J. Quant. Spectrosc. Radiat. Transfer Vol. 48, No. 5/6, pp. 743–762, 1992
Printed in Great Britain.

0022-4073/92 $5.00 + 0.00

THE VOIGT AND COMPLEX ERROR FUNCTION: A COMPARISON OF COMPUTATIONAL METHODS

F. SCHREIER
DLR—Deutsche Forschungsanstalt für Luft- und Raumfahrt, Institute of Optoelectronics, D-8031 Oberpfaffenhofen, Germany

(Received 16 October 1991; received for publication 18 March 1992)

Abstract—Several computational procedures for the Voigt function and complex error function are discussed and compared with respect to accuracy and running time. Vectorization of the codes is applied where possible. Computational speed varied over two orders of magnitude. Even without vectorization, restructuring of the source code can yield a significant acceleration. The computational effort for Fourier transform methods is estimated and compared with other methods. For applications involving least-squares-fitting, the evaluation of the complex error function provides an efficient way to calculate both the Voigt function and its partial derivatives.

1. INTRODUCTION

In molecular spectroscopy and atmospheric radiative transfer, one frequently has to take into account the combined effects of pressure broadening (corresponding to a Lorentzian line shape g_L) and Doppler broadening (corresponding to a Gaussian line shape g_D). The convolution of these profiles leads to the well known Voigt line profile[1,2]

$$g_V(\tilde{\nu}) = g_L \otimes g_D = \int_{-\infty}^{+\infty} g_L(\tilde{\nu} - \tilde{\nu}') g_D(\tilde{\nu}')\, d\tilde{\nu}', \tag{1}$$

where $\tilde{\nu}$ is the wavenumber (or frequency) and g_L, g_D, and g_V are normalized to unity.

The convolution integral defining the Voigt profile or the closely related complex error function cannot be evaluated in closed form and therefore has to be computed numerically. This has led to the development of a wide variety of algorithms as well as to extensive tabulations (see Armstrong[1] for a review up to 1966). For comparisons of the various algorithms, both accuracy and speed have to be considered. These are in general conflicting requirements, because improved accuracy of an expansion or Gauss–Hermite integration is in general achieved only by increasing the number of terms used. The decision for a certain algorithm therefore has to be done with the specific application in mind.

Although two comparative studies have already been presented by Twitty, Rarig, and Thompson[3] and by Klim[4], a new investigation is useful for several reasons. First, some of the newer algorithms have not been included in the comparisons of Twitty et al and Klim. Secondly, the implications on speed due to vectorization on supercomputers should be considered. Finally, for various applications, the imaginary part of the complex error function is also useful and should be included in the discussion.

2. DEFINITIONS AND FUNDAMENTAL RELATIONS

It is convenient to introduce the Voigt function $K(x, y)$ defined by

$$g_V = \frac{\sqrt{\ln 2/\pi}}{\gamma_D} K(x, y) \tag{2}$$

and

$$K(x, y) = \frac{y}{\pi} \int_{-\infty}^{+\infty} \frac{e^{-t^2}}{(x - t)^2 + y^2}\, dt. \tag{3}$$

The dimensionless variables x, y are defined in terms of the distance from the line center, $\tilde{\nu} - \tilde{\nu}_0$, and the Lorentzian and Doppler half widths γ_L, γ_D:

$$x = \sqrt{\ln 2}\,\frac{\tilde{\nu} - \tilde{\nu}_0}{\gamma_D}, \qquad y = \sqrt{\ln 2}\,\frac{\gamma_L}{\gamma_D}. \tag{4}$$

Pressure broadening of molecular transitions is proportional to the pressure p with typical values $\gamma_L \approx 0.1p[\mathrm{cm}^{-1}/\mathrm{atm}]$ (see also Table 2 of Rothman et al[5]). The Doppler half width is proportional to the line center wavenumber $\tilde{\nu}_0$ and varies with temperature T and molecular mass m according to

$$\gamma_D = \tilde{\nu}_0 \sqrt{\frac{2 \ln 2kT}{mc^2}},$$

where k and c are the Boltzmann constant and the speed of light. For atmospheric molecules of average mass, one finds $\gamma_D \approx 6 \cdot 10^{-8}\, \tilde{\nu}_0 \sqrt{T[\mathrm{K}]}$.

For vanishing arguments x or y one has $K(0, y) = \mathrm{e}^{y^2}[1 - \mathrm{erf}(y)]$ and $K(x, 0) = \mathrm{e}^{-x^2}$ respectively, where erf is the error function.[6] A plot of the Voigt function for $0 < x < 5$ and $0 < y < 5$ is given in Fig. 1.

Combining x and y into the complex variable $z = x + iy$, the Voigt function can be represented as the real part $K(x, y) = \mathrm{Re}[W(z)]$ of the complex function

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

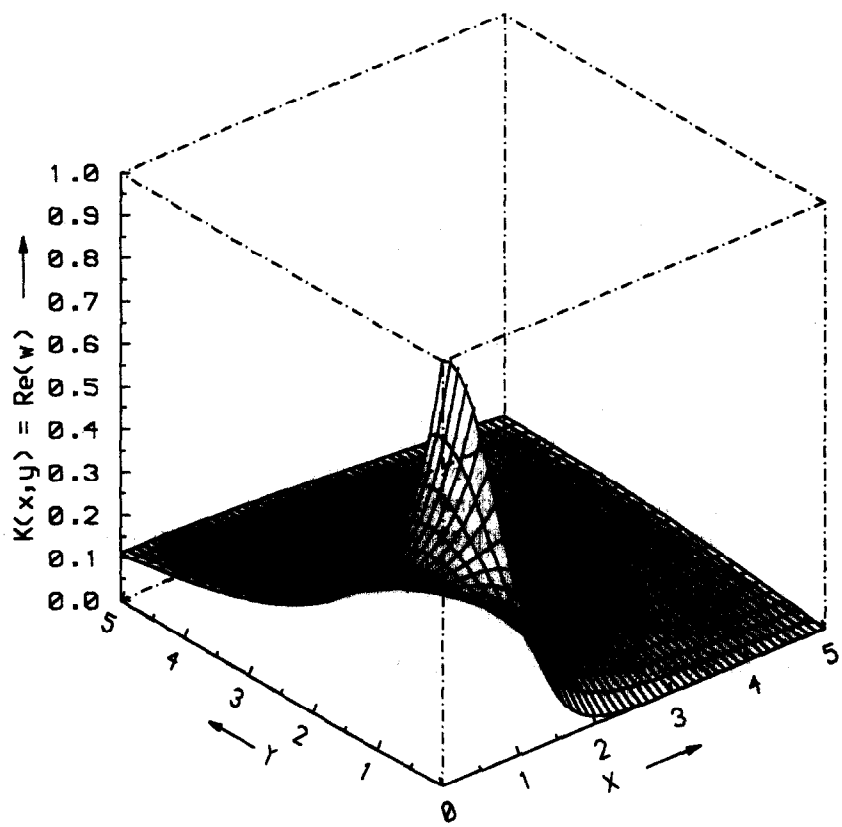

Fig. 1. Voigt function: Armstrong's algorithm.

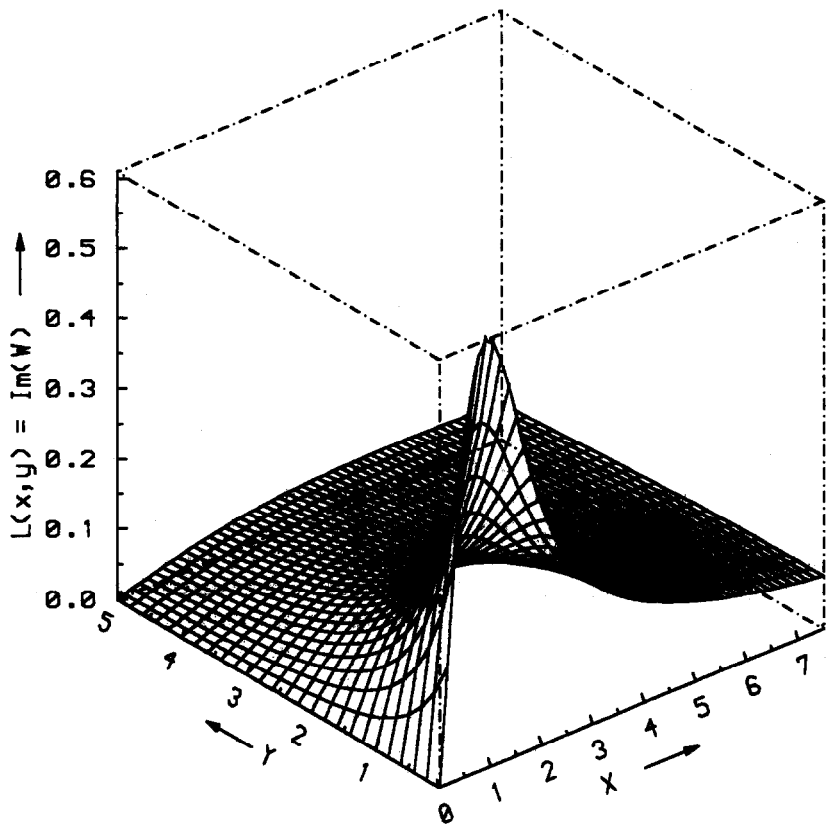

Fig. 2. Imaginary part of complex error function: Hui's algorithm.

The imaginary part is shown in Fig. 2 and equals

$$L(x, y) = \mathrm{Im}[W(z)] = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{(x-t)\mathrm{e}^{-t^2}}{(x-t)^2 + y^2}\, \mathrm{d}t. \tag{6}$$

It can be used for efficient calculation of the derivatives of the Voigt function. For $y = 0$, the integral defining $L(x, y)$ is only defined in the Cauchy principal value sense because of the pole at $t = x$.

