

The Voigt and complex error function: Humlíček’s rational approximation generalized

Franz Schreier [★]

DLR — Deutsches Zentrum für Luft- und Raumfahrt, Institut für Methodik der Fernerkundung, 82234 Oberpfaffenhofen, Germany

Accepted 2018 June 20. Received 2018 June 18; in original form 2018 May 2

ABSTRACT

Accurate yet efficient computation of the Voigt and complex error function is a challenge since decades in astrophysics and other areas of physics. Rational approximations have attracted considerable attention and are used in many codes, often in combination with other techniques. The 12-term code “`cpf12`” of Humlíček (1979) achieves an accuracy of five to six significant digits throughout the entire complex plane. Here we generalize this algorithm to a larger (even) number of terms. The $n = 16$ approximation has a relative accuracy better than 10^{-5} for almost the entire complex plane except for very small imaginary values of the argument even without the correction term required for the `cpf12` algorithm. With 20 terms the accuracy is better than 10^{-6} . In addition to the accuracy assessment we discuss methods for optimization and propose a combination of the 16-term approximation with the asymptotic approximation of Humlíček (1982) for high efficiency.

Key words: Techniques: spectroscopic – Methods: numerical – Line: profiles

1 INTRODUCTION

The Voigt profile (Olver et al. 2010, print companion to the NIST Digital Library of Mathematical Functions, <http://dlmf.nist.gov/>), i.e. the convolution of a Lorentzian and Gaussian profile, is ubiquitous in many branches of physics including astrophysics, e.g., Peraiah (2002); Heng (2017). From a computational point it is more convenient to consider the Voigt function $K(x, y)$ depending on two variables x and y (essentially the distance from the center peak and the ratio of the Lorentz to Gauss width) and the complex error function $w(x + iy)$, whose real part gives the Voigt function. Because there is no closed-form solution for the convolution integral, numerous algorithms have been developed in the past for efficient and/or accurate evaluation utilizing series or asymptotic expansions, continued fractions, rational approximations, Gauss-Hermite quadrature etc. (for an old review, still worth while to read, see Armstrong 1967).

For an assessment of Voigt and complex error function algorithms the intended application has to be considered. Highly accurate codes such as Poppe & Wijers (1990a,b) (14 significant digits stated accuracy) and Zaghoul & Ali (2011) (20 significant digits) or arbitrary precision codes (Boyer & Lynas-Gray 2014; Molin 2011) are indispensable for tests of an algorithm’s accuracy, but not necessarily fast. High resolution line-by-line (lbl) radiative transfer modeling (Bailey & Kedziora-Chudczer 2012; Schreier et al. 2014) presents

a “million- to billion-line challenge” (Grimm & Heng 2015; Heng 2017) where numerous Voigt functions have to be evaluated (Rothman et al. 2010; Tennyson & Yurchenko 2012) at thousands to millions of frequency grid points and computational speed becomes a prime concern.

In view of the few digits used for line strengths and broadening parameters in spectroscopic data bases such as HITRAN (Gordon et al. 2017), HITEMP (Rothman et al. 2010), or GEISA (Jacquinot-Husson et al. 2016), four significant digits for the Voigt function can be regarded as appropriate for many lbl calculations (Schreier 2011). On the other hand, the increasing quality of spectroscopic measurements has indicated deficiencies of the Voigt profile. More sophisticated line shapes accounting for collisional Dicke narrowing, line mixing, or speed-dependence are required (Varghese & Hanson 1984; Tennyson et al. 2014) that can often be expressed in terms of the complex error function, where the imaginary parts of the arguments can differ by more than ten orders of magnitude (Tran et al. 2013; Schreier 2017). For “standard” Voigt lbl modeling in astrophysical and planetary atmospheric spectroscopy y can be as large as 10^5 and as small as 10^{-8} (Lynas-Gray 1993; Wells 1999; Tepper-García 2006, 2007; Schreier 2011).

Rational approximations are particularly attractive because they can be implemented efficiently (i.e. only one division plus additions and multiplications) and allow for high accuracy. Hui et al. (1978) presented a single rational approximation with a fifth-degree numerator polynomial and a sixth-degree denominator polynomial that should be “suf-

[★] E-mail: franz.schreier@dlr.de

ficiently accurate for most applications over the whole complex plane”. Humlíček (1979) proposed a 12-term rational approximation “cpf12”; including a modification for small y the maximum relative error is less than $5 \cdot 10^{-6}$. Humlíček (1982) suggested a division of the complex plane into four subdomains; using appropriate rational approximations the “w4” code evaluates the complex error function accurate to four significant digits. Weideman (1994) developed a single rational approximation applicable in the entire complex plane whose accuracy can be adjusted by selection of the number N of terms. Kochanov (2011) presented two sums with four and six fractions with complex coefficients.

Problems of the Hui et al. (1978) code for small y have been reported early by Karp (1978) and Humlíček (1982) (see also Schreier 2011, Fig. 4). The Kochanov (2011) approximation shows significant errors for small $y < 10^{-2}$. Shippony & Read (1993, 2003) use the Hui et al. approach (for small $|x|$ and intermediate y) along with other techniques. Humlíček’s codes appear to belong to the most popular complex error function codes (according to Scopus (March 2018), there are 218 and 313 citing articles for the 1979 and 1982 paper, respectively). In particular the w4 code has been further developed by Kuntz (1997); Ruyten (2004); Imai et al. (2010) and the cpf12 code has been implemented by Tran et al. (2013) for evaluation of the “Hartmann-Tran” profile (Tennyson et al. 2014). Wells (1999) combined approximations of both Humlíček codes with slightly modified region bounds. Zaghoul (2017) allows to select the accuracy and uses Humlíček (1982) for low accuracy.

Except for Weideman (1994) all codes achieve moderate to high accuracy throughout the whole complex plane only by a combination of several methods. Unfortunately this can make the code difficult to optimize; this is especially true if the conditional branches are more than a simple “either-or” as for example in the Humlíček (1979) cpf12 algorithm. To avoid complicated (e.g. nested) if structures we have combined the asymptotic rational approximation of Humlíček (1982) for $|x| + y > 15$ with the Weideman (1994) approximation (Schreier 2011).

Whereas the four rational approximations of Humlíček (1982) have been utilized by several refinements, the somewhat older and slightly more accurate Humlíček (1979) cpf12 code has rarely been used in other complex error function codes (except for Wells (1999)). In particular we are not aware of any algorithm using the Humlíček (1979) rational approximation with a higher (or smaller) number of terms (i.e. $n \neq 12$).

