Therefore, this fact strongly motivated us to develop further an algorithm based on a vectorized interpolation for rapid computation of the Voigt/complex error function with accuracy that meets the requirement for the HITRAN spectroscopic applications (Hill et al., 2016).

In this work we propose a new method of algorithmic implementation for rapid computation of the integrals (1), (2a) and (2b) by vectorized interpolation that enables us to employ just a single domain. This approach provides both, the rapid computation and accuracy that meets the requirement for the HITRAN spectroscopic applications. To the best of our knowledge, this method of computation of the Voigt/complex error function is new and has never been reported in scientific literature.

2. Algorithmic Implementation

Computation of the Voigt function (1) by interpolation with help of the lookup table was first reported in the work (Drummond & Steckner, 1985) (see also (Sparks, 1997)). However, the lookup table involved in this method requires two dimensional interpolating grid-points that need extra memory and time to handle large-size data during computation to obtain a reasonable accuracy. In contrast, the vectorized (one dimensional) approach, where the interpolating grid points are computed dynamically at $\mathbf{x} = \{x_1, x_2, x_3, \ldots\}$ and given fixed value y (Abrarov et al., 2009) (rather than picked up from the lookup table (Drummond & Steckner, 1985; Sparks, 1997)) is considerably advantageous in interpolation due to significantly smaller quantity of the interpolating grid-points stored in computer memory.

Let us consider two-domain scheme shown by equation (6) more closely. We note that semi-major and semi-minor axises may be chosen arbitrarily depending on algorithmic implementations (see for example (Poppe & Wijers 1990a; Poppe & Wijers 1990b)). Since semi-major and semi-minor axises may be flexible in magnitude, we may suggest to increase them to cover the entire set $x, y \in \mathbb{R}$ for the HITRAN applications such that we could exclude completely the rational approximation of low order in equation (6) from the consideration.

Previously it was reported that the HITRAN spectroscopic database requires a domain $0 \le |x| < 40000$ and $10^{-4} < y < 10^2$ (Wells, 1999). Consequently, within the I-st quadrant a single domain should be large enough to cover rectangular area 40000×100. Furthermore, this area should be extended by factor of four if we want to include all 4 quadrants. However, as it has been mentioned earlier the inclusion of the all small values y into consideration would be preferable for practical applications to account for low pressure and high temperature of the HITRAN gases. In our approach, the single domain represents an area $|x| \leqslant 50000$ without a boundary for the parameter y since it is a scalar. Formally stating, this single domain is inclosed by a reshaped ellipse in such a way that

$$\frac{x^2}{50000^2} + \lim_{n\to\infty} \frac{y^2}{n^2} \leqslant 1.$$

In our work (Abrarov, et al., 2009) we noted that the interpolating grid-points may not be necessarily spaced equidistantly. This provides a significant advantage since we can minimize the number of the interpolating grid-points by putting them denser in more curved subintervals and sparser in more linear subintervals. Particularly, we separated interval $[-50000, 50000]$ along x-axis into subintervals as it is shown in the Table 1. Initially each subinterval contains a number of the interpolating grid-points multiple to 100 accumulating 1600 interpolating grid-points in total (see command lines below <function MF = mainF(x,y,opt)> in the MATLAB code shown in Appendix A). However, after exclusion of repeating values the total number of the interpolating grid-points decreases from 1600 to $197 + 398 + 4 \times 198 + 200 = 1587$ (see Table 1). This small quantity of the interpolating grid-points needs a very minor amount of computer memory. Therefore, it can be easily handled by practically any modern computer. We cannot infer that such a distribution provides the smallest number of the interpolating grid-points. Perhaps, this number of the interpolating grid-points can be reduced significantly by further optimization.

Table 1. Number of interpolating grid-points (IGP) in subintervals

Subintervals	**Number of IGP**
$(-2.5, 2.5)$	$1 \times 197 = 197$
$(-5.5, -2.5] \cup [2.5, 5.5)$	$2 \times 199 = 398$
$(-15, -5.5] \cup [5.5, 15)$	$2 \times 99 = 198$
$(-100, -15] \cup [15, 100)$	$2 \times 99 = 198$
$(-1000, -100] \cup [100, 1000)$	$2 \times 99 = 198$
$(-10000, -1000] \cup [1000, 10000)$	$2 \times 99 = 198$
$[-50000, -10000] \cup [10000, 50000]$	$2 \times 100 = 200$

If hypothetically there could be some input values $|x|$ greater than 50000, then it is very easy to extend the region along x-axis if required. For example, it is sufficient to include only 300 additional interpolating grid-points to extend the range for $|x|$ up to 100000; it is not necessary to include many interpolating grid-points since the functions $K(x, y)$ and $L(x, y)$ become nearly linear as $|x|$ increases.