The function W is closely related to the complex error function (probability function)[6] according to

$$w(z) = \begin{cases} W(z) & \text{for } y > 0, \\ W(z) + 2\,\mathrm{e}^{-z^2} & \text{for } y < 0, \end{cases} \tag{7}$$

where $w(z)$ is defined by

$$\begin{aligned} w(z) &= \mathrm{e}^{-z^2}\left(1 + \frac{2i}{\sqrt{\pi}} \int_0^z \mathrm{e}^{t^2}\, \mathrm{d}t\right) \\ &= \mathrm{e}^{-z^2}\Big[1 - \mathrm{erf}(-iz)\Big] = \mathrm{e}^{-z^2}\,\mathrm{erfc}(-iz). \end{aligned} \tag{8}$$

Further mathematical properties and relationships of the Voigt function and complex error function can be found in Armstrong's extensive reviews[1,2] or in Abramowitz and Stegun.[6] Various empirical expressions for the half width $x_{1/2}$ (HWHM) of the Voigt profile, defined through the relation

$$K(x_{1/2}, y) = \tfrac{1}{2} K(0, y) = \tfrac{1}{2}\mathrm{e}^{y^2}\mathrm{erfc}(y), \tag{9}$$

have been reviewed by Olivero and Longbothum.[7]

3. COMPUTATIONAL APPROACHES

The computational approaches can be classified in several groups. An overview of some of these methods is given in Table 1. (i) First, there are empirical fits to the Voigt function as proposed for example by Kielkopf[8] or Whiting.[9] These approximate the Voigt profile by a weighted sum of the Lorentz function and the Gauss function plus a correction term. The weight factors are determined by the requirement that the approximation should reduce to the Lorentz or Doppler function for pure pressure ($\gamma_D = 0$) or Doppler ($\gamma_L = 0$) broadening (compare appendix). Obviously an infinite set of weight functions will fulfil this condition. (ii) A second approach[10,11] evaluates the Voigt function in the Fourier-transformed space, where the convolution integral is just the product of the Fourier transforms of the Lorentz and Gauss functions. (iii) A third group of algorithms uses series expansions, asymptotic expansions, continued fraction expansions, rational approximations, Gauss–Hermite integration or combinations thereof. At first sight, Gauss–Hermite integration[12] seems to be a promising method because of the e^{-t^2}-factor in the integrand of Eqs. (3, 5, 6). In fact, for increasing $|x|$ and y the number of terms needed to meet a given accuracy requirement decreases. However, as discussed by Armstrong[1] for small y, the integral (3) defining $K(x, y)$ cannot be used for computation since this representation does not yield the correct limit $K(x, 0) = e^{-x^2}$ by simply letting $y \to 0$. Thus, the x, y-plane is generally divided into several regions and an appropriate method is selected for each region.[1,13–16] To avoid time-consuming tests for convergence of the expansions, the number of terms needed can be determined through an empirical relation depending on x and y.[13,16,17] (iv) Finally, there are methods relying on interpolation from precalculated 2-D tables. The algorithm implemented in the ATMOS software by Norton and Rinsland[18] uses four-point bivariate interpolation from a precalculated table for $0 < x < 4$ and $0 < y < 5.5$. The interpolation scheme of Drummond and Streckner[19] is based on a transformation from the x, y coordinate system to a finite grid in the P, S coordinate system with $P = 1/(1 + x)$ and $S = y/(1 + y)$.

4. COMPARISON: ACCURACY

In the last column of Table 1, the accuracies of the algorithms as claimed by the authors are shown. Armstrong's[1] and Gautschi's[17] methods appear to be most accurate. Gautschi originally considers an absolute error criterion but notes that relative errors are not substantially larger for positive x, y except for somewhat larger relative errors of $\mathrm{Im}[w(z)]$ for z close to the imaginary axis and $\mathrm{Re}[w(z)]$ for z close to the real axis with $x \gg 0$. Except when noted, Armstrong's code will be taken as reference for the following comparisons.

For the 3-D error plots shown, we have calculated the Voigt profile on a 2-D grid of 50×25 points in the x, y plane and then evaluated the relative error $(K - K_{ref})/K_{ref}$ of the algorithm under consideration vs the reference algorithm. It should be emphasized that in case of very narrow error spikes, the maximum errors retrieved from the plots depend on the resolution of the x, y grid.

Armstrong's and Gautschi's algorithm are compared in Fig. 3. Except for small values of y and $x \approx 5$, the relative error is 10^{-6} or less. The relative errors of Drayson's,[13] Hui's,[20] Humlicek's,[14,15] Norton's,[18] and Pierluissi's[16] algorithms, shown in Figs. 4–11, essentially confirm the authors' stated accuracy. The boundaries between regions of different approximations appear clearly in Figs. 4, 7, 9, 10. For some of the methods larger deviations can be observed for small y and intermediate x. For example, the 6-point Gauss–Hermite quadrature used by Pierluissi et al for $3 < x < 5$ leads to relative large errors for $x \approx 3$. The largest error of the Norton–Rinsland method, shown in Fig. 9, occurs for x, y in the asymptotic expansion region, whereas it is almost an order of magnitude smaller in the interpolation region.

In Hui's algorithm, the rational approximation cannot reproduce the exponential decrease e^{-x^2} for vanishing y. In order to avoid this deficiency, we followed Karp's[10] suggestion and replaced Hui's rational approximation for the real part of the complex error function by

$$K(x, y) = e^{-x^2} + \frac{y/\sqrt{\pi}}{x^2}$$

for small y and $y/x^2 < 10^{-4}$. As is evident from Armstrong's[1] Eqs. (19, 21), this is basically the first term of a series expansion in y and asymptotic expansion in x. However, according to Fig. 5, the

Table 1. Overview of different algorithms: the programs of Gautschi, Hui et al, and Humlicek compute the complex error function; all other programs compute the Voigt function.

Author(s)	Region	Method	Stated Accuracy
	$y < 1.0, \quad x < 4$ or $y < \frac{1.8}{x+1}, \quad x > 4$	Hummer-Faddeyeva/Terent'ev power series	
Armstrong	$1 < y < 2.5, \quad x \leqslant 4$	20-term Gauss-H.-integration of modified integrand	1–2 digits in 6th significant figure
	$y > 2.5, \quad x < 4$ or $y > \frac{1.8}{x+1}, \quad x > 4$	20-term Gauss-Hermite-integration	
Drayson†	$y < 1.0, \quad x < 5 - 0.8y$ $y > 1.0, \quad x < 1.85(3.6 - y)$ else, $y < 11.0 - 0.6875$ else, $y \geqslant 11.0 - 0.6875$	Taylor and Chebyshev expansion continued fraction 4-point Gauss-Hermite-integration 2-point Gauss-Hermite-integration	one part in 10^4
Drummond & Streckner	all x, y	two-dimensional interpolation	rel. error $< 3 \cdot 10^{-3}$ (400 by 100 table)
Gautschi‡	all x, y	truncated Taylor expansion (reduces to Gauss-H. quadrature for large $\|z\|$)	10 decimal places after decimal point
Hui et al	all x, y	rational approximation	rel. error $< 10^{-6}$, precise within 5 or 6 significant figures
Humlicek (1979)	$y > 0.85$ or $\|x\| < 18.1y + 1.65$ else	rational approximation rational approximation, modified for Re(w)	$2 \cdot 10^{-6}$ for real, $5 \cdot 10^{-6}$ for imaginary part
Humlicek (1982)	$\|x\| + y \geqslant 15$ $5.5 \leqslant \|x\| + y < 15$ $\|x\| + y < 5.5$ and $y \geqslant 0.195\|x\| - 0.176$ else	rational approximations	10^{-4} relative error
Karp	all x, y	Fourier transform method	
Kielkopf	all x, y	empirical fit	10^{-4} relative to peak value
Norton & Rinsland§	$0 \leqslant x < 4, \quad 0 < y < 5.5$ $x \geqslant 14$ or $y \geqslant 22$ else	bivariate linear interpolation Lorentz function 2-term asymptotic expansion	error $\leqslant 0.001$ relative to line center value
Pierluissi et al	$0 \leqslant x < 3, \quad 0 \leqslant y < 1.8$ $3 \leqslant x < 5, \quad 1.8 \leqslant y < 5$ $x \geqslant 5, \quad y \geqslant 5$	series expansion 6-point Gauss-Hermite integration 4-point Gauss-Hermite integration	peak deviation $\leqslant 1.4 \cdot 10^{-3}$ RMS deviation $\leqslant 1.4 \cdot 10^{-5}$
Whiting	all x, y	empirical fit	5% at worst

†Note that different specifications for Drayson's region I are given in the text and in the source code.
‡Gautschi's procedure is implemented in the IMSL SFUN/LIBRARY as Fortran functions CERFE or ZERFE in single or double precision, respectively.
§The specifications for the Norton–Rinsland method were taken from the source code provided by C. Rinsland.

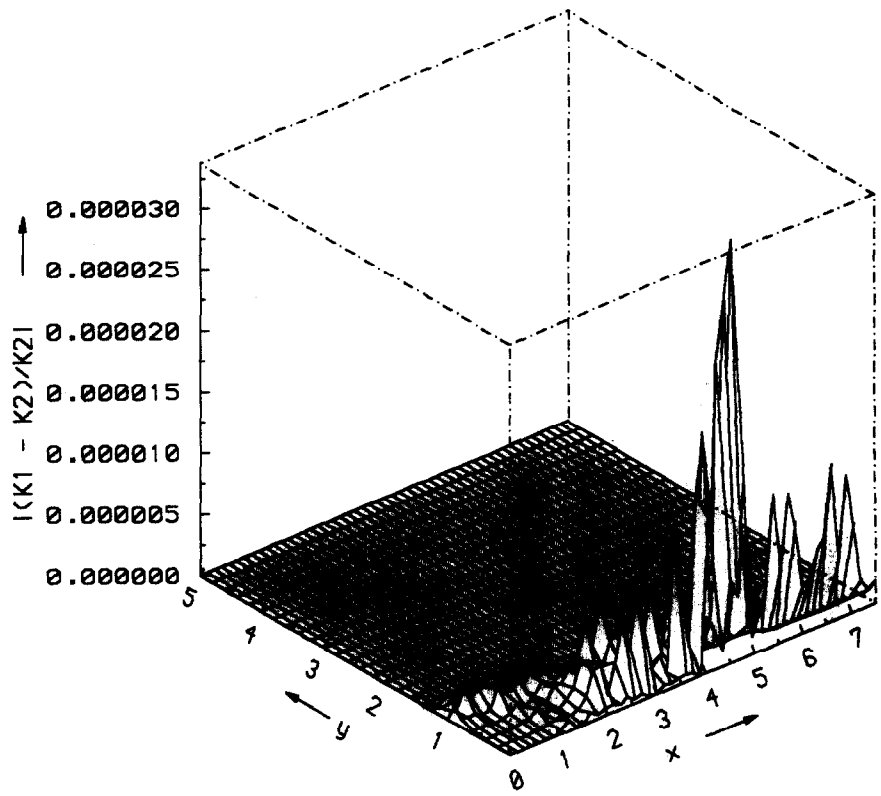

Fig. 3. Relative error of Voigt function: Armstrong's (K_1) vs Gautschi's (K_2) algorithm (IMSL-ZERFE).

