The Voigt profile function and the Planck function

M. Kuntz and M. Höpfner

Abstract: A new implementation of Humlicek's algorithm for approximating the Voigt profile function is applied in KOPRA and compared with several other implementations with respect to computational speed. On scalar computers this new implementation is considerably faster than other implementations by more than a factor 3.3 on the average. However, on the vector computer the acceleration between the new implementation and a fully vectorized implementation as described by Schreier [3] is only between 1.2 and 1.8, depending on the region of the x, y-space under consideration.
KOPRA is especially designed for the calculation of spectral microwindows. Therefore, the Planck function, the non-LTE source function and the non-LTE correction factor for absorption cross sections are calculated in such a way that a given relative accuracy over the microwindow range is reached.

1 Voigt profile

1.1 Introduction

The Voigt profile function is the convolution of a Gaussian and a Lorentzian function. It is expressed as

$$K(x,y) = \frac{y}{\pi}\int_{-\infty}^{+\infty}\frac{\exp(-t^2)}{(x-t)^2+y^2}dt, \tag{1}$$

where x is the distance from the line center in units of Doppler halfwidths and y is the ratio of the Doppler halfwidth to the Lorentzian halfwidth. The Voigt profile function is used in a wide range of contexts and there are—relevant to practical numerical algorithms—numerous ways it can be computed [1]. Many papers have been published describing routines for evaluating the Voigt profile function based upon various numerical expansions in different regions of the x,y space. Some of them—including Humlicek's algorithm [2]—take advantage of the fact that the Voigt profile function can be manipulated into an expression in terms of the complementary error function of complex argument, erfc(z), as

$$K(x,y) = \mathrm{Re}\left[\exp(z^2)\mathrm{erfc}(z)\right], \qquad z = y - ix. \tag{2}$$

This relation between the Voigt profile function $K(x,y)$ and erfc(z) allows for approximations of the error function erf(z) to be used also for K, since erfc(z) = $1-\mathrm{erf}(z)$. Taking advantage of this fact Humlicek's algorithm divides the x,y space into four regions, see Fig. 1. It then approximates the complex expression on the right hand side of Eq. (2), for each region, by appropriate rational polynomials. These polynomials are chosen to optimize the combination of accuracy and speed of computation. The Humlicek algorithm thus simultaneously calculates both the real and imaginary parts of Eq. (2). However, in optical spectroscopy only the real part is needed, especially if line coupling effects need not be taken into account. The purpose of this paper is to present a highly efficient algorithm for calculating the real part of the right hand side of Eq. (2).
In addition to its speed this new algorithm is still very accurate, since Humlicek's complex rational approximations are substituted by real ones. As test calculations revealed, the relative error compared with Humlicek's original implementation is less than $2\,10^{-6}$ throughout the x,y space.

1.2 Acceleration of Humlicek's algorithm

The first modification we propose refers to overall organization of the algorithm with respect to the nesting of DO-loops and IF-inquiries. Most computer programs that use the Voigt profile function apply it to a set of regularly spaced x values for a given y value rather than to selected x,y points. This is especially true if a complete atomic or rotational line is calculated. While the original Humlicek algorithm determines the region (and thus the related polynomials) for each x,y pair individually, we propose to take advantage of the regularity of the grid points in x. The regions then need only be determined at their end points, see Fig. 1. This results in an acceleration of Humlicek's algorithm without any loss of accuracy, only by reducing the number of IF-inquiries which are necessary to assign each x value to its appropriate region. The following sequence of operations, which defines a recursive algorithm, avoids repeated IF-inquiries and thus is well suited for evaluation of the Voigt profile function for a large set of x values. In order to apply this algorithm the x values have to be arranged in increasing order. For the sake of simplicity we shall further assume that x takes on only positive values. The extension to negative values can be easily performed.