The objective of this paper is an assessment of the Humlíček (1979) rational approximation for an “arbitrary” number of terms. We briefly review some basic facts in the following section, followed by a presentation of the Humlíček algorithm, and a discussion of our optimization strategies. In Section 3 we evaluate the accuracy and efficiency of the “generalized” Humlíček (1979) code, and provide our conclusions in Section 4.

2 THEORY AND METHODS

2.1 The Voigt and complex error function

The Voigt (or Hjerting) profile and the closely related Voigt function are defined by

$$g_V(\nu - \hat{\nu}, \gamma_L, \gamma_G) = \int_{-\infty}^{\infty} d\nu' g_L(\nu - \nu', \gamma_L) \times g_G(\nu' - \hat{\nu}, \gamma_G) \quad (1)$$

$$= \frac{\sqrt{\ln 2/\pi}}{\gamma_G} K(x, y) \\ K(x, y) = \frac{y}{\pi} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{(x-t)^2 + y^2} dt, \quad (2)$$

where γ_L and γ_G are the half widths at half maximum (HWHM) of the Lorentzian g_L and Gaussian g_G , respectively, and $\hat{\nu}$ is the center wavenumber or frequency (i.e. corresponding to the energy difference of an atomic or molecular transition). The dimensionless arguments of the Voigt function are defined as ratios

$$x = \sqrt{\ln 2} \frac{\nu - \hat{\nu}}{\gamma_G} \quad \text{and} \quad y = \sqrt{\ln 2} \frac{\gamma_L}{\gamma_G}. \quad (3)$$

Note that the profiles in Eq. (1) are normalized to one, $\int g(\nu) d\nu = 1$, whereas the Voigt function is normalized to $\sqrt{\pi}$.

The Voigt function is closely related to the complex error function (a.k.a. complex probability function, Fadde(y)eva function, or plasma dispersion function)

$$w(z) \equiv K(x, y) + iL(x, y) = \frac{i}{\pi} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{z-t} dt \quad (4)$$

with $z = x + iy$. Asymptotically the complex error function behaves as $i/(\sqrt{\pi}z)$, corresponding to a Lorentz-like Voigt function for large x . For $y = 0$ the real part becomes a Gauss function, i.e. $K(x, 0) = \text{Re}(w(x)) = \exp(-x^2)$. The half width at half maximum of the Voigt function can be estimated as (Olivero & Longbothum 1977; Schreier 2011)

$$x_{1/2} = \frac{1}{2} \left(y + \sqrt{y^2 + 4 \ln 2} \right). \quad (5)$$

2.2 The rational approximation by Humlíček (1979)

Humlíček presented a rational approximation in the form

$$w(z) = \sum_{\substack{k=-n/2 \\ k \neq 0}}^{n/2} \frac{\alpha_k + i\beta_k}{z - x_k + i\delta} = \sum_{k=1}^{n/2} \left(\frac{\alpha_k + i\beta_k}{z - x_k + i\delta} - \frac{\alpha_k - i\beta_k}{z + x_k + i\delta} \right) \quad (6)$$

with real constants

$$\alpha_k = -\alpha_{-k} = -\frac{1}{\pi} \omega_k e^{\delta^2} \sin(2x_k \delta) \quad (7)$$

$$\beta_k = +\beta_{-k} = +\frac{1}{\pi} \omega_k e^{\delta^2} \cos(2x_k \delta). \quad (8)$$

Here x_k and ω_k are the roots and weights of the n -point Gauss-Hermite formula (with n even and the nodes and weights antisymmetric and symmetric in k (e.g. Abramowitz & Stegun 1964, Table 25.10)), and δ is a positive constant to be chosen appropriately. Note that the constants α_k , β_k , x_k , ω_k , and δ depend on the number n of nodes, but the

superscript (n) used in the original paper has been omitted here; furthermore, Humlíček uses the notation $\delta_k^{(n)} \equiv y_0^{(n)}$ synonymously. As emphasized by Humlíček, “the choice of n and δ involves a compromise between several demands”, i.e. a larger number of terms increases the quality of the approximation, but also the computational effort. Table 1 of Humlíček (1979) gives some hints on the proper choice of δ for $8 \leq n \leq 20$.

To circumvent accuracy problems near the real axis where (6) fails to approximate an almost Gaussian function (see subsection 3.1), Humlíček introduced a modification for the real part

$$K(x, y) = e^{-x^2} + \sum_{\substack{k=-n/2 \\ k \neq 0}}^{n/2} \frac{y}{(x - x_k)^2 + \delta^2} \times \frac{\beta_k[(x - x_k)^2 - \delta(y + \delta)] - \alpha_k(x - x_k)(y + 2\delta)}{(x - x_k)^2 + (y + \delta)^2} \quad (9)$$

for $y < 0.85$ and $|x| > 18.1y + 1.65$. The “cpf12” code with $n = 12$ and $\delta = 1.5$ achieves a maximum relative error less than $2 \cdot 10^{-6}$ and $5 \cdot 10^{-6}$ for the real and imaginary part, respectively.

2.3 Optimization and a combination with the Humlíček (1982) asymptotic approximation

Equations (6) and (9) are suboptimal w.r.t. computational efficiency because of the numerous divisions: According to Ueberhuber (1997) divisions are significantly slower than additions or multiplications, and Goedecker & Hoisie (2001) noted that “the calculation of special functions, such as divisions, square roots, exponentials and logarithms requires anywhere from a few dozen cycles up to hundreds of cycles ... these calculations have to be decomposed into a sequence of elementary instructions such as multiplies and adds.” However, using elementary calculus (6) can be rewritten to the “standard form” of a rational approximation

$$w(z) = \frac{P(z)}{Q(z)} = \frac{\sum_{k=0}^{n-1} a_k z^k}{\sum_{l=0}^n b_l z^l} \quad (10)$$

Note that $a_{n-1} = i/\sqrt{\pi}$ and $b_n = 1$ in accordance with the asymptotic expansion of the complex error function. The numerator coefficients a_k are real for even k and purely imaginary for odd k ; furthermore $b_l = 0$ for all odd l . Finally it is worth to mention that the coefficients depend on the parameter δ .