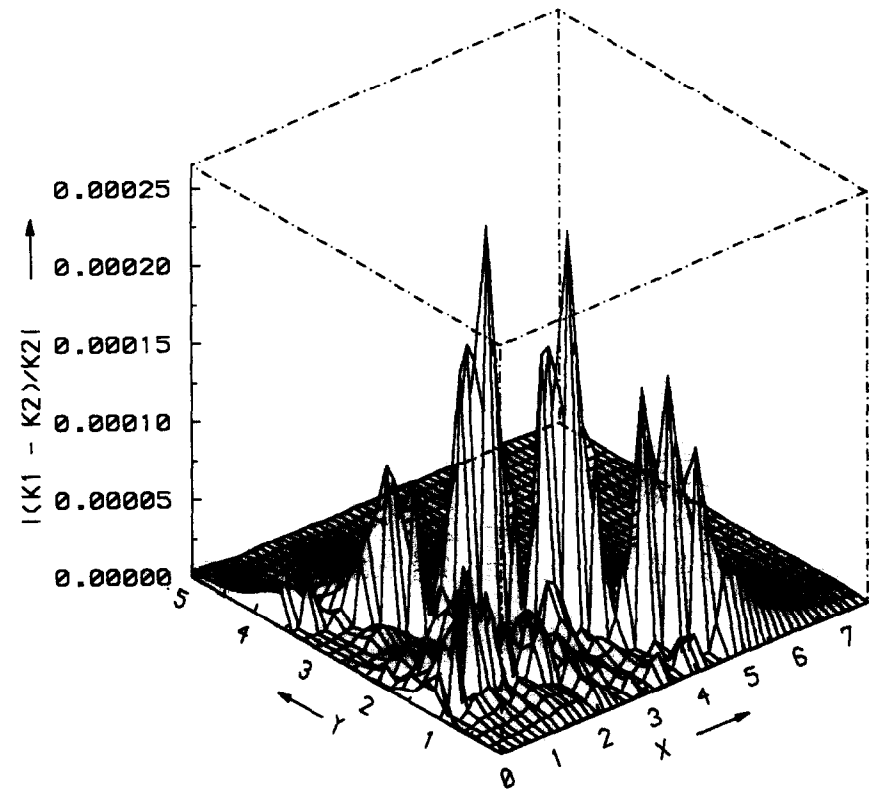

Fig. 4. Relative error of Voigt function: Drayson's vs Armstrong's algorithm [for $y = 0$, we have complemented Drayson's algorithm by $K(x, 0) = e^{-x^2}$].

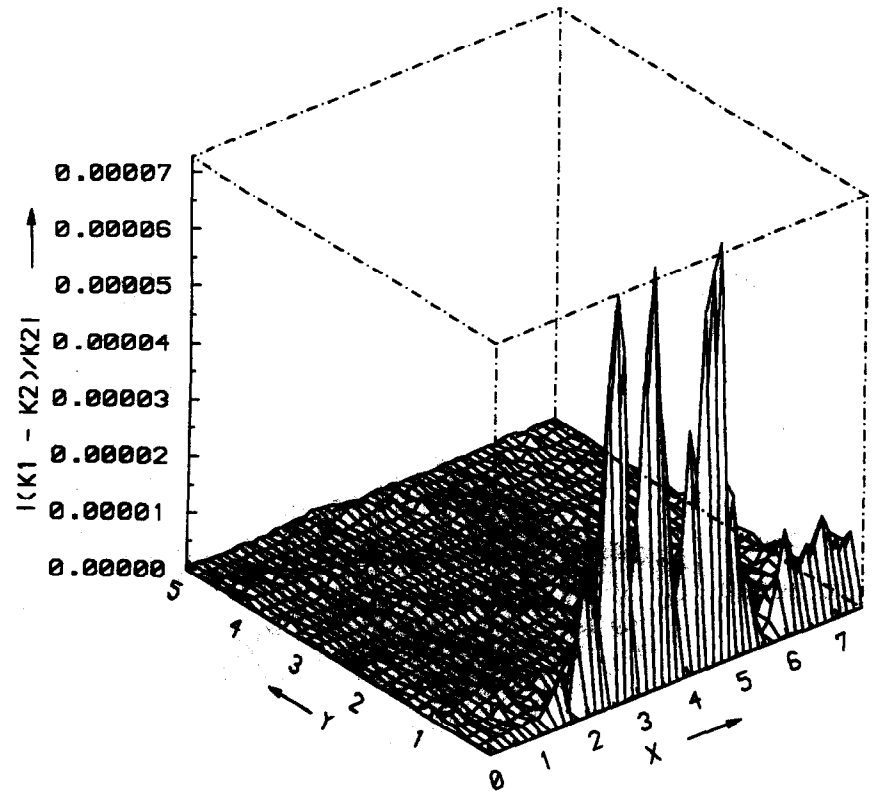

Fig. 5. Relative error of Voigt function: Hui's vs Armstrong's algorithm [for $y/x^2 < 10^{-4}$, the rational approximation has been replaced by $K(x, y) = e^{-x^2} + y/\sqrt{\pi}/x^2$, as suggested by Karp].

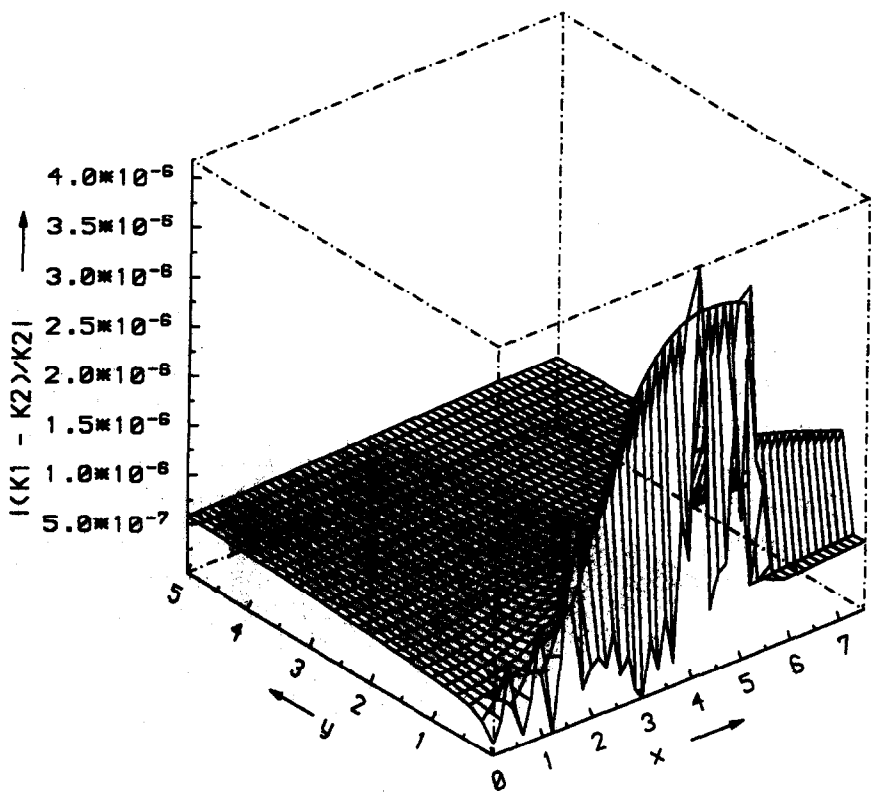

Fig. 6. Relative error of Voigt function: Humlicek's (1979) vs Gautschi's algorithm (IMSL-ZERFE).

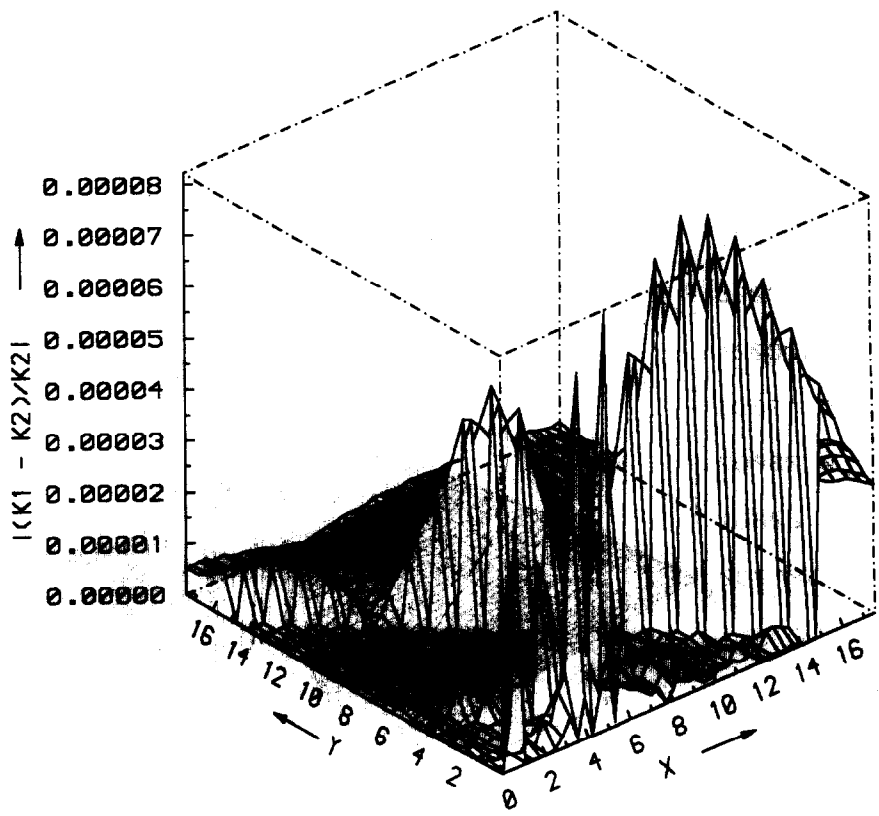

Fig. 7. Relative error of Voigt function: Humlicek's (1982) vs Armstrong's algorithm.