For proper interpolation the accuracy of the computation should be two orders of the magnitude better than 10^{-6}. Generally, any highly accurate algorithm can be used to compute 1587 interpolating grid-points. We applied the highly accurate MATLAB function fadsamp.m shown in our recent paper (Abrarov et al. 2018a) as a subroutine for this purpose (see a brief description of this method in Appendix B). Alternatively, a MATLAB function file that is highly accurate and suitable for computation of the interpolating grid-points can be found in (Abrarov & Quine, 2011) (these two codes can also be downloaded from the Matlab Central websites (Matlab Central file ID #: 66752) and (Matlab Central file ID #: 47801), respectively). Highly accurate C/C++ implementation by Johnson (Johnson, 2017) with MEX plugins for MATLAB (Matlab Central file ID #: 38787) can also be used for a subroutine that can be invoked from the MATLAB environment to compute these 1587 interpolating grid-points.

It should be noted that this methodology can also be generalized further to other functions (including spectral line profiles). For example, our preliminary results demonstrate that this methodology is also applicable for rapid and quite accurate computation of the Spectrally Integrated Voigt Function (SIVF) that enables us to perform line-by-line atmospheric modeling at reduced spectral resolution (Quine & Abrarov, 2013).

3. Error Analysis and Run-Time Test

In order to perform error analysis define the relative errors for the real

$$\Delta_{\mathrm{Re}} = \left|\frac{K_{\mathrm{ref.}}(x,y) - K(x,y)}{K_{\mathrm{ref.}}(x,y)}\right|$$

and imaginary parts

$$\Delta_{\mathrm{Im}} = \left|\frac{L_{\mathrm{ref.}}(x,y) - L(x,y)}{L_{\mathrm{ref.}}(x,y)}\right|$$

of the complex error function (1), where $K_{\mathrm{ref.}}(x,y)$ and $L_{\mathrm{ref.}}(x,y)$ are the highly accurate reference values that can be readily obtained by using the CAS.

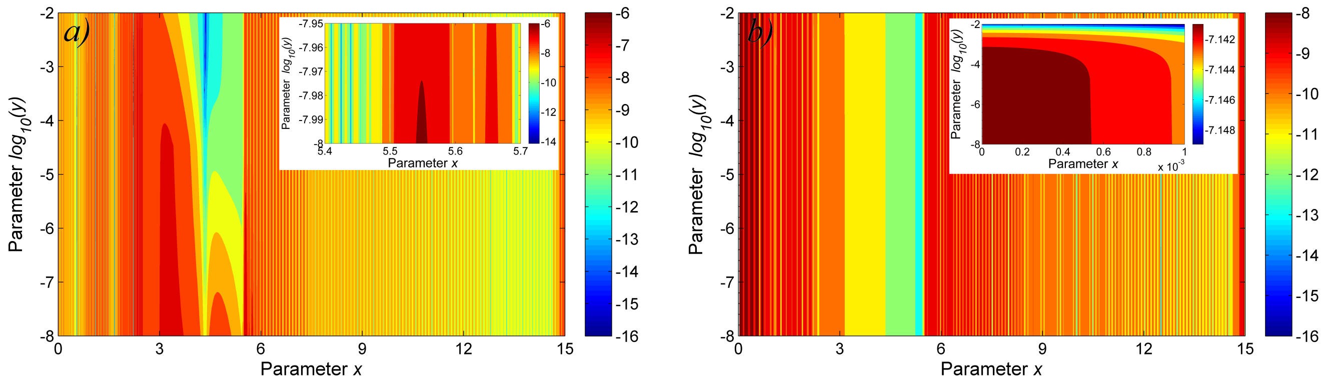

Figure 1. Logarithms of relative error for the real a) and imaginary b) parts of the complex error function over the area $x \in [0, 15] \cap y \in \left[10^{-8}, 10^{-2}\right]$ computed by vectorized interpolation. Insets show the subareas with worst accuracies for the real a) and imaginary b) parts

Figures 1a and 1b show the relative errors for the real and imaginary parts over the area $x \in [0, 15]$ and $y \in \left[10^{-8}, 10^{-2}\right]$ computed by vectorized interpolation. The largest relative errors over this area for the real and imaginary parts are 1.0589×10^{-6} and 7.236×10^{-8}, respectively. Thus, our algorithm satisfies the accuracy criterion 10^{-6} for the required range $y \geq 10^{-8}$. In comparison, the computational test we perform reveals that approximation (4) sustains the required accuracy only at $y \geq 10^{-6}$.