For the 16-term approximation transformed into the form (10) the number of adds and multiplies is similar to the $N = 32$ rational approximation of Weideman (1994). However, the benchmark tests in Schreier (2011) have indicated that this approximation is significantly slower than the asymptotic rational approximation

$$R_{1,2}(z) = \frac{iz/\sqrt{\pi}}{z^2 - \frac{1}{2}} \quad (11)$$

used by Humlíček (1982) for $s = |x| + y > 15$. (The subscript on the lefthand side indicates the degree of the numerator and denominator polynomial.) Note that for lbl calculations

function values for only a few grid points have to be evaluated in the line center, and most values have to be evaluated in the line wings with large $|x|$. Accordingly we had suggested to use the Weideman approximation ($N = 24$) near the line center only and approximation (11) otherwise. Similarly we propose a combination of the two Humlíček (1979, 1982) approximations

$$w(z) = \begin{cases} \frac{iz/\sqrt{\pi}}{z^2 - \frac{1}{2}} & |x| + y > 15 \\ \frac{\sum_{k=0}^{n-1} a_k z^k}{\sum_{l=0}^n b_l z^l} & \text{otherwise.} \end{cases} \quad (12)$$

3 RESULTS

In the first subsection we provide an assessment of the accuracy of the Humlíček rational approximation (6) for $n \geq 12$. As an accuracy reference SciPy's wofz function is used, a combination of the Poppe & Wijers (1990a,b) and Zaghoul & Ali (2011) codes with a stated accuracy of at least 13 significant digits (Scientific Python `scipy.special.wofz` implementation based on S.G. Johnson's “Faddeeva package”, <http://ab-initio.mit.edu/Faddeeva>). Preliminary timing tests are presented in Subsection 3.2, and the performance of lbl cross section modeling is discussed in the third subsection. The combination of (6) and (9) with $n = 12$ is denoted as `cpf12`, whereas the optimized form (10) with $n = 12$ or $n = 16$ is denoted as `zpf12` and `zpf16`, respectively.

3.1 Accuracy

Fig. 1 depicts the relative error $\epsilon(x, y) = |K_{\text{cpf}} - K_{\text{wofz}}|/K_{\text{wofz}}$ of the real part of the $n = 12$ approximation. Fig. 1a clearly confirms Humlíček's statement w.r.t. accuracy problems of the approximation (6) for small y (the imaginary part $L(x, y)$ is computed correctly with a relative accuracy better than $4 \cdot 10^{-6}$). Including the correction term (9) drastically reduces the errors, and the maximum relative error of about 10^{-6} is in accordance with Humlíček's claims. However, the drawback of the correction term is the difficulty to implement this efficiently.

The $n = 16$ rational approximation is compared to the `wofz` reference code in Fig. 2. Variation of δ indicates that for $\delta = 1.3118$ the approximation is optimal w.r.t. overall accuracy, i.e. the relative error is $\leq 7.86 \cdot 10^{-5}$; for $y > 10^{-6}$ the error is less than about 10^{-5} for all x, y and the largest error is observed at $x \approx 4.8$ and $y = 10^{-8}$. Interestingly, the imaginary part $L(x, y)$ is slightly less accurate compared to the $n = 12$ approximation. For $\delta \approx 1.33$ (and very small y) the location of the maximum error moves into the line wings. For $\delta = 1.35$ the maximum of the relative error is slightly larger, about $2 \cdot 10^{-4}$, but this maximum is achieved for large x and very small y where the function values are already tiny ($K(5, 10^{-6}) \approx 10^{-8}$). For the imaginary part the maximum relative deviation is $2.6 \cdot 10^{-6}$.

In Fig. 3a we compare the maximum relative error $\max_x \epsilon(x, y)$ as a function of y . For $y > 10^{-4}$ the approximation (6) or (10) is accurate to four significant digits for all δ considered.

Increasing the number of terms in the rational approximation (6) further improves the accuracy. Fig. 3b shows

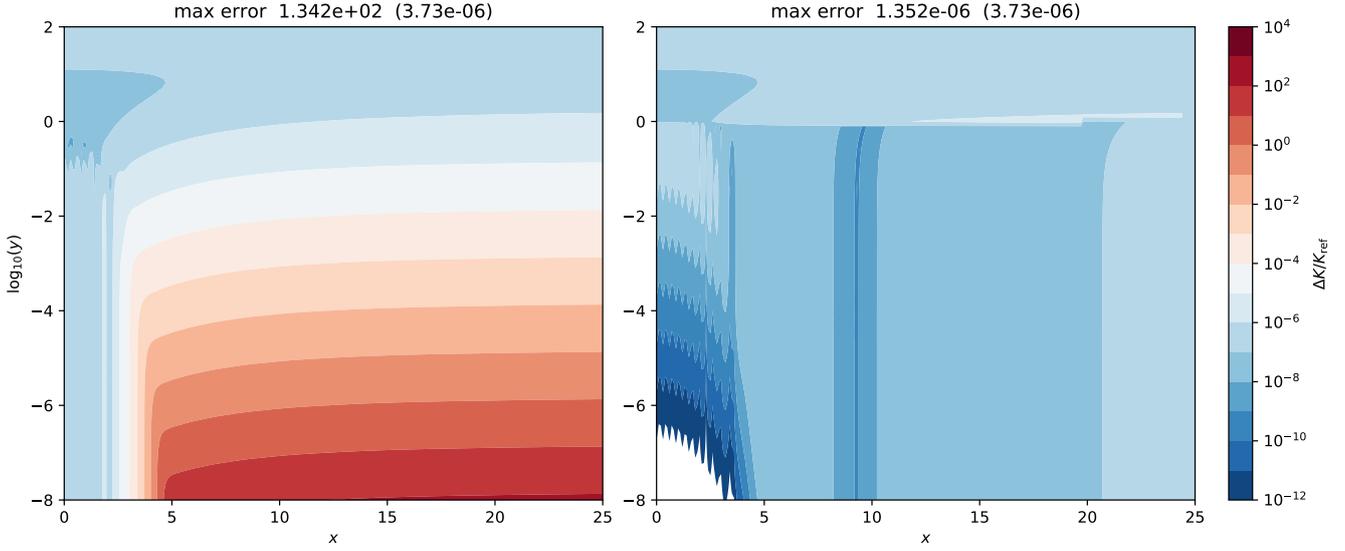


Figure 1. Comparison of the Humlíček *cpf12* approximation for $\delta = 1.5$ with the *wofz* reference. Left: approximation (6); Right: (6) and (9) combined. White areas indicate a relative error better than 10^{-12} . The numbers in the title indicate the maximum relative error of K and L (in parentheses).