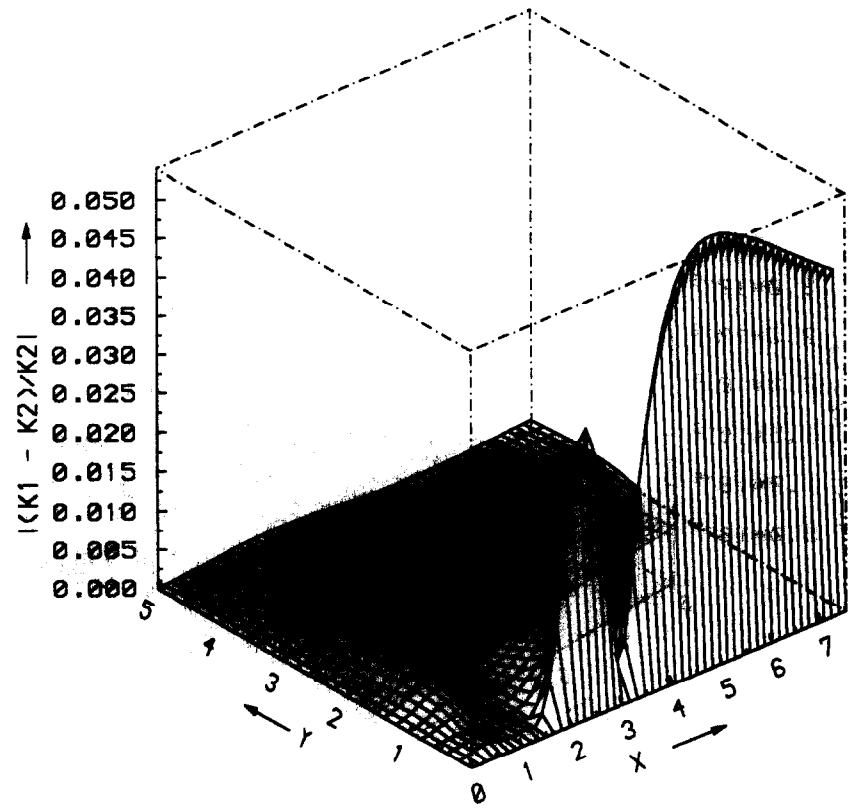

Fig. 8. Relative error of Voigt function: Kielkopf's vs Armstrong's algorithm.

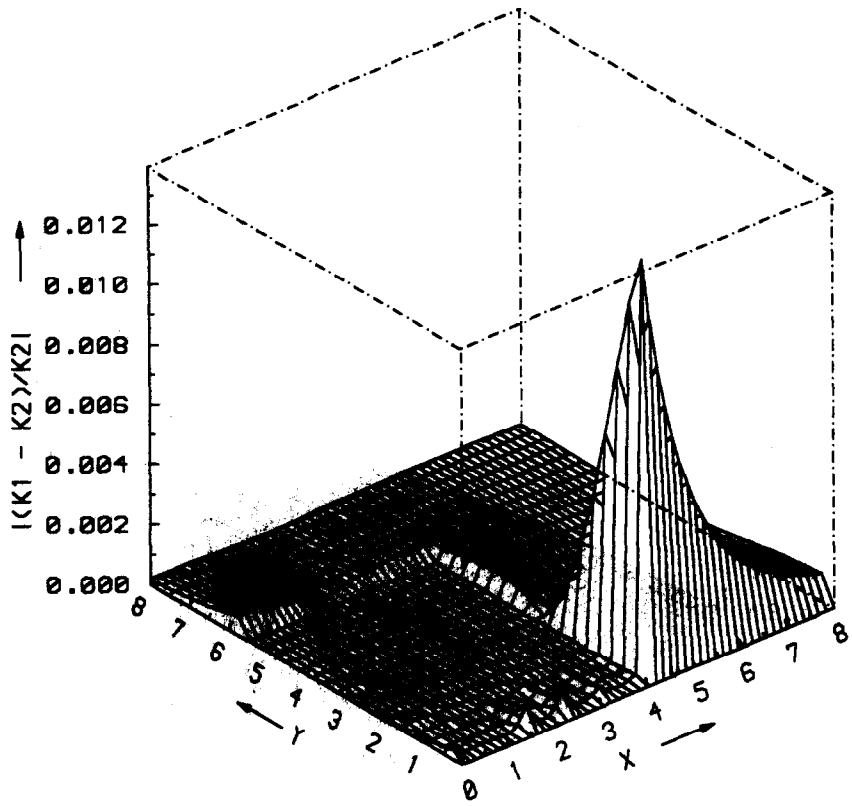

Fig. 9. Relative error of Voigt function: Norton–Rinsland vs Armstrong's algorithm.

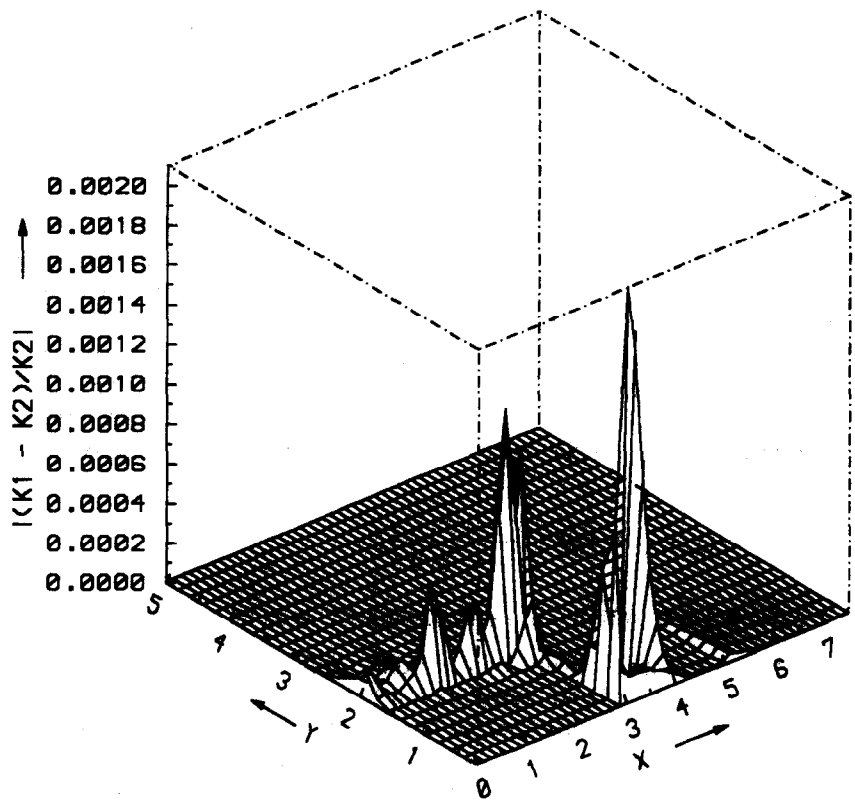

Fig. 10. Relative error of Voigt function: Pierluissi's vs Armstrong's algorithm (Pierluissi's code changed according to the suggestion of Twitty et al).

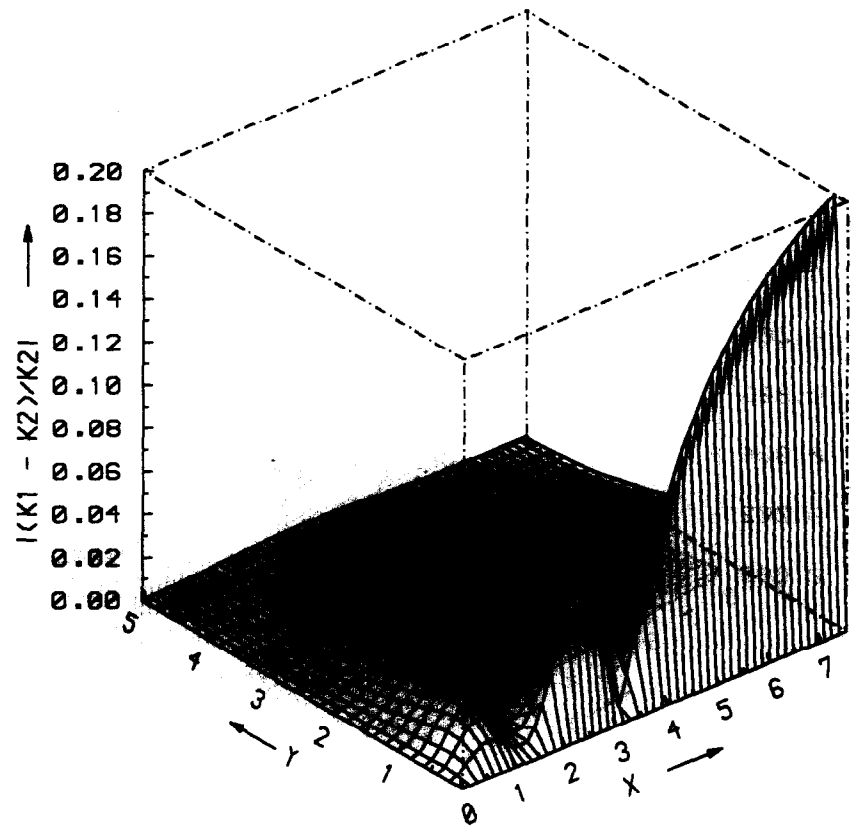

Fig. 11. Relative error of Voigt function: Whiting's vs Armstrong's algorithm.

largest deviations occur for $x \approx 3$–5 and small y, where an asymptotic expansion provides an inefficient alternative to Hui's approximation.

Humlicek's CPF 12 algorithm,[14] that originally had been written in single precision, has been converted to REAL*8 because of the limited accuracy of only about 6 or 7 figures of IBM single precision numbers. In addition we also used higher precision abscissas and weights[6] of the Gauss–Hermite quadrature rule and accordingly recalculated the constants α_k and β_k used in his program. For small y and $x \approx 4$–5 discrepancies have been found between Humlicek's and Armstrong's profile one magnitude larger than expected, but the relative error with respect to Gautschi is of the order of 10^{-6} or less.

Compared with other methods the empirical approaches of Kielkopf and Whiting give only a rough estimate of the Voigt profile. This is especially true for Whiting's formula, which yields a relative error of more than 10% in the line wings. Replacing Kielkopf's or Whiting's approximations for the Voigt half width by the more accurate (to within 0.01%) expression of Olivero and Longbothum,[7]

$$x_{1/2} = (y + \sqrt{\ln 2})\, R\left(\frac{y - \sqrt{\ln 2}}{y + \sqrt{\ln 2}}\right),$$
$$R(d) = 1 - 0.1821(1 - d^2) + (0.023665\, e^{0.6d} + 0.00418\, e^{-1.9d})\sin \pi d, \tag{10}$$

did not yield a significant improvement for the Voigt profile.