Figures 2a and 2b show the relative errors for the real and imaginary parts over the area $x \in [0, 15]$ and $y \in \left[10^{-2}, 15\right]$. The largest relative errors over this area for the real and imaginary parts are 2.7766×10^{-7} and 7.0619×10^{-8}, respectively.

Thus, from Figs. 1a, 1b and 2a, 2b we can conclude that our algorithm satisfies accuracy requirement for the HITRAN-based applications and effectively resolves the problem that is inherent to the single quotient shown in equations (4) and (5) as well as many other approximations (Armstrong, 1967; Amamou et al., 2013).

In order to obtain most objective results for the run-time test we used an independently written code for the Voigt/complex error function. Specifically, the run-time test has been performed by comparing our MATLAB code shown in Appendix A with C/C++ implementation that was developed by Johnson (Johnson, 2017; Matlab Central file ID #: 38787). This implementation is known to be the fastest C/C++ program. It represents a modified Algorithm 680 (Poppe & Wijers 1990a; Poppe & Wijers, 1990b) with inclusion of the Salzer's approximations (Abrarov & Quine, 2018b) (see also (Salzer, 1951)). As the computation complexity prevails at smaller values of the parameter y, we imply that it is close to zero, say

$y = 10^{-5}$. The detailed description of how to run the programs and perform the time execution test is shown in Appendix C.

The run-time test shows that for 10 million input values within the most important area such that $\{x, y \in \mathbb{R}\colon |x + iy| \le 15\}$, the MATLAB code is almost 8 times faster than the C/C++ implementation while for the entire domain that is required for the HITRAN spectroscopic applications the MATLAB code is faster than the C/C++ implementation by a factor about 3.

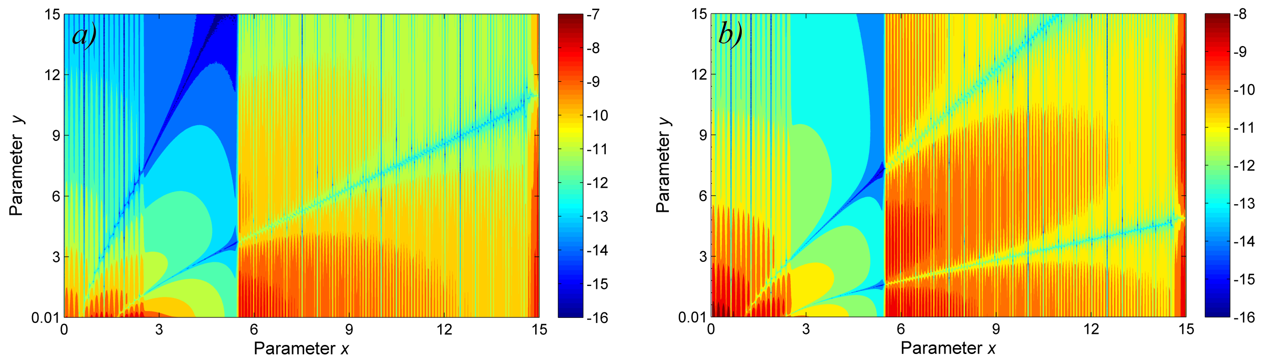

Figure 2. Logarithms of relative error for the real a) and imaginary b) parts of the complex error function over the area $x \in [0, 15] \cap y \in \left[10^{-2}, 15\right]$ computed by vectorized interpolation

The MATLAB is one of the fastest scientific languages in computation. However, it is generally slower than C/C++. The more rapid computation has been achieved because of two main reasons. The function Faddeeva.cc is unnecessarily complicated as it utilizes a large number of domains that due to multiple logical operations and sorting of the input numbers x and y decelerate computation. Furthermore, our approach *de facto* performs 1587 actual computations only as the remaining is just an interpolation. By choosing different options for interpolation, we found experimentally that the MATLAB built-in method 'spline' provides the best performance. All these results can be readily confirmed by running the codes provided in Appendices A and C.