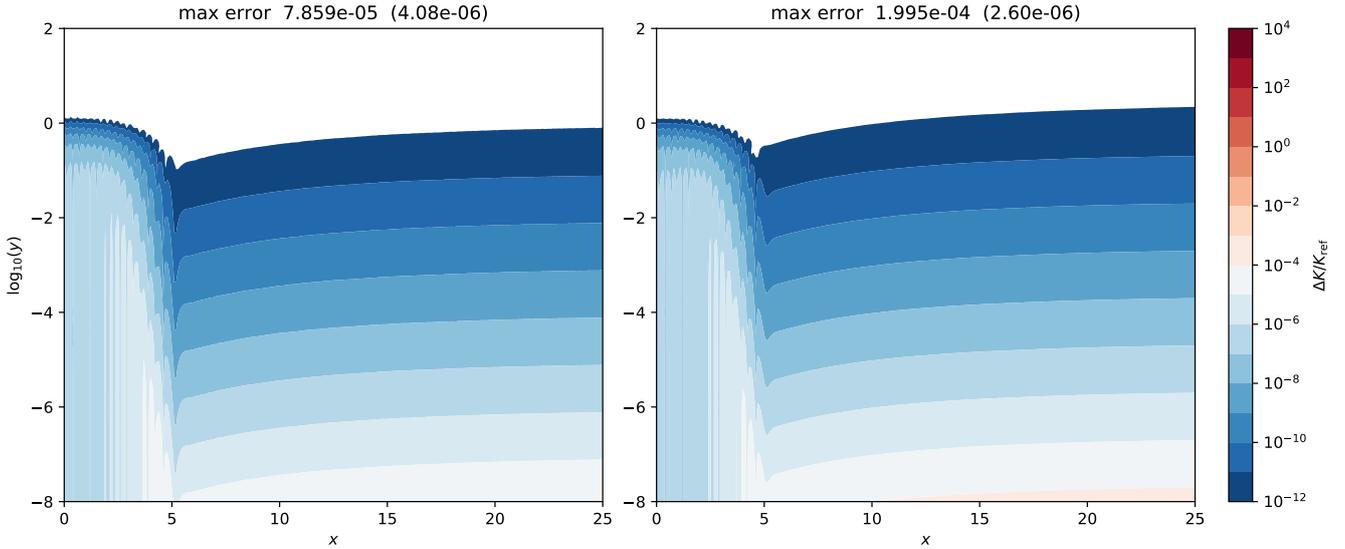


Figure 2. Comparison of Humlíček *zpf16* approximation for $\delta = 1.3118$ (left) and $\delta = 1.35$ (right) with the *wofz* reference.

that six significant digits can be achieved with $n = 20$ and $\delta \approx 1.55$. Contour plots of the relative error of the $n = 18$ and $n = 20$ approximations with optimal δ are shown in Fig. 4.

For a further appraisal of the generalized Humlíček (1979) approximation (6) we evaluate the Voigt function (i.e. $\text{Re}(w)$) for selected values of $z = x+iy$ and compare these with the results of Lether & Wenston (1991) who use a corrected midpoint quadrature rule. The values of $K(x, y)$ compiled in their Table 2 “are believed to be correct to twenty-five significant digits” and have also been cross-checked by Zaghoul (2007, his table 2) and Boyer & Lynas-Gray (2014, table 1). Table 1 shows that for $n = 16$ and $n = 20$ the expansion (6) is correct to at least five significant digits for all x, y pairs

except for $x = y = 10^{-3}$. Note that for $y = 10^{-20}$ (first data row) the Voigt function is essentially a Gaussian, and the correct value $\exp(-1.0) = 0.36787\ 94411\ 71442$ is reproduced with six, eight, and nine digits for $n = 16$, $n = 20$, and $n = 24$, respectively. In the last three rows with moderate x and y the *cpf* values are correct to ten digits.

3.2 Preliminary Speed Tests with IPython

To get a first impression about the performance of the various approximations, we have used IPython’s builtin “magic” function `%timeit`. A typical “experiment” looks like

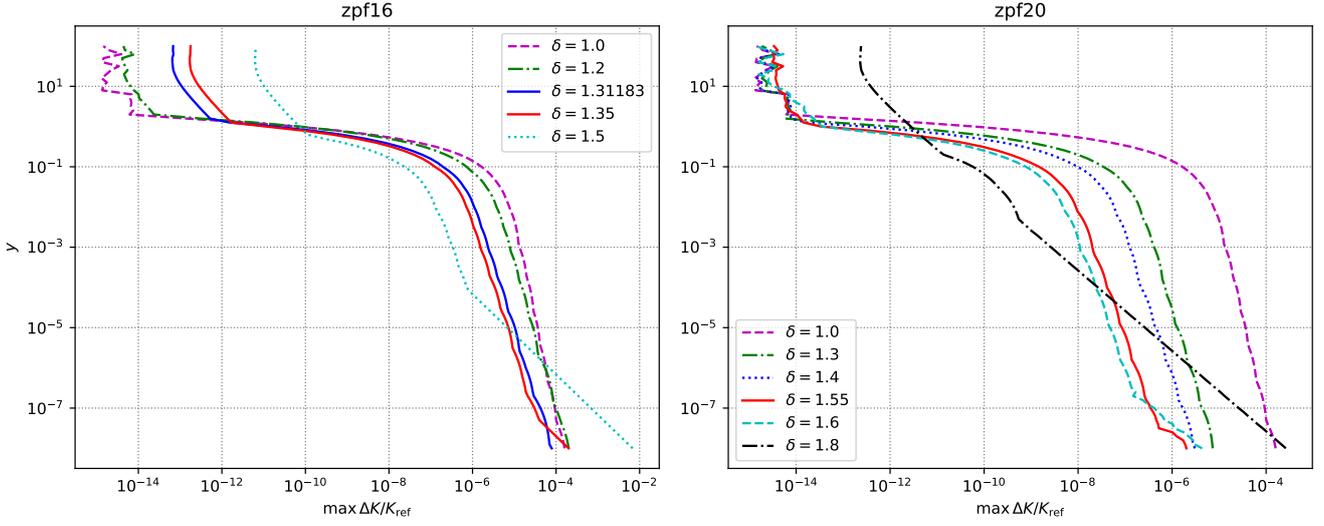


Figure 3. The maximum relative error of the Humlíček $n = 16$ and $n = 20$ approximations for various δ compared to the `wofz` reference.