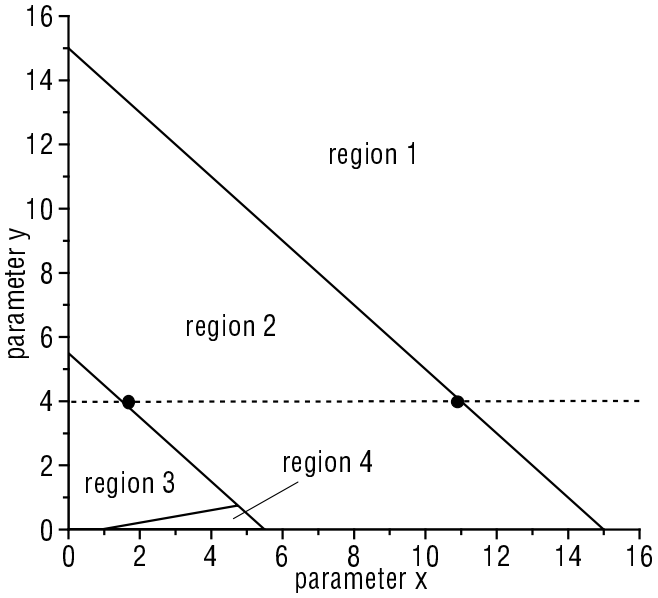

Figure 1: The four regions in the x,y space relevant for Humlicek's approximation of the Voigt profile function.

(a) If the left and right grid point of the current succession of x values belong to the same region (which implies, that all x values of the current succession also belong to this region), then store the indices of the extreme left and right grid points of this region so far.

(b) Else subdivide the current succession of x values into a left and right succession of approximately the same size and pop the right succession onto the stack. (In programming languages which do not support recursion the stack has to be provided by the user). Continue with the left succession at step 1.

(c) Push the next succession of x values from the stack and continue with step 1. If the stack is empty, finish the recursive algorithm.

As a consequence the number of IF-inquiries which are necessary to assign each x value to its appropriate region can be reduced from originally being equal to four times the number n of x values to less than $4\ln(n)/\ln(2)$. In addition the indices of the extreme left and right grid points of each region are known. This allows to substitute the former DO-loop containing conditional IF-branches (running over all x values) by four DO-loops (running over all x values lying within the same region) which do not contain IF-inquiries any longer. The new implementation can thus be better vectorized.
The second second modification we propose makes use of the fact that for most applications only the real part of Eq. (2) is needed. We thus substitute the complex rational polynomials Humlicek uses for approximation within each region by real ones. The latter can be calculated from the original approximations by separating them into a real and imaginary part, neglecting the imaginary one. In view of the intricacy of Humlicek's complex approximations all calculations were performed using a computer algebra system. A complete printout of the calculated real approximations for the Voigt profile function is given in the appendix. This leads to a considerable reduction of the number of floating point operations which have to be performed for each grid point in x individually. The reason is that those parts of the

calculations which concern only the fixed parameter y can be drawn forward and already be calculated before the DO-loops. The extent of the reduction of floating point operations within each region is indicated in Tab. 1.

Table 1: Achieved reduction of floating point operations in each of Humlicek's four regions. (In the case of Humlicek's complex rational approximation of the Voigt profile function all necessary complex floating operations have been expressed as real floating point operation equivalents.)

region	complex			real		
	$+/-$	$*/\div$	exp	$+/-$	$*/\div$	exp
1	7	16	0	3	4	0
2	17	24	0	7	8	0
3	37	38	0	9	10	0
4	56	62	3	29	30	2

1.3 Comparison of computing times

Most applications require the Voigt profile function for an array of x values at some selected y values, rather than for a limited set of x, y points. We thus follow a proposal of Schreier [3] and consider two sets of y values, $0 < y < 1$ and $1 < y < 10$, for comparison corresponding to Doppler and Lorentzian "dominance". For each of 50 y values in these intervals, we calculate the Voigt profile function for 1000 grid points in the interval $0 \leq x_i \leq x_{\max}$ with $x_{\max} = 10\, x_{\text{half}}$. For the halfwidth x_{half} of the Voigt profile function we use an empirical approximation $x_{\text{half}}(y) = \frac{1}{2}(y + \sqrt{y^2 + 4 \ln 2})$ proposed by Whiting [4]. In agreement with Schreier 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 $x_{\text{xmax}} = 20\, x_{\text{half}}$. Furthermore, computing times varied linearly with the number of grid points in the x, y-plane.

The tests were performed on a 90 MHz Pentium PC under DOS, using WATCOM's nonoptimizing WATFOR87 compiler (16 bit) and WATCOM's FORTRAN 77^{32} optimizing compiler (32 bit), respectively. The SUN SPARC20 workstation ran under Solaris3.1 operating system with a SunSoft FORTRAN 77 4.0 compiler while the CRAYJ90 (CMOS) used UNICOS 8.0.4.2 with a CF77 compiler. Highest optimization and vectorization level has been applied where possible.