4. Conclusion

We propose a new single-domain vectorized interpolation method for rapid computation of the Voigt/complex error function (1) that enables us to achieve required accuracy for the HITRAN-based spectroscopic applications. The computational test we performed reveals that within intervals $x \in [0, 15] \cap y \in \left[10^{-8}, 15\right]$ and $x \in [0, 50000] \cap y \ge 10^{-8}$ our algorithmic implementation is faster in computation by factors of about 8 and 3, respectively, as compared to the fastest known C/C++ code for the Voigt/complex error function.

Acknowledgments

This work is supported by National Research Council Canada, Thoth Technology Inc., York University, Epic College of Technology and Epic Climate Green (ECG) Inc. The authors wish to thank principal developer of the MODTRAN Dr. Alexander Berk for constructive discussions.

References

Abramowitz, M., & Stegun, I. A. (1972). *Error function and Fresnel integrals. Handbook of mathematical functions with formulas, graphs, and mathematical tables*. 9th ed. New York 1972, 297-309.

Abrarov, S. M., Quine, B. M., & Jagpal, R. K. (2009). A simple interpolating algorithm for the rapid and accurate calculation of the Voigt function. *Journal of Quantitative Spectroscopy and Radiative Transfer, 110*(6-7), 376-383. https://doi.org/10.1016/j.jqsrt.2009.01.003

Abrarov, S. M., & Quine, B. M. (2011). Efficient algorithmic implementation of the Voigt/complex error function based on exponential series approximation. *Applied Mathematics and Computation, 218*(5), 1894-1902. https://doi.org/10.1016/j.amc.2011.06.072

Abrarov, S. M., & Quine, B. M. (2015a). Sampling by incomplete cosine expansion of the sinc function: Application to the Voigt/complex error function. *Applied Mathematics and Computation, 258*, 425-435. https://doi.org/10.1016/j.amc.2015.01.072

Abrarov, S. M., & Quine, B. M. (2015b). A rational approximation for efficient computation of the Voigt function in quantitative spectroscopy. *Journal of Mathematics Research, 7*(2), 163-174. https://doi.org/10.5539/jmr.v7n2p163

Abrarov, S. M., Quine, B. M., & Jagpal, R. K. (2018a). A sampling-based approximation of the complex error function and its implementation without poles. *Applied Numerical Mathematics, 129*, 181-191. https://doi.org/10.1016/j.apnum.2018.03.009

Abrarov, S. M., & Quine, B. M. (2018b). A rational approximation of the Dawson's integral for efficient computation of the complex error function. *Applied Mathematics and Computation, 321*, 526-543. https://doi.org/10.1016/j.amc.2017.10.032

Amamou, H., Ferhat, B., & Bois, A. (2013). Calculation of the Voigt Function in the region of very small values of the parameter *a* where the calculation is notoriously difficult. *American Journal of Analytical Chemistry, 4*(12), 725-731. https://doi.org/10.4236/ajac.2013.412087

Armstrong, B. H. (1967). Spectrum line profiles: The Voigt function. *Journal of Quantitative Spectroscopy and Radiative Transfer, 7*(1), 61-88. https://doi.org/10.1016/0022-4073(67)90057-X

Berk, A., & Hawes, F. (2017). Validation of MODTRAN®6 and its line-by-line algorithm. *Journal of Quantitative Spectroscopy and Radiative Transfer, 203*, 542-556. https://doi.org/10.1016/j.jqsrt.2017.03.004

Drummond, J. R., & Steckner, M. (1985). Voigt-function evaluation using a two-dimensional interpolation scheme. *Journal of Quantitative Spectroscopy and Radiative Transfer, 34*(6), 517-521. https://doi.org/10.1016/0022-4073(85)90145-1

Faddeyeva, V. N., & Terent'ev, N. M. (1961). *Tables of the probability integral* $w(z) = e^{-z^2}\left(1 + \frac{2i}{\sqrt{\pi}}\int_0^z e^{t^2}dt\right)$ *for complex argument*. Pergamon Press, Oxford, 1961.