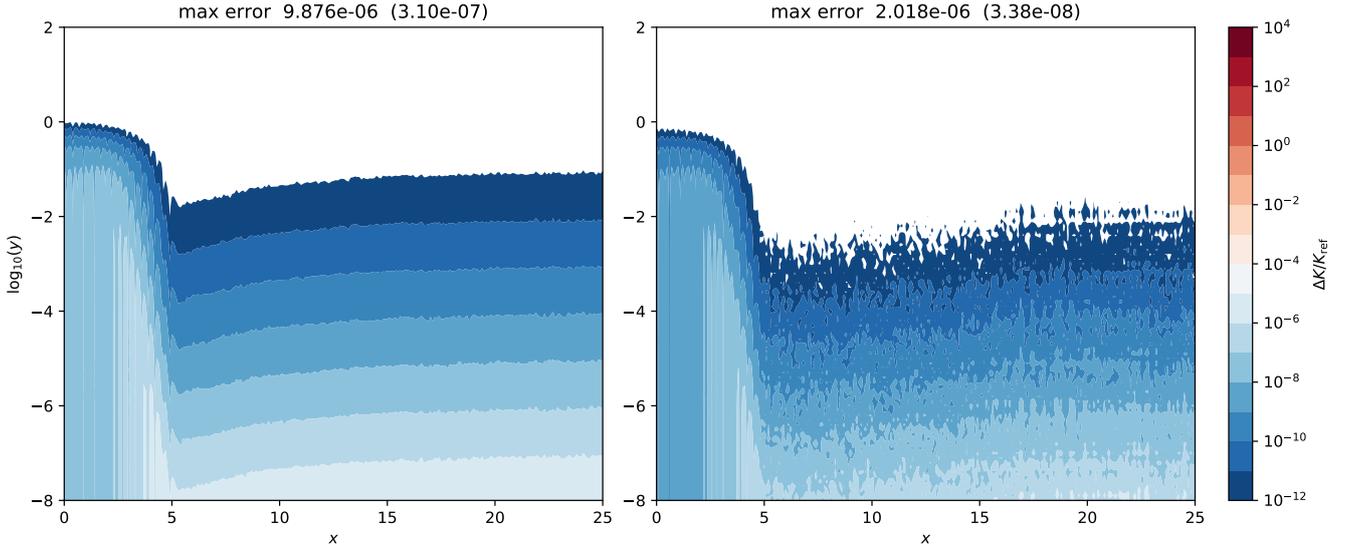


Figure 4. Comparison of Humlíček $n = 18$ ($\delta = 1.45$, left) and $n = 20$ ($\delta = 1.55$, right) approximation with the `wofz` reference.

Table 1. Comparison of Voigt function values for the $n = 16$, $n = 20$, and $n = 24$ approximation (6) with the precise values of Lether & Wenston (1991, their table 2 (given with 25 digits)).

x	y	Lether & Wenston	$n = 16$ ($\delta = 1.35$)	$n = 20$ ($\delta = 1.55$)	$n = 24$ ($\delta = 1.4$)
1.0	10^{-20}	0.36787 94411 71442	0.36787 93403 59605	0.36787 94396 18318	0.36787 94414 03711
10.	10^{-4}	0.57287 17561 64533 $\cdot 10^{-6}$	0.57287 17507 11831 $\cdot 10^{-6}$	0.57287 17561 87614 $\cdot 10^{-6}$	0.57287 17561 66040 $\cdot 10^{-6}$
10^{-3}	10^{-3}	0.99887 16233 35411	0.99885 13083 62977	0.99886 16184 44897	0.99887 16170 22162
0.0	0.25	0.77034 65477 30996	0.77034 65303 12796	0.77034 65475 77724	0.77034 65475 71460
1.0	0.50	0.35490 03328 67577	0.35490 03328 53826	0.35490 03328 64360	0.35490 03328 66028
5.0	5.0	0.56965 43988 71769 $\cdot 10^{-6}$	0.56965 43988 81711 $\cdot 10^{-1}$	0.56965 43988 81771 $\cdot 10^{-1}$	0.56965 43988 81769 $\cdot 10^{-1}$
1.0	10.0	0.55598 31964 10553 $\cdot 10^{-6}$	0.55598 31964 10505 $\cdot 10^{-1}$	0.55598 31964 10555 $\cdot 10^{-1}$	0.55598 31964 10553 $\cdot 10^{-1}$

```
In [1]: from cpfX import zpf16p
In [2]: x=np.linspace(0.,100.,10001); y=0.001;
        z=x+1j*y
In [3]: %timeit zpf16p(z)
1000 loops, best of 3: 1.04 ms per loop
```

Table 2 summarizes the results of these experiments. Note that the grid point spacing $\delta x = 0.1$ (first three columns) is more than adequate to sample the line center region as the half width (5) is approximately $x_{1/2} \approx 1$ for small y .

For the Python translation of the original Fortran `cpf12` code two versions have been implemented: A scalar function with an `if` statement that is transformed in an array function using NumPy’s `vectorize` function class and an array function exploiting a mask and NumPy’s `where`. The first two rows of the table clearly indicate that the vectorized version is significantly slower.

The polynomials in (10) can be implemented either with the Horner scheme coded by hand (`zpf12h`, `zpf16h`) or exploiting NumPy’s `poly1d` class (`zpf12p`, `zpf16p`), and the `%timeit` tests demonstrate that the hand-coded Horner scheme is slightly faster for all cases. It should also be noted that implementing the denominator as a polynomial of z^2 (the odd coefficients are zero) is more efficient than the polynomial of z .

For comparison we also report times for several Weideman (1994) approximations. His N -term approximation requires $N+4$ multiplications plus one division, so the computational effort for `weideman32` is similar to `zpf16`. However, as mentioned above, the 16-degree denominator polynomial of `zpf16` can be implemented more efficiently because of the vanishing odd coefficients, i.e. with 24 add-multiplies for `zpf16` similar to `weideman24`.

The `wofz` implementation documented in the very last row appears to be faster than all rational approximations. Note, however, that this special function imported from SciPy is actually a C code linked to Python in contrast to our pure Numeric Python codes.

Note that the execution time is approximately proportional to the number of function evaluations only for the `cpf12v` case (first data row) that is by far the slowest implementation. Furthermore, times in the fourth column ($n_x = 10000$) are roughly ten times larger compared to the third column ($n_x = 1000$). The deviations from strict proportionality might indicate that `%timeit` estimates also include overhead for function call etc.

3.3 Performance of lbl cross section modeling

For realistic benchmarks we test the generalized Humlíček approximation (10) in an atmospheric radiative transfer lbl modeling context. Radiative transfer in Earth’s atmosphere is clearly important in the geosciences, esp. for climate modeling and remote sensing, but is also relevant for astronomy, i.e. for corrections of telluric absorption and emission in ground-based observations (e.g. Seifahrt et al. 2010; Bertaux et al. 2014; Smette et al. 2015) or radiative transfer modeling of Earth-like exoplanets (e.g. Des Marais et al. 2002; Robinson & Reinhard 2018). To make comparison with our previous studies (Schreier & Kohlert 2008; Schreier 2011) easier, we compute high resolution molecular absorption cross sec-

tions for the Submillimeter Radiometer (SMR) aboard the Swedish small satellite ODIN (Murtagh et al. 2002; Nordh et al. 2003), an aeronomy and astronomy mission launched in 2001 (the astronomy mission was successfully concluded in spring 2007, see also Hjalmarnson et al. (2003)).