Times required for 1000×50 evaluations are listed in Tab. 2. Besides Humlicek's original [2] and our new implementation two additional implementations have been included in our comparison of computational speed. The first one is part of GENLIN [5], a computer program for modeling the atmospheric radiative transfer line-by-line; the second one has been suggested by Schreier [3], who has drawn much attention to the vectorizability of his implementation. By far the fastest program on all scalar computers was our new implementation of Humlicek's algorithm, which on the average led to an acceleration factor of more than 3.3. On the SPARC workstation the acceleration even exceeded a factor 5.5. The other implementations were all nearly equally slow with slight advantages for the one or other depending on the computer and region under examination. However, on the vector computer Schreier's implementation was nearly as fast as our new implementation, with slight advantages for the latter in the region $1 < y < 10$ (factor 1.8). Both implementations were superior to the other implementations with respect to computational speed by at least a factor 10 due to their higher vectorizability.

Table 2: Computing times in seconds for various implementations of Humlicek's algorithm (1000×50 points and $0 < x < 10\,x_{\mathrm{half}}$).

Time (sec)	$0 < y < 1$			
	Humlicek	GENLIN	Schreier	new
PC 16 bit	2,447	2,070	1,771	1,224
PC 32 bit	2,410	1,538	1,494	0,517
SPARC20	0,359	0,204	0,216	0,058
CRAYJ90	0,220	0,387	0,046	0,025

Time (sec)	$1 < y < 10$			
	Humlicek	GENLIN	Schreier	new
PC 16bit	1,202	1,426	1,078	0,682
PC 32bit	0,647	0,547	0,548	0,203
SPARC20	0,108	0,095	0,089	0,015
CRAYJ90	0,138	0,184	0,016	0,013

In summary, both, the implementation of Schreier as well as our new implementation, represent good alternatives for a relatively fast calculation of the Voigt profile function on vector computers. On scalar computers our new implementation seems to be superior to the other implementations by more than a factor 3.3 on the average and a factor 5.5 for the SPARC20 workstation.

1.4 Conclusion on the Voigt profile implementation

We have proposed a new implementation of Humlicek's algorithm for approximating the Voigt profile function and have compared it with several other implementations with respect to accuracy and computational speed. While the accuracy of our new implementation can be regarded as very satisfactory, its computational speed is considerably higher than for the other implementations on scalar computers. On the vector computer our new implementation is only slightly faster than the implementation of Schreier (the acceleration factor varies between 1.2 and 1.8 depending on the region of the x,y space under examination) but it still exceeds the implementations of Humlicek and in GENLIN by at least a factor 10.

2 Planck function, non-LTE source function and non-LTE correction for absorption cross-sections

2.1 Introduction

For radiative transfer the Planck function B and in case of non-LTE the source function J^{NLTE} and the correction factor for absorption cross-sections α must be determined for each spectral grid point ν_i under consideration:

$$B(T_{kin,li}) = 2hc^2\nu_i^3 \left(\exp\left(\frac{hc\nu_i}{k_B T_{kin,l}}\right) - 1\right)^{-1}, \tag{3}$$

$$J^{NLTE}_{gkli} = 2hc^2\nu_i^3 \left(\frac{r_{gm_1l}}{r_{gm_2l}} \exp\left(-\frac{hc\nu_i}{k_B T_{kin,l}}\right) - 1\right)^{-1}, \tag{4}$$

$$\alpha_{gkli} = \frac{r_{gm_1l} - r_{gm_2l} \exp\left(-\frac{hc\nu_i}{k_B T_{kin,l}}\right)}{1 - \exp\left(-\frac{hc\nu_i}{k_B T_{kin,l}}\right)}. \quad (5)$$

(For a detailed description of all variables see the illustration of radiative transfer in Part X: 'Non-LTE and radiative transfer')
For the determination of analytical derivatives the derivatives of these three functions with respect to the kinetic temperature T_{kin} and the non-LTE/LTE population ratios r are calculated.
All these functions are, compared with the spectral line structure of the absorption cross sections, smoothly varying with wavenumber. Therefore, it is not necessary to recalculate them for each spectral grid point ν_i which would be very time consuming due to the exponentials.

2.2 Optimized implementation in KOPRA

The implementation in KOPRA is optimized particularly with regard to the calculation of spectral microwindows. Four steps with increasing complexity of wavenumber interpolation and microwindow subdivision are tested:

(a) *Constant value:* the function is calculated exactly at the end points and one point in the middle of the microwindow. If the maximum difference is less than a certain relative threshold (10^{-5}) the function value of the middle point is used for all grid points.