The relative large deviations of both methods can be easily understood in view of Armstrong's discussion of regions with "pure" Lorentzian or Doppler behaviour.[1] In brief, both the parameters x and y have to be chosen appropriately to specify the pure cases (compare Armstrong's Fig. 1). However, in Kielkopf's or Whiting's first approximation, the relative weight on the Lorentzian and Gaussian contributions is a function of the ratio of Lorentzian and Doppler widths, $y \sim \gamma_L/\gamma_D$, alone.

Some of the programs considered compute the complex error functions (5, 8) and obtain the Voigt function as its real part. As will be discussed in Sec. 7, the imaginary part of the complex error function (6) can be used to determine the partial derivatives of the Voigt function with respect to x and y. Furthermore, when line coupling (line mixing)[21] is taken into account with a modified Lorentzian line shape (Y is the coupling coefficient),

$$g_{L'}(\tilde{\nu}) = \frac{1}{\pi}\frac{\gamma_L + Y(\tilde{\nu} - \tilde{\nu}_0)}{(\tilde{\nu} - \tilde{\nu}_0)^2 + \gamma_L^2}, \tag{11}$$

the resulting modified Voigt profile can be readily obtained from the real and imaginary part of the complex function (5) according to

$$g_{V'} = \frac{\sqrt{\ln 2/\pi}}{\gamma_D}[K(x, y) + YL(x, y)]. \tag{12}$$

A comparison of the accuracies of $L(x, y)$ is given in Figs. 12–14, where the relative error with respect to the IMSL-ZERFE routine, corresponding to Gautschi's algorithm, is plotted.

5. COMPARISON: COMPUTING TIMES

For a comparison of computational speed, Twitty et al[3] divided the x, y-plane in several regions common to the algorithms to be tested and determined the times for each of these regions. However, rather than calculating the Voigt profile for such limited sets of x, y values, most applications require the Voigt profile for an array of x mesh points and some specified y values according to the Lorentzian and Doppler half widths γ_L, γ_D. We therefore considered two sets of y-values, $0 < y < 1$ and $1 < y < 10$, corresponding to Doppler and Lorentzian "dominance" (compare Armstrong[1] for a discussion of "pure" Doppler or Lorentz regions). For each of 50 y values in these intervals, we calculated the Voigt profile for 1000 grid points in the interval $0 \leqslant x_i \leqslant x_{max}$ with $x_{max} = 10x_{1/2}(y)$.

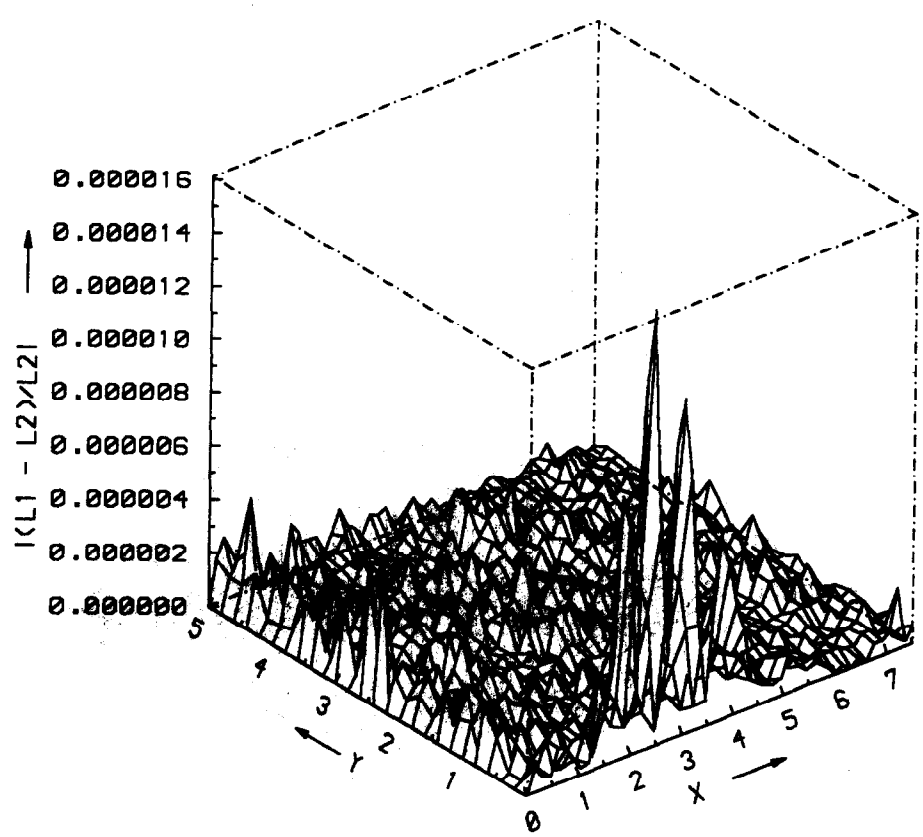

Fig. 12. Relative error of imaginary part of complex error function: Hui vs Gautschi (IMSL-ZERFE).

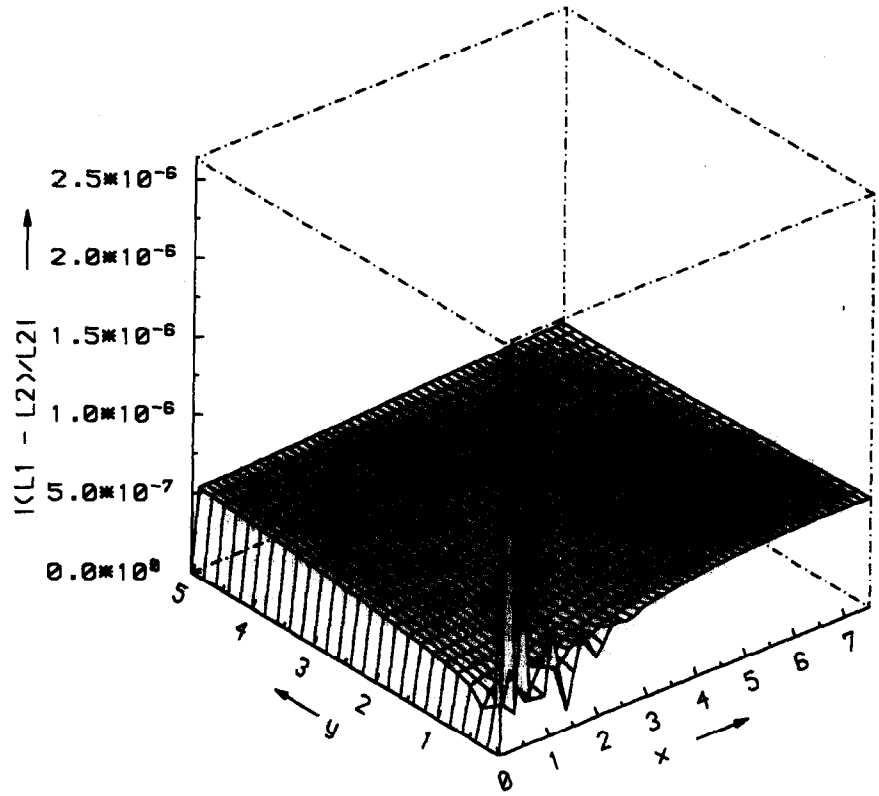

Fig. 13. Relative error of imaginary part of complex error function: Humlicek (1979) vs Gautschi (IMSL-ZERFE).

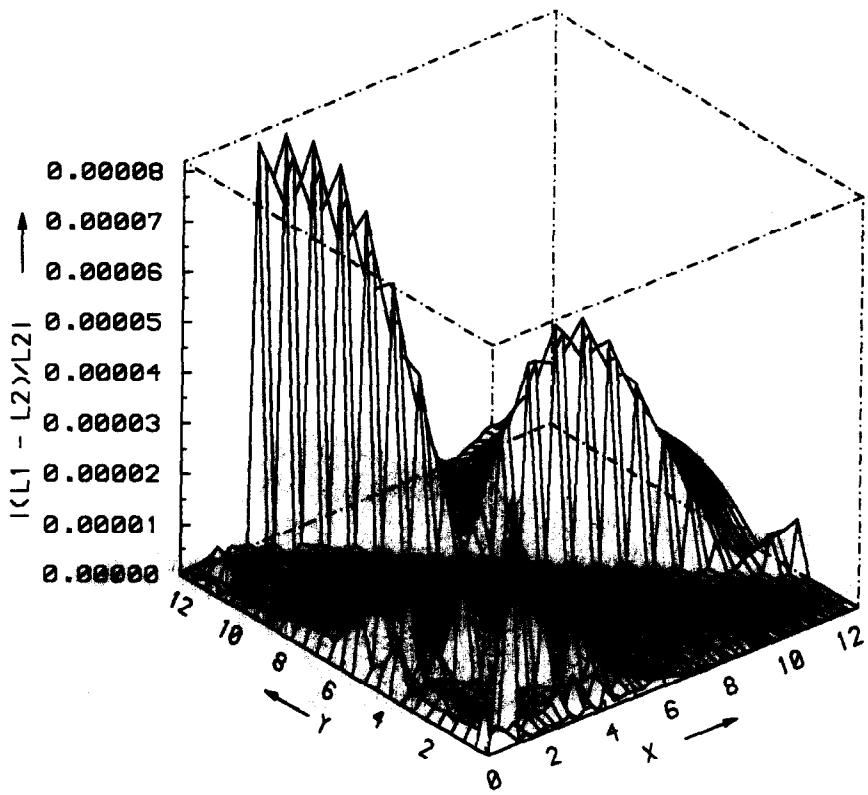

Fig. 14. Relative error of imaginary part of complex error function: Humlicek (1982) vs Gautschi (IMSL-ZERFE).