Gautschi, W. (1970). Efficient computation of the complex error function. *SIAM Journal of Numerical Analysis, 7*(1), 187-198. https://doi.org/10.1137/0707012

Gearhart, W. B., & Shultz, H. S. (1990). The function $\sin\frac{x}{x}$. *The College Mathematics Journal, 21*(2), 90-99. http://dx.doi.org/10.1080/07468342.1990.11973290

Grimm, S. L., & Heng, K. (2015). HELIOS-K: An ultrafast, open-source opacity calculator for radiative transfer. *The Astrophysical Journal*, *808*(182). https://doi.org/10.1088/0004-637X/808/2/182

Hill, C., Gordon, I. E., Kochanov, R. V., Barrett, L., Wilzewski, J. S., & Rothman, L. S. (2016). HITRAN*online*: An online interface and the flexible representation of spectroscopic data in the HITRAN database. *Journal of Quantitative Spectroscopy and Radiative Transfer, 177*, 4-14. https://doi.org/10.1016/j.jqsrt.2015.12.012

Humlíček, J. (1982). Optimized computation of the Voigt and complex probability functions. *Journal of Quantitative Spectroscopy and Radiative Transfer, 27*(4), 437-444. https://doi.org/10.1016/0022-4073(82)90078-4

Imai, K., Suzuki, M., & Takahashi, C. (2010). Evaluation of Voigt algorithms for the ISS/JEM/SMILES L2 data processing system. *Advances in Space Research, 45*(5), 669-675. https://doi.org/10.1016/j.asr.2009.11.005

Jagpal, R. K., Quine, B. M., Chesser, H., Abrarov, S., & Lee, R. (2010). Calibration and in-orbit performance of the Argus 1000 spectrometer - the Canadian pollution monitor. *Journal of Applied Remote Sensing, 4*(1), 049501. https://doi.org/10.1117/1.3302405

Johnson, S. G. (2017). Faddeeva package. Retreived from http://ab-initio.mit.edu/wiki/index.php/Faddeeva_Package

Kuntz, M. (1997). A new implementation of the Humlicek algorithm for calculation of the Voigt profile function. *Journal of Quantitative Spectroscopy and Radiative Transfer, 51*(6), 819-824. https://doi.org/10.1016/S0022-4073(96)00162-8

Letchworth, K. L., & Benner, D. C. (2007). Rapid and accurate calculation of the Voigt function. *Journal of Quantitative Spectroscopy and Radiative Transfer, 107*(1), 173-192. https://doi.org/10.1016/j.jqsrt.2007.01.052

Lynas-Gray, A. E. (1993). VOIGTL – a fast subroutine for Voigt function evaluation on vector processors. *Computer Physics Communications, 75*(1-2), 135-142. https://doi.org/10.1016/0010-4655(93)90171-8

Matlab Central, file ID #: 47801. Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/47801-the-voigt-complex-error-function-second-version

Matlab Central, file ID #: 66752. Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/66752-a-sampling-based-algorithm-for-the-voigt-complex-error-function

Matlab Central, file ID #: 38787. Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/38787-faddeeva-package-complex-error-functions

Pliutau D., & Roslyakov, K. (2017). Bytran -|- spectral calculations for portable devices using the HITRAN database. *Earth Science Informatics, 10*(3), 395-404. https://doi.org/10.1007/s12145-017-0288-4

Poppe, G. P. M., & Wijers, C. M. J. (1990). More efficient computation of the complex error function. *ACM Transactions on Mathematical Software (TOMS), 16*(1), 38-46. https://doi.org/10.1145/77626.77629

Poppe, G. P. M., & Wijers, C. M. J. (1990). Algorithm 680: evaluation of the complex error function. *ACM Transactions on Mathematical Software (TOMS), 16*(1), 47. https://doi.org/10.1145/77626.77630

Quine B. M., & Abrarov, S. M. (2013). Application of the spectrally integrated Voigt function to line-by-line radiative transfer modelling. *Journal of Quantative Spectroscopy and Radiative Transfer, 127*, 37-48. https://dx.doi.org/10.1016/j.jqsrt.2013.04.020

Salzer, H. E. (1951). Formulas for calculating the error function of a complex variable. *Mathematical Tables and Other Aids to Computation 5*(34), 67-70. https://doi.org/10.2307/2002163

Schreier, F., & Kohlert, D. (2008). Optimized implementations of rational approximations – a case study on the Voigt and complex error function. *Computer Physics Communications, 179*(7), 457-465. https://doi.org/10.1016/j.cpc.2008.04.012

Schreier, F. (2011). Optimized implementations of rational approximations for the Voigt and complex error function. *Journal of Quantative Spectroscopy and Radiative Transfer, 112*(6), 1010-1025. https://doi.org/10.1016/j.jqsrt.2010.12.010