(b) *Linear interpolation:* If the previous criterion is not fulfilled it is checked if the mean of the function values at the microwindow boarders differ less than the relative threshold from the function value in the middle of the microwindow. Then, linear interpolation to all other grid points is performed.

(c) *Quadratic interpolation:* In case 1. and 2. are not valid new exact function values in the middle of the two previous intervals are determined and with these it is tested if a quadratic interpolation (using the end points and the middle of the microwindow) is sufficient.

(d) *Quadratic interpolation in sub-intervals:* If even 3. is not sufficient the microwindow is iteratively subdivided into smaller intervals until the quadratic interpolation reaches the relative error limit.

This scheme is used in the following KOPRA routines:

- `planckn@radtra_m`: Planck function and derivative with respect to kinetic temperature
- `sourcen@radtra_m`: non-LTE source function and derivative with respect to kinetic temperature
- `alphan@radtra_m`: non-LTE correction factor for cross-sections and derivative with respect to kinetic temperature
- `dsourcedTvibn@radtra_m`: derivative of the non-LTE source function with respect to the non-LTE/LTE population ratios
- `dalphadTvibn@radtra_m`: derivative of the non-LTE source function with respect to the non-LTE/LTE population ratios

Appendix A

Printout of used polynominals

Printout of the rational polynominals used for the approximation of the Voigt profile function. **region 1:** $|x| + y > 15$

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

$$K(x,y) = \frac{a_1 + b_1 x^2}{a_2 + b_2 x^2 + x^4}$$

region 2: $5.5 < |x| + y < 15$

$$\begin{aligned} a_3 &= 1.05786y + 4.65456y^3 + 3.10304y^5 + 0.56419y^7 \\ b_3 &= 2.962y + 0.56419y^3 + 1.69257y^5 \\ c_3 &= 1.69257y - 2.53885y^3 \\ d_3 &= 0.56419y \\ a_4 &= 0.5625 + 4.5y^2 + 10.5y^4 + 6y^6 + y^8 \\ b_4 &= -4.5 + 9y^2 + 6y^4 + 4y^6 \\ c_4 &= 10.5 - 6y^2 + 6y^4 \\ d_4 &= -6 + 4y^2 \end{aligned}$$

$$K(x,y) = \frac{a_3 + b_3 x^2 + c_3 x^4 + d_3 x^6}{a_4 + b_4 x^2 + c_4 x^4 + d_4 x^6 + x^8}$$

region 3: $|x| + y < 5.5$ and $y > 0.195|x| - 0.176$

$$
\begin{aligned}
a_5 &= 272.102 + 973.778y + 1629.76y^2 + 1678.33y^3 + 1174.8y^4 \\
&\quad +581.746y^5 + 204.501y^6 + 49.5213y^7 + 7.55895y^8 + 0.564224y^9 \\
b_5 &= -60.5644 - 2.34403y + 220.843y^2 + 336.364y^3 + 247.198y^4 \\
&\quad +100.705y^5 + 22.6778y^6 + 2.25689y^7 \\
c_5 &= 4.58029 + 18.546y + 42.5683y^2 + 52.8454y^3 + 22.6798y^4 \\
&\quad +3.38534y^5 \\
d_5 &= -0.128922 + 1.66203y + 7.56186y^2 + 2.25689y^3 \\
e_5 &= 0.000971457 + 0.564224y \\
a_6 &= 272.102 + 1280.83y + 2802.87y^2 + 3764.97y^3 + 3447.63y^4 \\
&\quad +2256.98y^5 + 1074.41y^6 + 369.199y^7 + 88.2674y^8 + 13.3988y^9 \\
&\quad +y^{10} \\
b_6 &= 211.678 + 902.306y + 1758.34y^2 + 2037.31y^3 + 1549.68y^4 \\
&\quad +793.427y^5 + 266.299y^6 + 53.5952y^7 + 5y^8 \\
c_6 &= 78.866 + 308.186y + 497.302y^2 + 479.258y^3 + 269.292y^4 \\
&\quad +80.3928y^5 + 10y^6 \\
d_6 &= 22.0353 + 55.0293y + 92.7568y^2 + 53.5952y^3 + 10y^4 \\
e_6 &= 1.49645 + 13.3988y + 5y^2
\end{aligned}
$$

$$
K(x,y) = \frac{a_5 + b_5x^2 + c_5x^4 + d_5x^6 + e_5x^8}{a_6 + b_6x^2 + c_6x^4 + d_6x^6 + e_6x^8 + x^{10}}
$$