The cross sections

$$k(\nu, p, T) = \sum_l S_l(T) g_V(\nu - \hat{\nu}_l, \gamma_l^{(L)}(p, T), \gamma_l^{(G)}(T)). \quad (13)$$

(with line position $\hat{\nu}_l$ and line strength S_l) are computed in the $16\text{--}17\text{ cm}^{-1}$ interval (480 – 510 GHz) for thirteen pressure-temperature pairs (corresponding to the 0–120 km altitude range with $\delta z = 10$ km steps). Because of the importance of line wing contributions all lines in the extended $6\text{--}27\text{ cm}^{-1}$ interval are considered: For nitric acid (HNO_3) HITRAN 00 (Rothman et al. 2003) lists 2376 lines (HITRAN 16 (Gordon et al. 2017) has about 250 000 lines), and there are 21 565 lines of ozone (O_3) in all versions of HITRAN since 1996. The mean Lorentz to Doppler (Gauss) width ratio y decreases from $7 \cdot 10^3$ (at bottom-of-atmosphere, BoA) to $1.4 \cdot 10^{-4}$ (at top-of-atmosphere, ToA) for HNO_3 ; the Doppler width is slightly larger for O_3 because of the smaller mass, hence y is somewhat smaller.

3.3.1 HNO_3 cross sections — Python

The HNO_3 cross sections are modeled using the `lbl2xs.py` function script of `Py4CATS` — Python for Computational Atmospheric Spectroscopy (available at <https://atmos.eoc.dlr.de/tools/Py4CATS/>) using a “brute force” approach, i.e. without further approximations in the line wings (e.g. coarse grid). Note that the spectral resolution is depending on line width (essentially pressure, hence altitude), i.e. the grid point spacing δx is set to a fraction of the mean half width, typically $\delta x = \bar{x}_{1/2}/4$ or $\delta \nu = \bar{\gamma}_V/4$. The number of wavenumber grid points increases from less than one hundred at BoA to almost half a million at an altitude of 90 km.

The execution times for a single function value (i.e. the total elapsed CPU time measured with the `clock` function of Python’s `time` module divided by the number of lines and the number of x grid points) shown in Fig. 5 essentially confirm the IPython `%timeit` tests reported in the previous subsection. Note that pre- and postprocessing steps (e.g. reading the line data) have not been considered for the timing.

Both versions of the `cpf12` code are significantly slower than all other functions: the `cpf12w` code needs about 600 ns for $p \leq 10$ mb and is not shown. Except for large pressures the various Weideman functions and the two implementations of the approximation (10) with $n = 16$ require about 100 to 175 ns, and the use of an asymptotic approximation for large x is tempting. The performance of the combination (12) as well as the Humlíček (1982)–Weideman (1994) combination (with $N = 24$) are almost identical, and somewhat worse to the `wofz` C function linked to Python. The combination of the $R_{3,4}$ rational approximation of Humlíček (1982) with the Weideman $N = 32$ approximation (with a cut at $s = 8$, cf. Schreier (2017)) has a relative error better than $2 \cdot 10^{-6}$ and is slightly slower.

The good performance of the `wofz` C function reported in the previous subsection suggests that implementation of compute intensive code segments in a compiled language might be advantageous. Accordingly we have used the

Table 2. Execution time measured by the `% timeit` function in the IPython interpreter. For all columns $y = 0.001$ has been used. n_x is the number of x grid points and function evaluations. The “zpf” indicates implementations of the optimized rational approximation (10). The tests have been performed on a laptop with an Intel x86_64 CPU running at 2.7 GHz and cache size 3072 KB.

	$0 \leq x \leq 25$	$0 \leq x \leq 50$	$0 \leq x \leq 100$	$0 \leq x \leq 1000$
n_x	251	501	1001	10001
cpf12v	12.1 ms	24.8 ms	49.4 ms	492 ms
cpf12w	877 μ s	1.01 ms	1.17 ms	4.39 ms
zpf12h	64.2 μ s	84 μ s	119 μ s	829 μ s
zpf12p	116 μ s	145 μ s	185 μ s	1.01 ms
zpf16h	90.1 μ s	110 μ s	147 μ s	904 μ s
zpf16p	117 μ s	146 μ s	185 μ s	1.04 ms
weideman16	78.5 μ s	100 μ s	134 μ s	776 μ s
weideman24	105 μ s	138 μ s	172 μ s	1.04 ms
weideman32	132 μ s	162 μ s	216 μ s	1.31 ms
wofz	52.6 μ s	66.8 μ s	91.7 μ s	886 μ s

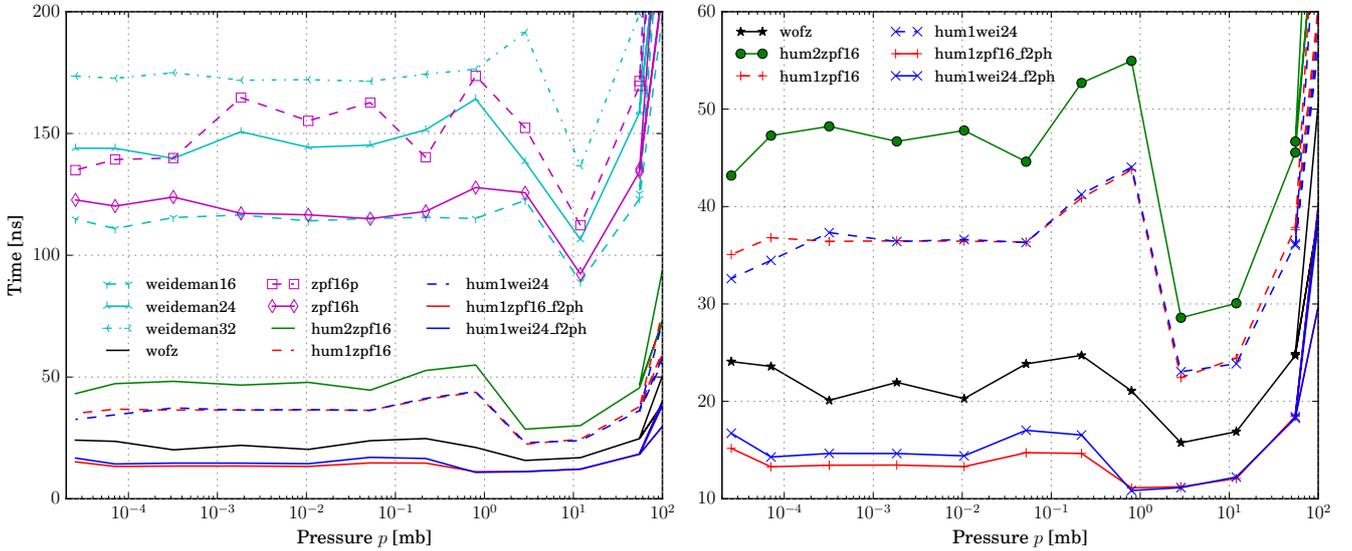


Figure 5. Execution time (ns) per function evaluation. The tests have been performed on an idle node of a Linux cluster with an Intel XEON CPU at 2.36 GHz.