Schreier, F. (2018). The Voigt and complex error function: Humlíček's rational approximation generalized. *Monthly Notices of The Royal Astronomical Society, 479*, 3068-3075. https://doi.org/10.1093/mnras/sty1680

Siddiqui, R., Jagpal, R., Salem N. A., & Quine, B. M. (2015). Classification of cloud scenes by Argus spectral data. *International Journal of Space Science and Engineering, 3*(4), 295-311. https://doi.org/10.1504/IJSPACESE.2015.075911

Siddiqui, R., Jagpal, R., & Quine, B. M. (2017). Short wave upwelling radiative flux (SWupRF) within near infrared (NIR) wavelength bands of O_2, H_2O, CO_2, and CH_4 by Argus 1000 along with GENSPECT line by line radiative transfer model. *Canadian Journal of Remote Sensing, 43*(4), 330-344. https://doi.org/10.1080/07038992.2017.1346467

Sparks, L. (1997). Efficient line-by-line calculation of absorption coefficients to high numerical accuracy. *Journal of Quantitative Spectroscopy and Radiative Transfer, 57*(5), 631-650. https://doi.org/10.1016/S0022-4073(96)00154-9

Srivastava, H. M., & Miller, E. A. (1987). A unified presentation of the Voigt functions. *Astrophysics and Space Science, 135*(1), 111-118. https://doi.org/10.1007/bf00644466

Srivastava, H. M., & Chen, M. P. (1992). Some unified presentations of the Voigt functions. *Astrophysics and Space Science, 192*(1), 63-74. https://doi.org/10.1007/BF00653260

Tsarapkina, D., & Jeffrey, D. J. (2014). Exploring rounding errors in Matlab using extended precision. *Procedia Computer Science, 29*, 1423-1432. https://doi.org/10.1016/j.procs.2014.05.129

Wells, R. J. (1999). Rapid approximation to the Voigt/Faddeeva function and its derivatives. *Journal of Quantitative Spectroscopy and Radiative Transfer, 62*(1), 29-48. https://doi.org/10.1016/S0022-4073(97)00231-8

Weisstein, E. W. (2019a). Hermite polynomial. Retrieved from http://mathworld.wolfram.com/HermitePolynomial.html

Weisstein, E. W. (2019b). Hermite–Gauss quadrature. Retrieved from http://mathworld.wolfram.com/Hermite-GaussQuadrature.html

Weisstein, E. W. (2019c). Sinc function. Retrieved from http://mathworld.wolfram.com/SincFunction.html

Appendix A

```
function vecFF = vecfadf(x,y,opt)

% This function file computes the Voigt/complex error function, also known
% as the Faddeeva function, providing rapid computation at required
% accuracy for the HITRAN-based radiative transfer application.
%
% SYNOPSIS:
%          x   - row or column vector
%          y   - scalar
%          opt - option for the real and imaginary parts
%
% The code is written by Sanjar M. Abrarov, Brendan M. Quine, Rehan
% Siddiqui and Rajinder K. Jagpal, York University, Canada, May 2019.

bound = 5*1e4; % default bound to cover the HITRAN domain
num = 1e2; % common number for grid-points in interpolation

if max(size(y)) ~= 1
    disp('Parameter y must be a scalar!')
    return
elseif ~isvector(x)
    disp('Parameter x is not a vector!')
end

if max(abs(x)) > bound || abs(y) < 1e-8
    disp('x or y is beyond HITRAN range! Computation is terminated.')
    return
end

if nargin == 2
    opt = 3;
    disp('Default value opt = 3 is assigned.')
end

if opt ~= 1 && opt ~= 2 && opt ~= 3
    disp(['Wrong parameter opt = ',num2str(opt),'! Use either 1, 2 or 3.'])
    return
end

if y >= 0
    vecFF = mainF(x,y,opt); % upper half-plane
else
    vecFF = mainF(x,-y,opt); % lower half-plane
    vecFF = conj(2*exp(-(x + 1i*y).^2) - vecFF);
end

    function MF = mainF(x,y,opt)

        % Forming non-equidistantly spaced interpolating grid-points (IGP)
        IGP = linspace(0,2.5,num);
        IGP = [IGP,linspace(2.5,5.5,100 + num)]; % 100 more grid-points
        IGP = [IGP,linspace(5.5,15,num)];
        IGP = [IGP,linspace(15,100,num)];
        IGP = [IGP,linspace(100,1000,num)];
        IGP = [IGP,linspace(1000,10000,num)];
        IGP = [IGP,linspace(10000,bound,num)];
        IGP = [-(flip(IGP)),IGP];
        IGP = unique(IGP); % exclude repeated values

        MF = interp1(IGP,fadsamp(IGP + 1i*y),x,'spline'); % call ...
        % external function <fadfsamp.m>. This MATAB function file is ...
        % provided in paper [Abrarov, Quine & Jagpal, Appl. Num. Math., ...
        % 129 (2018) 181-191].
        % URL: https://doi.org/10.1016/j.apnum.2018.03.009

        switch opt
            case 1
                MF = real(MF);
            case 2
            MF = imag(MF);
        end
    end
end
```