No significant difference in computer time was found for smaller and larger values of y ($0 < y < 0.1$ or $10 < y < 100$) or for larger $x_{max} = 20x_{1/2}(y)$. Furthermore, the times varied linear with the number of grid points in the x, y-plane. The CPU time used by each function or subroutine has been determined by the IMSL routine *CTIME*.[22] Jobs were run on a IBM 3090-30S VF(2) under MVS/XA operating system and with VS FORTRAN Version 2 Release 4.0 Compiler. The CRAY Y-MP2/232 runs under UNICOS 6.0 operating system with a CFT77 compiler. Highest optimization level has been used on both systems. Because of the higher accuracy of Cray's REAL type numbers, double precision declaratives REAL*8 are treated as real, single precision.

Initially, we consider the programs with only minor modifications of the original source codes. Arithmetic IF statements and GOTOs have been replaced by IF ... THEN ... ELSE ... END IF structures of FORTRAN77. Pierluissi's code has been modified according to the suggestion of Twitty et al. The subsequent call of functions K1, K2, K3 corresponding to the three computational regimes of Armstrong's algorithm has been replaced by in-line code inserted into the body of block-if structures. All algorithms are coded as FUNCTIONs, except Kielkopf's[8] and Whiting's,[9] that require the calculation of some parameters such as half widths and weight factors common to all mesh points x_i.

The times required for 1000×50 evaluations are listed in Table 2. The high accuracy of Armstrong's method[1] is achieved with a large amount of computing time. However, the IMSL CERFE routine,[22] representing a FORTRAN implementation of Gautschi's[17] ALGOL procedure, is even slower. In contrast the simple ansatz of Kielkopf or Whiting results in significant less computing time with Whiting's method being the slower of the two because of the exponentation with 2.25. Contrary to the statement of Pierluissi et al[16] we found Drayson's algorithm[13] to be faster than Pierluissi's. For small x, y (Pierluissi's region I) the execution became slower due to the increased number of terms used in the series expansion. On the other hand, for large x, y 11 multiplications and two divisions are required for Pierluissi's region III, whereas only four multiplications and two divisions are required for Drayson's region IIIb (6 multiplications and 4 divisions in IIIa). Furthermore, on the IBM double precisions arithmetic is utilized in Pierluissi's code. Thus, especially when many evaluations for the Voigt line wings are required, Drayson's code is faster than Pierluissi's.

In summary the methods of Drayson,[13] Hui, Armstrong and Wray,[20] Humlicek[15] or Pierluissi, Vanderwood and Gomez[16] represent good alternatives for relatively fast and accurate algorithms.

The advent of supercomputers has drawn much attention to the vectorizability of frequently needed program units. With regard to Voigt profile calculations the approach of Hui et al[20] promises to be a highly accurate algorithm which can easily be vectorized. Therefore, we have coded this algorithm as SUBROUTINE including a DO loop over all mesh points $X(I)$, that yields two real arrays $VGTK(I)$, $VGTL(I)$ for $I = 0, \ldots, NX$ (or one complex array) as output. As is shown in Table 3, this modification results in an execution two orders of magnitude faster on the Cray, whereas only a factor of 5 has been gained on the IBM.

A further acceleration can be achieved by observing that a complex polynomial in $\tilde{z} = -iz = y - ix$ with real coefficients c_m can be written as a polynomial in x with complex

Table 2. Computing times for 1000 mesh points in $0 < x < 10 \cdot x_{1/2}(y)$ and 50 y: all methods are coded as FUNCTIONs except for Kielkopf's and Whiting's.

	Time (sec)			
	IBM 3090		CRAY Y-MP2	
Algorithm	$0 < y < 1$	$1 < y < 10$	$0 < y < 1$	$1 < y < 10$
Armstrong	1.407	1.070	2.042	1.555
Drayson	0.428	0.227	1.612	1.427
Hui et al	0.478	0.476	1.467	1.465
Humlicek 79 (REAL*4)	0.968	0.848	1.554	1.522
Humlicek 82	0.480	0.310	1.542	1.431
IMSL CERFE	5.763	3.282	4.167	2.675
Kielkopf (vector)	0.051	0.051	0.015	0.015
Kielkopf (novector)	0.172	0.173	0.148	0.147
Pierluissi et al	0.610	0.247	1.603	1.427
Whiting (vector)	0.150	0.149	0.037	0.036
Whiting (novector)	0.355	0.359	0.413	0.410

Table 3. Computing times for various implementations of Hui's method (1000 × 50 points and $0 < x < 10 \cdot x_{1/2}$). Except for the rows labeled $p = 5$, Hui's rational approximation with a polynomial of order $p = 6$ for the nominator has been used. Karp's modification is used only for $y \leqslant 0.1$.

	Time (sec)			
	IBM 3090		CRAY Y-MP2	
Algorithm	$0 < y < 1$	$0 < y < 10$	$0 < y < 1$	$1 < y < 10$
Function	0.478	0.476	1.469	1.465
Subroutine	0.093	0.093	0.020	0.020
Modified subroutine	0.037	0.037	0.010	0.010
Mod. subroutine, novector	0.143	0.144	0.059	0.059
Mod. subroutine, Karp's mod.	0.040	0.037	0.010	0.010
Mod. subr., Karp, novector	0.146	0.143	0.063	0.059
Function, p = 5	0.432	0.430	1.457	1.455
Subroutine, p = 5	0.083	0.083	0.018	0.017

coefficients $d_m = d'_m + id''_m$ depending on y, where the coefficients for even powers of x are real and the coefficients of odd powers of x are purely imaginary,

$$P(\tilde{z}) = \sum_{m=0}^{M} c_m(y - ix)^m = \sum_{k=0}^{M} d_k(y)x^k = \sum_{k\,\text{even}} d'_k x^k + i \sum_{k\,\text{odd}} d''_k x^k. \tag{13}$$

The d_k are polynomials in y with its coefficients determined as products of c_m and binomial coefficients,

$$d_k = (-i)^k \sum_{l=0}^{M-k} \binom{l+k}{k} c_{l+k} y^l. \tag{14}$$

Thus, instead of 12 complex multiplications and one complex division, equivalent to 54 real multiplications and one real division, for the complex rational approximation,

$$w(z) = \frac{\sum_{m=0}^{6} a_m(y - ix)^m}{\sum_{n=0}^{7} b_n(y - ix)^n} \quad \text{with} \quad b_7 = 1, \tag{15}$$

we need only 21 real multiplications and one real division for every mesh point x_i. This is reflected by approximately doubling the computational speed of the modified code. However, the modified routine is efficient only when a large number ($NX > 10$) of data points is to be evaluated, i.e. when the 78 multiplications needed to calculate the polynomial coefficients d_k, Eq. (14), are negligible. It should be emphasized that the implementation of Hui's rational approximation (15) as a subroutine and the rearrangement of the complex polynomials (13) also leads to a significant reduction of computational time on non-vectorizing machines, as also shown in Table 3. If the calculation of x and y is included in the considerations of computing times, a further optimization of Hui's method is easily possible as shown in the following chapter.

Vectorization is not confined to simple DO loops as in Hui's code, but can also be performed in DO loops containing conditional IF branches[23] such as in Humlicek's algorithms.[14,15] Thus, we have written Humlicek's CPF 12 algorithm as SUBROUTINE with the sum over K as outer loop and the grid points x_i as inner vectorizable loop. Similar to Hui's program, the computational gain, shown in Table 4, is more significant for the Cray than for the IBM.

Straightforward vectorization of Humlicek's newer W4 algorithm containing a quadruple alternative block-IF statement results in a significant improvement on the Cray only (Table 4). However, it is obvious from Table 1, that for $y \geqslant 15$ all mesh points x_i are located in region I and IF branches can be avoided in this case. Similarly, for $5.5 \leqslant y < 15$, all mesh points are either in region I or II, and only two cases need to be considered. Restructuring the code according to this observation results in a further acceleration for $y > 1$.

A saving of two orders of magnitude of Cray run time is also achieved for the Norton–Rinsland method[18] if the function is rewritten as subroutine with a vectorizable loop over the x grid. Note,

Table 4. Computing times for various implementations of Humlicek's CPF12 and W4 algorithms, shown in the first and second sections, respectively. Except for the first row, double-precision arithmetic has been used in the CPF 12 routine (1000×50 points and $0 < x < 10 \cdot x_{1/2}$).

	Time (sec)			
	IBM 3090		CRAY Y-MP2	
Algorithm	$0 < y < 1$	$1 < y < 10$	$0 < y < 1$	$1 < y < 10$
Function, region I and II, real*4	0.967	0.848	1.547	1.509
Function, region I and II	1.281	1.221		
Subroutine, only region I	0.183	0.184	0.036	0.036
Subroutine, region I and II	0.454	0.175	0.054	0.036
Subroutine, I and II, novector	0.885	0.740	0.449	0.362
Function	0.480	0.310	1.542	1.431
Subroutine	0.339	0.283	0.029	0.016
Subroutine, novector	0.368	0.207	0.170	0.072
Restructured subroutine	0.357	0.165	0.027	0.009

however, that the loop was not vectorized on the IBM because of indirect addressing in conditionally processed code (Table 5).

6. FOURIER TRANSFORM METHODS

An interesting computational approach is provided by the simple representation of the Voigt function in the Fourier transform domain in combination with highly efficient Fast Fourier Transform (FFT) algorithms.[12] Furthermore it has been emphasized by Karp[10] that, for an absorption coefficient with many contributing lines, only two transforms are required because of the frequency shift properties of the Fourier transformation. However, for the evaluation of the absorption coefficient at n_v grid points in the wave number domain, the same number of evaluations is necessary in the Fourier domain, i.e. for n_l contributing lines we have $n_l n_v$ evaluations of the Fourier transformed Voigt function. Therefore, for a comparison of the computational efficiency of Fourier transform and other methods a more detailed discussion will be necessary.