f2py Fortran-Python interface generator (Peterson 2009) to make calls to Fortran subroutines possible. Both combinations of the Humlíček (1982) $R_{1,2}$ asymptotic approximation (11) with the Humlíček (1979) zpf16 approximation (10) or with the 24-term Weideman approximation (denoted `hum1zpf16_f2py` and `hum1wei24_f2py`) are more than a factor two faster than the corresponding pure Python implementation and clearly superior with respect to computational efficiency.

Compared to the tests reported in Schreier (2011, Fig. 11) the Humlíček–Weideman as well as the pure Weideman functions are about a factor 10 faster now. Note that the minimum times seen here are comparable to the Fortran execution times shown in Fig. 10 of Schreier (2011). This considerable speed-up is presumably a combination of better hardware and better software (improved performance of NumPy).

3.3.2 O_3 cross sections — Fortran

The Humlíček (1979) $n = 16$ approximation (10) along with several other complex error function algorithms has also been implemented in Fortran90. The execution time for the lbl modeling of O_3 cross sections is measured by means of the `cpu_time` intrinsic function.

In contrast to the Python tests the runtime appears to be fairly constant for pressures smaller than about 10 mb where the number of wavenumber grid points is already large ($n_x > 10^4$). Hence we only report the average run time in Table 3. `cpf12` roughly corresponds to the original code documented in the Humlíček (1979) paper and Fortran can execute this code significantly faster compared to NumPy. The optimized implementation (10) of the 16-term approximation `zpf16` is executed in Fortran with roughly double speed. The combinations of the `zpf16` approximation (10) or Weideman’s approximation with the Humlíček (1982) R_{12} or R_{34} approximation for large $|x|$ have approximately identical run

Table 3. Execution time [ns] per function evaluation for the Fortran 90 implementation. GNU Fortran or NAG compiler with optimization flag -O3 (the numbers indicate the version). Linux cluster node as in Fig. 5.

	gfortran 4.8	gfortran 6.2	nagfor 5.3
cpf12	97	102.6	125
zpf16	64.6	61.2	66.24
hum1zpf16	14.52	14.61	16.98
hum1wei24	14.55	14.49	16.39
hum2wei32	14.75	14.86	18.15
weideman16	70.34		
weideman32	131.1		

times. Comparison with Figure 5 reveals that the speed of the Fortran and NumPy-f2py codes is roughly similar, i.e. 15 ns per function value. The two versions of the GNU gfortran compiler produce code with approximately the same performance, whereas the NAG compiled code is somewhat slower (similar to Schreier (2011, Fig. 9)).

There appears to be no pure Fortran array implementation of `wofz`. The `libcerf` numeric library (<http://apps.jcns.fz-juelich.de/libcerf>) implementation of the Faddeeva package also provides Fortran bindings, but only for scalar arguments. However, the execution time of the `hum1zpf16` and `hum1wei24` pure Fortran subroutines and the functions wrapped to Python via `f2py` are almost identical, so we can expect that a Fortran array version of `wofz` would perform similar to the SciPy-C version that was about a factor 1.5 slower compared to `hum1zpf16` and `hum1wei24` (compare Figure 5 and Table 3).

4 SUMMARY AND CONCLUSIONS

Generalizations of the Humlíček (1979) `cpf12` rational approximation (with 12 terms) for the Voigt and complex error function have been presented and optimized implementations have been developed. The rational approximation with a degree 16 denominator polynomial is accurate to at least five significant digits except for very small y , and somewhat larger deviations show up in the wings where the function value is already extremely small. For $n \geq 16$ the correction term needed to compensate for the problems of the $n = 12$ approximation (6) for large $|x|$ and small y is not required anymore, i.e. a single rational approximation can be used to evaluate the complex error function over almost the entire complex plane.

The sum of n fractions of the original approximation of Humlíček (1979) are computationally inefficient, but can be readily transformed into a single fraction with an $n-1$ degree numerator polynomial and an n degree denominator polynomial. The performance of the original and generalized Humlíček algorithms implemented as Numeric Python functions has been tested and compared to other rational approximations. For a more realistic scenario molecular absorption cross sections have been computed line-by-line using Python and Fortran implementations. The combination of the $n = 16$ approximation with the asymptotic approximation also developed by Humlíček (1982) further improved the performance. The combination of the Humlíček (1979, 1982) meth-

ods as well as the Humlíček–Weideman combination need about 15 ns for a single function value both with Fortran and Python, in strong contrast to our findings seven years ago (Schreier 2011) where Fortran was about a factor ten faster. This impressive speed of the Python implementation on a par with the Fortran code explains the increasing popularity of the “Python-based ecosystem” (www.scipy.org) and packages such as Astropy (Astropy Collaboration 2013).

In conclusion, the combinations of the Humlíček (1979) or Weideman (1994) approximation with the Humlíček (1982) asymptotic approximation can be recommended for both accurate (five or more digits) and efficient Voigt and complex error function evaluations. For high performance with Python, a Fortran (or C) implementation linked to Python is advantageous. When speed is not an issue, the generalization of the original `cpf12` code to 16 or more terms can be used for all x, y without any conditional branches.

SUPPLEMENTARY MATERIAL

Two Python source files are provided as supplementary material with the online version of this paper. Additionally we provide the Fortran code used for the tests of subsection 3.3.2 including the Humlíček (1979, 1982) combination (12).

`cpfX.py` Various implementations of the Humlíček (1979) algorithm:

- Python/NumPy implementations of the original Fortran 77 source code (with two versions to combine the two regions);
- A generalization of the region I rational approximation to arbitrary (even) number of terms;
- An optimized implementation of the region I approximation for 16 terms using a single fraction;
- The combination (12) for NumPy.

`fig124.py` Execution of this script inside the IPython interpreter should produce the Figures 1, 2, and 4 of the manuscript.