Appendix B

In our publications (Quine & Abrarov, 2013; Abrarov & Quine 2015a) we have introduced the following product-to-sum identity

$$\prod_{m=1}^{k}\cos\left(\frac{t}{2^m}\right)=\frac{1}{2^{k-1}}\sum_{m=1}^{2^{k-1}}\cos\left(\frac{m-1/2}{2^{k-1}}t\right),\qquad \forall k\geq 1$$

and since (Gearhart & Shultz, 1990; Weisstein, 2019c)

$$\operatorname{sinc}(t)=\prod_{m=1}^{\infty}\cos\left(\frac{t}{2^m}\right)$$

from this product-to-sum identity it immediately follows that the sinc function can be expanded as a sum of cosines

$$\operatorname{sinc}(t)=\lim_{k\to\infty}\frac{1}{k}\sum_{m=1}^{k}\cos\left(\frac{m-1/2}{k}t\right)$$

or

$$\operatorname{sinc}(t)\approx\frac{1}{k}\sum_{m=1}^{k}\cos\left(\frac{m-1/2}{k}t\right),\qquad k>>1. \tag{A.1}$$

Using a new method of sampling based on incomplete cosine expansion of the sinc function (A.1) we can obtain (Abrarov & Quine, 2015a) (see also (Abrarov & Quine, 2015b) and cited literature in context therein)

$$\begin{aligned} w(z)&\approx\Omega(z+i\varsigma/2)\\ \Rightarrow\Omega(z)&\triangleq\sum_{m=1}^{M}\frac{A_m+B_m z}{C_m^2-z^2},\end{aligned} \tag{A.2}$$

where $N=23$, $M=23$, $h=0.25$, $\varsigma=2.75$ and the expansion coefficients

$$A_m=\frac{\sqrt{\pi}\,(m-1/2)}{2M^2h}\sum_{n=-N}^{N}e^{\varsigma^2/4-n^2h^2}\sin\left(\frac{\pi\,(m-1/2)\,(nh+\varsigma/2)}{Mh}\right),$$

$$B_m=-\frac{i}{M\sqrt{\pi}}\sum_{n=-N}^{N}e^{\varsigma^2/4-n^2h^2}\cos\left(\frac{\pi\,(m-1/2)\,(nh+\varsigma/2)}{Mh}\right)$$

and

$$C_m=\frac{\pi\,(m-1/2)}{2Mh}.$$

Similar to the Humlíček's approximation (5), the series expansion (A.2) also deteriorates in accuracy with decreasing y. We have shown how to overcome this problem by transformation of equation (A.2) into following form

$$w(z)\approx e^{-z^2}+z\sum_{m=1}^{M+2}\frac{\kappa_m-\lambda_m z^2}{\mu_m-\nu_m z^2+z^4}, \tag{A.3}$$

where

$$\kappa_m=B_m\left[C_m^2-\left(\frac{\varsigma^2}{2}\right)^2\right]+iA_m\varsigma=B_m\left[\left(\frac{\pi\,(m-1/2)}{2Mh}\right)^2-\left(\frac{\varsigma}{2}\right)^2\right]+iA_m\varsigma,$$

$$\lambda_m=B_m,$$

$$\mu_m=C_m^4+\frac{C_m^2\varsigma^2}{2}+\frac{\varsigma^4}{16}=\left[\left(\frac{\pi\,(m-1/2)}{2Mh}\right)^2+\left(\frac{\varsigma}{2}\right)^2\right]^2$$

and

$$\nu_m = 2C_m^2 - \frac{\varsigma^2}{2} = 2\left(\frac{\pi\,(m-1/2)}{2Mh}\right)^2 - \frac{\varsigma^2}{2}.$$