The total absorption coefficient for n_l lines, defined by positions $\tilde{\nu}_l$, line strengths S_l, Lorentzian and Doppler and half widths γ_l^{L}, γ_l^{D}, is given by the sum

$$k(\tilde{\nu}) = \sum_{l=1}^{n_l} S_l \frac{\sqrt{\ln 2/\pi}}{\gamma_l^{\mathrm{D}}} K(x_l, y_l), \tag{16}$$

with the line profiles described by the Voigt function $K(x_l, y_l)$ according to Eqs. (3, 4). Inserting the Fourier representation

$$K(x, y) = \frac{1}{\sqrt{\pi}} \int_0^\infty \mathrm{e}^{-q^2/4 - yq} \cos qx \, \mathrm{d}q \tag{17}$$

and introducing the new parameters

$$\delta_l \equiv \gamma_l^{\mathrm{D}}/\sqrt{\ln 2} \quad \text{and} \quad \lambda_l \equiv \gamma_l^{\mathrm{L}},$$

one finds after some algebraic and trigonometric manipulations and a change of variables $qx_l = q(\tilde{\nu} - \tilde{\nu}_l)/\delta_l = p(\tilde{\nu} - \tilde{\nu}_l)$ that

$$k(\tilde{\nu}) = \frac{1}{\pi} \int_0^\infty \mathrm{d}p \cos p\tilde{\nu} \left[\sum_l S_l \mathrm{e}^{-(p\delta_l)^2/4 - p\lambda_l} \cos p\tilde{\nu}_l \right] + \frac{1}{\pi} \int_0^\infty \mathrm{d}p \sin p\tilde{\nu} \left[\sum_l S_l \mathrm{e}^{-(p\delta_l)^2/4 - p\lambda_l} \sin p\tilde{\nu}_l \right]. \tag{18}$$

Table 5. Computing times for various implementations of Norton and Rinsland's interpolation scheme (1000×50 points and $0 < x < 10 \cdot x_{1/2}$).

	Time (sec)			
	IBM 3090		CRAY Y-MP2	
Algorithm	$0 < y < 1$	$1 < y < 10$	$0 < y < 1$	$1 < y < 10$
Function	0.185	0.153	1.415	1.399
Subroutine, vector			0.010	0.007
Subroutine, novector	0.081	0.066	0.077	0.062

Thus, the time required for the computation of the absorption coefficient is essentially determined by two FFTs and the number of operations to compute the expressions enclosed in square brackets, which will be approximately proportional to the number of lines and the number of grid points in the Fourier domain:

$$t_{[\ldots]} = n_l n_p (t_{\exp} + 2t_{\sin} + 6t_{\text{mult}} + 3t_{\text{add}}).$$

For equidistant grid points, $p_j = j\Delta p$, recursive relations can be utilized for the computation of the exponential and trigonometric functions, and the time required for the calculation of the absorption coefficient will be approximately given by

$$t_{\text{Karp}} = n_l n_p (12t_{\text{mult}} + 4t_{\text{add}}) + n_l (2t_{\exp} + 2t_{\sin} + 6t_{\text{mult}}) + 2n_\nu \log_2 n_\nu \tau_{\text{FFT}}, \tag{19}$$

where τ_{FFT} characterizes a time typical for the FFT.

The discussion of Hui's algorithm,[20] which proved to be accurate, fast and computationally simple, was based on the complex error function as a function of x and y. For a detailed comparison with the Fourier transform approach a reformulation in terms of the total absorption coefficient is appropriate. Inserting the real part of the complex rational approximation (15) into Eq. (16), we find

$$k(\tilde{\nu}) = \sum_{l=1}^{n_l} \frac{S_l}{\sqrt{\pi}\,\delta_l} \operatorname{Re} \frac{\sum_{m=0}^{6} a_m (y_l - ix_l)^m}{\sum_{n=0}^{7} b_n (y_l - ix_l)^n}. \tag{20}$$

The linear transformation from wave numbers $\tilde{\nu} - \tilde{\nu}_1$ to the Voigt parameter x_l and the scaling of the Voigt profiles with line strengths S_l etc., which would be required for all wave number grid points and all lines, can be avoided by defining the new coefficients

$$\begin{aligned} \alpha_{ml} &\equiv S_l a_m / \sqrt{\pi}\,\delta_l^{m+1}, \\ \beta_{nl} &\equiv b_n / \delta_l^m . \end{aligned} \tag{21}$$

The absorption coefficient can then be expressed as

$$k(\tilde{\nu}) = \sum_{l=1}^{n_l} \operatorname{Re} \frac{\sum_{m=0}^{6} \alpha_{ml} [\lambda_l - i(\tilde{\nu} - \tilde{\nu}_1)]^m}{\sum_{n=0}^{7} \beta_{nl} [\lambda_l - i(\tilde{\nu} - \tilde{\nu}_l)]^n}. \tag{22}$$

Using the rearrangement of the complex polynomials as discussed in the preceding chapter, Hui's approach will take a time

$$t_{\text{Hui}} = n_l n_\nu (16t_{\text{mult}} + t_{\text{div}} + 14t_{\text{add}}) + n_l (28t_{\text{mult}} + t_{\text{div}} + t_{\text{add}}). \tag{23}$$

Note that Hui's rational approximation with a polynomial of order $p = 5$ in the nominator will take $15 n_l n_\nu$ multiplications.

Comparing Eqs. (19) and (23) with $n_p = n_\nu$, the additional division and some more multiplications required for Hui's rational approximation (22) must be weighed against two Fourier transforms and $2n_l$ evaluations of trigonometric and exponential functions in Eq. (18). Vectorization of Hui's algorithm is possible with respect to wave number grid points or spectral lines, whereas the computation of the sine and cosine transforms in Eq. (18) can be vectorized only with respect to lines because of the recursions used for the exponential and trigonometric functions. Furthermore, FFT algorithms such as described in Numerical Recipes[12] are recursive and not vectorizable, but highly optimized FFT routines are implemented on computers such as the Cray. Fourier transform methods will be especially advantageous when hardware FFT processors are available. For completeness, problems inherent to Fourier transform methods, such as the restriction for the number of grid points to powers of two or error sources like aliasing, should also be mentioned here (compare Karp[10] for a detailed discussion).

7. DERIVATIVES OF THE VOIGT FUNCTION

Nonlinear least squares fitting algorithms are frequently used to retrieve molecular line parameters (line position, strength and width) from high resolution spectra. Iterative solution procedures based on Gauss–Newton- or related methods require the partial derivatives with respect to the line parameters.[12] This implies the necessity to calculate the Voigt function $K(x, y)$ as well as the partial derivatives with respect to x and y.

An efficient method for calculation of these derivatives is provided by the integral representation of the Voigt function in the complex plane. This has already been indicated in the paper of Humlicek,[14] but statements that partial derivatives of the Voigt function are difficult to obtain and costly in computer time can still be found in the literature.[24]

With Eqs. (3, 6) for the real and imaginary part of the complex error function one can easily prove that $w(z)$ satisfies the Cauchy–Riemann conditions. Using the differential equation satisfied by $w(z)$,

$$w'(z) = \frac{2i}{\sqrt{\pi}} - 2zw(z), \tag{24}$$

one finds

$$\frac{\partial K(x,y)}{\partial x} = -2\mathrm{Re}[zw(z)], \quad \frac{\partial K(x,y)}{\partial y} = +2\mathrm{Im}[zw(z)] - \frac{2}{\sqrt{\pi}}. \tag{25}$$

These relations can also be verified by direct differentiation of Eq. (3) and partial integration.

The advantage of using Eq. (25) for the derivatives becomes especially obvious considering the calculation of derivatives using finite differences. In case of three parameters $\tilde{\nu}_0$, S, γ_L this requires three additional calculations of the Voigt profile for slightly shifted parameter values $\tilde{\nu}_0 + \delta\tilde{\nu}_0$, etc. However near an extremum with respect to one of these parameters a two-point finite difference approximation (forward or backward) will not yield reliable derivatives and more accurate and time consuming approximations (three-point differences etc.) have to be used. Thus by the use of algorithms for the complex error function and of Eq. (25) the computer time spent for the evaluation of the Voigt profile and its derivatives can be reduced by approx. 75% or more.

For the numerical evaluation of the derivatives using Eq. (25) it should be noted that $\mathrm{Re}(zw) = xK - yL$ involves subtracting two numbers of approximately equal magnitude. This can be easily understood because the main contribution to the integrals in Eqs. (3, 6) is for small t. Neglecting t in the $(x - t)$ factor of the nominator of

$$yL = \frac{y}{\pi}\int_{-\infty}^{+\infty} \frac{(x-t)\mathrm{e}^{-t^2}}{(x-t)^2 + y^2}\,\mathrm{d}t,$$

immediately yields xK. A similar observation can be made for $\partial K/\partial y$ because $xL + yK \approx 1/\sqrt{\pi}$.

8. CONCLUDING REMARKS

We have compared different Voigt function algorithms with respect to their accuracy and computational speed. The accuracy of most algorithms can be regarded as satisfactory. For applications, however, further aspects have to be considered. Deviations from the Lorentzian line shape function, such as asymmetric lines or sub- and super-Lorentzian line wings, have been observed.[25,26] For atmospheric line-by-line radiance and transmittance models other error sources arise from the limited accuracy of molecular line parameters, especially line strength S and air-broadened half width γ_L.[5] Furthermore, there are uncertainties from the exact temperature dependence of the air-broadened half width. For applications involving least-squares-fit procedures in molecular spectroscopy higher accuracy is especially important when the complex error function is also used to calculate the partial derivatives of the Voigt function.

Computational speed of the various programs varied over two order of magnitudes. Black-box approaches such as IMSL's CERFE function offer a quick solution for single applications, but can be very time consuming, when thousands of evaluations are required routinely. Vectorization of Voigt function evaluations results in a significant acceleration of the code. However, even for non-vectorizing machines restructuring of the program can yield a computational gain. Furthermore exploiting the complex error function for the calculation of derivatives of the Voigt function yields an additional increase in performance.

An important area of application of Voigt function routines are radiative transfer line-by-line models (see Armstrong and Nicholls[2] for further examples). High computational efficiency of these models is also important because a fine enough wavenumber grid has to be chosen such that the narrowest line will be adequately sampled. However a proper treatment of the line wings requires the calculation of the absorption for distances up to typically 25 cm^{-1} from the linecenter, where the line shape is slowly varying. In the Air Force Geophysics Laboratory's FASCODE model[27,28] this dilemma is solved by first decomposing the Lorentz function into sub-functions of similar functional behaviour and increasing widths,[29] and by approximating the Voigt profile by an extension of Whiting's method. In Edwards' GENLN2 code,[21] the line-by-line calculation proceeds in two stages, both utilizing Humlicek's W4 algorithm:[15] a 'wide-mesh' calculation for the slowly varying line-wing and continuum contributions and a 'fine-mesh' calculation for the rapidly changing portions of the line close to the line center. Mankin's line-by-line model[11] employs a Fourier transform method; the assumption of constant line shape and line widths significantly reduces the computational effort for the Fourier transformed absorption coefficient, and most of the time is taken in the Fast Fourier Transform.