`Test_voigt_speed.f90` The Fortran 90 program including subroutines of various complex error function algorithms.

ACKNOWLEDGEMENTS

Financial support by the DFG project SCHR 1125/3-1 is greatly appreciated. I would also like to thank Thomas Trautmann for critical reading of the manuscript. The transformation of the sum of quotients (6) to the optimized form (10) has been computed with SYMPY ([HTTP://WWW.SYMPY.ORG/](http://www.sympy.org/)).

REFERENCES

- ABRAMOWITZ M., STEGUN I., 1964, HANDBOOK OF MATHEMATICAL FUNCTIONS. NATIONAL BUREAU OF STANDARDS, AMS55, NEW YORK
- ARMSTRONG B., 1967, *JQSRT*, 7, 61
- ASTROPY COLLABORATION T., ROBITAILLE T., TOLLERUD E., ET AL., 2013, *A&A*, 558, A33
- BAILEY J., KEDZIORA-CHUDCZER L., 2012, *MNRAS*, 419, 1913

- BERTAUX L., LALLEMENT R., FERRON S., BOONNE C., BODICHON R., 2014, *A&A*, 564, A46
- BOYER W., LYNAS-GRAY A., 2014, *MNRAS*, 444, 2555
- DES MARAIS D., ET AL., 2002, *ASTROBIOLOGY*, 2, 153
- GOEDECKER S., HOISIE A., 2001, PERFORMANCE OPTIMIZATION OF NUMERICALLY INTENSIVE CODES. SIAM, PHILADELPHIA, PA
- GORDON I., ET AL., 2017, *JQSRT*, 203, 3
- GRIMM S. L., HENG K., 2015, *APJ*, 808, 182
- HENG K., 2017, EXOPLANETARY ATMOSPHERES — THEORETICAL CONCEPTS AND FOUNDATIONS. PRINCETON UNIVERSITY PRESS
- HJALMARSON A., ET AL., 2003, *A&A*, 402, L39
- HUI A., ARMSTRONG B., WRAY A., 1978, *JQSRT*, 19, 509
- HUMLÍČEK J., 1979, *JQSRT*, 21, 309
- HUMLÍČEK J., 1982, *JQSRT*, 27, 437
- IMAI K., SUZUKI M., TAKAHASHI C., 2010, *ADV. SPACE RES.*, 45, 669
- JACQUINET-HUSSON N., ET AL., 2016, *J. MOL. SPECTROSC.*, 327, 31
- KARP A., 1978, *JQSRT*, 20, 379
- KOCHANOV V., 2011, *ATMOSPHERIC AND OCEANIC OPTICS*, 24, 432
- KUNTZ M., 1997, *JQSRT*, 57, 819
- LEATHER F., WENSTON P., 1991, *J. COMP. APPL. MATH.*, 34, 75
- LYNAS-GRAY A., 1993, *COMP. PHYS. COMM.*, 75, 135
- MOLIN P., 2011, MULTI-PRECISION COMPUTATION OF THE COMPLEX ERROR FUNCTION, PREPRINT AT [HTTPS://HAL.ARCHIVES-OUVERTES.FR/HAL-00580855](https://hal.archives-ouvertes.fr/hal-00580855)
- MURTAGH D., ET AL., 2002, *CAN. J. PHYS.*, 80, 309
- NORDH H. L., ET AL., 2003, *A&A*, 402, L21
- OLIVERO J., LONGBOTHUM R., 1977, *JQSRT*, 17, 233
- OLVER F., LOZIER D., BOISVERT R., CLARK C., EDS, 2010, NIST HANDBOOK OF MATHEMATICAL FUNCTIONS. CAMBRIDGE UNIVERSITY PRESS, NEW YORK, NY
- PERALIAH A., 2002, AN INTRODUCTION TO RADIATIVE TRANSFER: METHODS AND APPLICATIONS IN ASTROPHYSICS. CAMBRIDGE UNIVERSITY PRESS
- PETERSON P., 2009, *INT. J. COMP. SCI. ENG.*, 4, 296
- POPPE G., WIJERS C., 1990A, *ACM TOMS*, 16, 38
- POPPE G., WIJERS C., 1990B, *ACM TOMS*, 16, 47
- ROBINSON T., REINHARD C., 2018, PREPRINT, ([arXiv:1804.04138](https://arxiv.org/abs/1804.04138))
- ROTHMAN L., ET AL., 2003, *JQSRT*, 82, 5
- ROTHMAN L., ET AL., 2010, *JQSRT*, 111, 2139
- RUYTEN W., 2004, *JQSRT*, 86, 231
- SCHREIER F., 2011, *JQSRT*, 112, 1010
- SCHREIER F., 2017, *JQSRT*, 187, 44
- SCHREIER F., KOHLERT D., 2008, *COMP. PHYS. COMM.*, 179, 457
- SCHREIER F., GIMENO GARCÍA S., HEDELT P., HESS M., MENDROK J., VASQUEZ M., XU J., 2014, *JQSRT*, 137, 29
- SEIFAHRT A., KÄUFL H., ZÄNGL G., BEAN J., RICHTER M., SIEBENMORGEN R., 2010, *A&A*, 524, A11
- SHIPPONY Z., READ W., 1993, *JQSRT*, 50, 635
- SHIPPONY Z., READ W., 2003, *JQSRT*, 78, 255
- SMETTE A., ET AL., 2015, *A&A*, 576, A77
- TENNYSON J., YURCHENKO S. N., 2012, *MNRAS*, 425, 21
- TENNYSON J., ET AL., 2014, *PURE APPL. CHEM.*, 86, 1931
- TEPPER-GARCÍA T., 2006, *MNRAS*, 369, 2025
- TEPPER-GARCÍA T., 2007, *MNRAS*, 382, 1375
- TRAN H., NGO N., HARTMANN J.-M., 2013, *JQSRT*, 129, 199
- UEBERHUBER C., 1997, NUMERICAL COMPUTATION. SPRINGER
- VARGHESE P., HANSON R., 1984, *APPL. OPT.*, 23, 2376
- WEIDEMAN J., 1994, *SIAM J. NUM. ANAL.*, 31, 1497
- WELLS R., 1999, *JQSRT*, 62, 29
- ZAGHLOUL M. R., 2007, *MNRAS*, 375, 1043
- ZAGHLOUL M., 2017, *ACM TOMS*, 44, 22:1
- ZAGHLOUL M., ALI A., 2011, *ACM TOMS*, 38, 15:1

This paper has been typeset from a $\text{\TeX}/\text{\LaTeX}$ file prepared by the author.