The third equation that is used in the function file fadsamp.m is the Laplace continued fraction (Abramowitz & Stegun, 1972; Gautschi, 1970; Poppe & Wijers 1990a) given by

$$w\,(z) \approx \cfrac{\left(i/\sqrt{\pi}\right)}{z - \cfrac{1/2}{z - \cfrac{1}{z - \cfrac{3/2}{z - \cfrac{2}{z - \cfrac{5/2}{z - \cfrac{3}{z - \cfrac{7/2}{z - \cfrac{4}{z - \cfrac{9/2}{z - \cfrac{5}{z - \cfrac{11/2}{z}}}}}}}}}}}}, \tag{A.4}$$

Equations (A.2) and (A.3) cover the domains

$$\{x, y \in \mathbb{R}\colon |x + iy| \le 8\} \setminus \{y < 0.05\,|x|\}$$

and

$$\{x, y \in \mathbb{R}\colon |x + iy| \le 8\} \cap \{y < 0.05\,|x|\},$$

respectively, while equation (A.3) covers the domain $\{x, y \in \mathbb{R}\colon |x + iy| > 8\}$ (see Fig. 1 in (Abrarov et al., 2018a)). This three-domain scheme excludes all poles in computation and provides largest relative error $\sim 10^{-13}$ only.

Appendix C

The C/C++ function file Faddeeva.cc and its header Faddeeva.hh can be downloaded from the website [20]. To run the program the following lines can be added to the end of the Faddeeva.cc function file

```
/**
These command lines can be added at the end of the source
file 'Faddeeva.cc'.
*/

#ifdef __cplusplus
# include <cstdio>
# include <cstdlib>
# include <iostream>
# include <iomanip>

#else
# include <stdio.h>
#endif

/**
The function fRand(minNum, maxNum) generates a random number
within the range from 'minNum' to 'maxNum'.
*/

double fRand(double minNum, double maxNum){

    double fMin = minNum, fMax = maxNum;
    double f = (double)rand() / RAND_MAX;

return fMin + f * (fMax - fMin);
};

int main(void){

double(epsVal) = 1E-6; // epsilon value for accuracy

    for(int k = 0; k < 1e7; k++){ // execute the computation ...
    // 10 million times with random numbers 0 < x < 15 and positive y << 1.

        FADDEEVA(w)(C(fRand(0,15),1E-5),epsVal);
    };
return 0;
};
```

As we can see from the body of the function <int main(void)>, this program computes 10 million random digits of x at fixed value of $y = 10^{-5}$. The epsilon value that determines accuracy of computation is taken to be 10^{-6}. The execution time on a typical laptop computer (we used Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz, 8GB RAM) takes about 14 seconds.

It is interesting to note that with <double(espVal) = 1E-12;> the executions time increases to 17 seconds. As a subsequent step, at <double(espVal) = 0;> we should expect the highest accuracy and, therefore, the longest execution time. Instead, however, the program becomes significantly faster and it takes about 8.5 seconds to compute all 10 million digits. This unexpected behavior of the C/C++ implementation suggests that the program, most likely, does not guarantee the prescribed accuracy for all input numbers x and y at <double(espVal) = 0;>. Therefore, we did not take this option as a reference.

The following is the MATLAB code that also computes 10 million random numbers $x \in [0, 15]$ at fixed $y = 10^{-5}$. The execution time is about 1.8 seconds only. As we can see, the MATLAB code based on interpolation is almost 8 times faster than the C/C++ implementation.

```
% ******************************************************************
x = 15*rand(1e7,1); % this generates 10 million random numbers ...
                    % in the interval 0 < x < 15
y = 1e-5; % y is a scalar
tic;vecfadf(x,y,3);toc % this shows the run-time
% ******************************************************************
```

In order to compare the execution times for the interval $x \in [0, 50000]$, we can simply replace the command lines above in the C/C++ and MATLAB codes as

```
FADDEEVA(w)(C(fRand(0,50000),1E-5),epsVal);
```

and

```
x = 50000*rand(1e7,1); % this generates 10 million random numbers ...
                       % in the interval 0 < x < 50000
```

respectively. The execution times for the C/C++ and MATLAB implementations for this case become about 3 and about 1 seconds, respectively. Therefore, the MATLAB code is nearly 3 times faster.

Execution times can be decreased by more than an order of the magnitude on a more powerful computer of the latest generation. However, we should anticipate that these ratios 8 and 3 will remain same.

Copyrights