In view of line-by-line modelling for atmospheric radiative transfer as well as molecular spectroscopy the algorithms of Hui et al[20] and Humlicek[15] can both be recommended as accurate and fast. However, because Hui's restructured routine is inefficient for a small number of data points, the Humlicek algorithm appears to be more flexible.

Acknowledgements—The author wishes to thank M. Birk and S. Miller for helpful discussions and critical reading of the manuscript. P. Herchenbach of the DLR computer center provided valuable hints for the numerics and R. Färber gave useful support in programming and plotting. Finally, the author thanks C. Rinsland for providing the source code of the Norton–Rinsland interpolation algorithm.

REFERENCES

1. B. H. Armstrong, *JQSRT* **7**, 61 (1967).
2. B. H. Armstrong and R. W. Nicholls, *Emission, Absorption and Transfer of Radiation in Heated Atmospheres*, Pergamon Press, Oxford (1972).
3. J. T. Twitty, P. L. Rarig, and R. E. Thompson, *JQSRT* **24**, 529 (1980).
4. A. Klim, *JQSRT* **26**, 5378 (1981).
5. L. S. Rothman, R. R. Gamache, A. Goldman, L. R. Brown, R. A. Toth, H. M. Pickett, P. L. Poynter, J.-M. Flaud, C. Camy-Peyret, A. Barbe, N. Husson, C. P. Rinsland, and M. A. H. Smith, *Appl. Opt.* **26**, 4058 (1987).
6. M. Abramowitz and I. A. Stegun, "Handbook of Mathematical Functions," National Bureau of Standards, AMS55, New York, NY (1964).
7. J. J. Olivero and R. L. Longbothum, *JQSRT* **17**, 233 (1977).
8. J. F. Kielkopf, *JOSA* **63**, 987 (1973).
9. E. E. Whiting, *JQSRT* **8**, 1379 (1968).
10. A. H. Karp, *JQSRT* **20**, 379 (1978).
11. W. G. Mankin, *Appl. Opt.* **18**, 3426 (1979).
12. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*, Cambridge University Press, Cambridge (1986).
13. S. R. Drayson, *JQSRT* **16**, 611 (1976).
14. J. Humlicek, *JQSRT* **21**, 309 (1979).
15. J. Humlicek, *JQSRT* **27**, 437 (1982).
16. J. H. Pierluissi, P. C. Vanderwood, and R. B. Gomez, *JQSRT* **18**, 555 (1977).
17. W. Gautschi, *Comm. ACM* **12**, 635 (1969); *SIAM J. Num. Anal.* **7**, 187 (1970).
18. R. H. Norton and C. P. Rinsland, *Appl. Opt.* **30**, 389 (1991).
19. J. R. Drummond and M. Steckner, *JQSRT* **34**, 517 (1985).
20. A. K. Hui, B. H. Armstrong, and A. A. Wray, *JQSRT* **19**, 509 (1978).
21. D. P. Edwards, "Modelling of the Atmosphere," *SPIE* **928**, 94 (1988).
22. IMSL Math/LIBRARY User's Manual (version 1.0, 1987); IMSL SFUN/LIBRARY User's Manual (version 2.0, 1987).
23. M. Metcalf, *FORTRAN Optimization*, Academic Press, London (1982).
24. L. R. Brown, J. S. Margolis, R. H. Norton and B. D. Stedry, *Appl. Spectrosc.* **37**, 287 (1983).
25. C. H. Townes and A. L. Schawlow, *Microwave Spectroscopy*, McGraw-Hill, New York, NY (1955).
26. H. Fischer, G. P. Anderson, T. v. Clarmann, S. A. Clough, M. T. Coffey, A. Goldman and F. X. Kneizys, "Intercomparison of Transmittance and Radiance Algorithms" (ITRA), Report of the Limb-Group of the ITRA Workshop at the University of Maryland, March 1986; KfK-report 4349, Kernforschungszentrum Karlsruhe, FRG (1988).

27. S. A. Clough, F. X. Kneizys, L. S. Rothman, and W. O. Gallery, *SPIE* **277**, 152 (1981).
28. S. A. Clough, F. X. Kneizys, E. P. Shettle, and G. P. Anderson, "Atmospheric Radiance and Transmittance: FASCOD2," in *Proc. 6. Conf. Atmospheric Radiation*, Williamsburg (1986).
29. S. A. Clough and F. X. Kneizys, *Appl. Opt.* **18**, 2329 (1979).

APPENDIX 1

Approximation for the Voigt Function

For the convenience, we give here a short summary of Kielkopf's[8] and Whiting's methods[9] that were originally formulated for the Voigt profile $g_V(\tilde{\nu})$ as a function of wavenumber $\tilde{\nu}$ and line parameters $\tilde{\nu}_0$, γ_L, γ_D. With Eqs. (1, 3, 4) relating $g_V(\tilde{\nu})$ and $K(x, y)$, one can express Kielkopf's or Whiting's first approximation as

$$K(x,y) = K(0,y)\{[1-\eta(y)]G(x) + \eta(y)L(x)\}, \tag{A1}$$

where the Gaussian and Lorentzian contributions are given by

$$G(x) = \exp\left[-\sqrt{\ln 2}\left(\frac{x}{x_{1/2}}\right)^2\right] \quad \text{and} \quad L(x) = \left[1+\left(\frac{x}{x_{1/2}}\right)^2\right]^{-1}.$$

The inclusion of the correction proportional to $\eta(1-\eta)$ is straightforward.

The weight factor η is defined by Kielkopf as

$$\eta(y) = \frac{yx_{1/2}}{1+yx_{1/2}},$$

$$x_{1/2} = \tfrac{1}{2}y(1+\epsilon\ln 2 + \sqrt{(1-\epsilon\ln 2)^2 + 4\ln 2/y}) \quad \text{with} \quad \epsilon = 0.0990.$$

Similarly, Whiting has chosen the weight factor and Voigt half width

$$\eta(y) = \frac{y}{x_{1/2}}, \tag{A2}$$

$$x_{1/2} = \tfrac{1}{2}(c_1 y + \sqrt{c_2 y^2 + 4\ln 2}) \quad \text{with} \quad c_1 = c_2 = 1.$$

According to Olivero and Longbothum,[7] Whiting's empirical Voigt half width is accurate to within about 1%, and Kielkopf's line width is worse. In addition, Whiting's approximation can be significantly improved to about 0.02% accuracy by modifying the parameters to $c_1 = 1.0692$ and $c_2 = 0.86639$.

APPENDIX 2.

Printout of Humlicek's Algorithm: Vectorized Code

```
************************************************************************
      SUBROUTINE HUMLICEK (NX,X,Y, PRBFCT)
*                                                                      *
*     complex probability function for complex argument Z=X+iY         *
*     real part = voigt function K(x,y)                                *
*                                                                      *
*     source:   j. humlicek, JQSRT 27, 437, 1982                       *
*                                                                      *
*     parameters:                                                      *
*      NX      number of intervals (NX+1  grid points)            in *
*      X       array of grid points                               in *
*      Y       Voigt function parameter, ratio lorentz/doppler    in *
*      PRBFCT complex array of function values                    out *
*                                                                      *
************************************************************************

  REAL      X(0:NX), Y, S
  COMPLEX   T, U, PRBFCT(0:NX)
  COMPLEX   APPROX1, APPROX2, APPROX3, APPROX4
  APPROX1(T)   = (T * .5641896) / (.5 + (T * T))
  APPROX2(T,U) = (T * (1.410474 + U*.5641896))/ (.75 + (U *(3.+U)))
  APPROX3(T)   = ( 16.4955 + T * (20.20933 + T * (11.96482 +
 -                                T * (3.778987 + 0.5642236*T))))
 -              / ( 16.4955 + T * (38.82363 + T *
 -                (39.27121 + T * (21.69274 + T * (6.699398 + T)))))
```

```
      APPROX4(T,U) = (T * (36183.31 - U * (3321.99 - U * (1540.787 - U
     -         *(219.031 - U *(35.7668 - U *(1.320522 - U * .56419))))))
     -       / (32066.6 - U * (24322.8 - U * (9022.23 - U * (2186.18
     -         - U * (364.219 - U * (61.5704 - U * (1.84144 - U)))))))
C     ===================================================================
      IF (Y.GT.15.) THEN
C        ----------------------------------------------------------------
C        all points are in region I
         DO 100, I=0,NX
           T         = CMPLX(Y,-X(I))
 100       PRBFCT(I) = APPROX1(T)
C        ----------------------------------------------------------------
      ELSE IF (Y.LT.15. .AND. Y.GE.5.5) THEN
C        ----------------------------------------------------------------
C        points are in region I or region II
         DO 200, I=0,NX
           T  = CMPLX(Y,-X(I))
           S  = ABS(X(I)) + Y
           IF (S .GE. 15.) THEN
              PRBFCT(I) = APPROX1(T)
           ELSE
              U      = T * T
              PRBFCT(I) = APPROX2(T,U)
           END IF
 200       CONTINUE
C       ---------------------------------------------------------------
      ELSE IF (Y.LT.5.5 .AND. Y.GT.0.75) THEN
C        ----------------------------------------------------------------
         DO 300, I=0,NX
           T  = CMPLX(Y,-X(I))
           S  = ABS(X(I)) + Y
           IF (S .GE. 15.) THEN
              PRBFCT(I) = APPROX1(T)
           ELSE IF (S.LT.5.5) THEN
              PRBFCT(I) = APPROX3(T)
           ELSE
              U      = T * T
              PRBFCT(I) = APPROX2(T,U)
           END IF
 300       CONTINUE
C       ---------------------------------------------------------------
      ELSE
C        ----------------------------------------------------------------
         DO 400, I=0,NX
           T  = CMPLX(Y,-X(I))
           AX = ABS(X(I))
           S  = AX + Y
           IF (S .GE. 15.) THEN
C               region I
                PRBFCT(I)= APPROX1(T)
           ELSE IF (S.LT.15.0 .AND. S.GE.5.5) THEN
C               region II
                U = T * T
                PRBFCT(I)= APPROX2(T,U)
           ELSE IF (S.LT.5.5 .AND. Y.GE.(0.195*AX-0.176)) THEN
C              region III
               PRBFCT(I)= APPROX3(T)
           ELSE
C              region IV
               U  = T * T
               PRBFCT(I)= CEXP(U) - APPROX4(T,U)
           END IF
 400       CONTINUE
C        ----------------------------------------------------------------
      END IF
C     ===================================================================
      RETURN
      END
```