region 4: $|x| + y < 5.5$ and $y < 0.195|x| - 0.176$

$$
\begin{aligned}
a_7 &= 1.16028e9y - 9.86604e8y^3 + 4.56662e8y^5 - 1.53575e8y^7 + 4.08168e7y^9 \\
&\quad -9.69463e6y^{11} + 1.6841e6y^{13} - 320772y^{15} + 40649.2y^{17} - 5860.68y^{19} \\
&\quad +571.687y^{21} - 72.9359y^{23} + 2.35944y^{25} - 0.56419y^{27} \\
b_7 &= -5.60505e8y - 9.85386e8y^3 + 8.06985e8y^5 - 2.91876e8y^7 + 8.64829e7y^9 \\
&\quad -7.72359e6y^{11} + 3.59915e6y^{13} - 234417y^{15} + 45251.3y^{17} - 2269.19y^{19} \\
&\quad -234.143y^{21} + 23.0312y^{23} - 7.33447y^{25} \\
c_7 &= -6.51523e8y + 2.47157e8y^3 + 2.94262e8y^5 - 2.04467e8y^7 + 2.29302e7y^9 \\
&\quad -2.3818e7y^{11} + 576054y^{13} + 98079.1y^{15} - 25338.3y^{17} + 1097.77y^{19} \\
&\quad +97.6203y^{21} - 44.0068y^{23} \\
d_7 &= -2.63894e8y + 2.70167e8y^3 - 9.96224e7y^5 - 4.15013e7y^7 + 3.83112e7y^9 \\
&\quad +2.2404e6y^{11} - 303569y^{13} - 66431.2y^{15} + 8381.97y^{17} + 228.563y^{19} \\
&\quad -161.358y^{21} \\
e_7 &= -6.31771e7y + 1.40677e8y^3 + 5.56965e6y^5 + 2.46201e7y^7 + 468142y^9 \\
&\quad -1.003e6y^{11} - 66212.1y^{13} + 23507.6y^{15} + 296.38y^{17} - 403.396y^{19} \\
f_7 &= -1.69846e7y + 4.07382e6y^3 - 3.32896e7y^5 - 1.93114e6y^7 - 934717y^9 \\
&\quad +8820.94y^{11} + 37544.8y^{13} + 125.591y^{15} - 726.113y^{17} \\
g_7 &= -1.23165e6y + 7.52883e6y^3 - 900010y^5 - 186682y^7 + 79902.5y^9 \\
&\quad +37371.9y^{11} - 260.198y^{13} - 968.15y^{15} \\
h_7 &= -610622y + 86407.6y^3 + 153468y^5 + 72520.9y^7 + 23137.1y^9 \\
&\quad -571.645y^{11} - 968.15y^{13} \\
o_7 &= -23586.5y + 49883.8y^3 + 26538.5y^5 + 8073.15y^7 - 575.164y^9 \\
&\quad -726.113y^{11} \\
p_7 &= -8009.1y + 2198.86y^3 + 953.655y^5 - 352.467y^7 - 403.396y^9 \\
q_7 &= -622.056y - 271.202y^3 - 134.792y^5 - 161.358y^7 \\
r_7 &= -77.0535y - 29.7896y^3 - 44.0068y^5 \\
s_7 &= -2.92264y - 7.33447y^3 \\
t_7 &= -0.56419y \\
a_8 &= 1.02827e9 - 1.5599e9y^2 + 1.17022e9y^4 - 5.79099e8y^6 + 2.11107e8y^8 \\
&\quad -6.11148e7y^{10} + 1.44647e7y^{12} - 2.85721e6y^{14} + 483737y^{16} - 70946.1y^{18} \\
&\quad +9504.65y^{20} - 955.194y^{22} + 126.532y^{24} - 3.68288y^{26} + y^{28} \\
b_8 &= 1.5599e9 - 2.28855e9y^2 + 1.66421e9y^4 - 7.53828e8y^6 + 2.89676e8y^8 \\
&\quad -7.01358e7y^{10} + 1.39465e7y^{12} - 2.84954e6y^{14} + 498334y^{16} - 55600y^{18} \\
&\quad +3058.26y^{20} + 533.254y^{22} - 40.5117y^{24} + 14y^{26} \\
c_8 &= 1.17022e9 - 1.66421e9y^2 + 1.06002e9y^4 - 6.60078e8y^6 + 6.33496e7y^8 \\
&\quad -4.60396e7y^{10} + 1.4841e7y^{12} - 1.06352e6y^{14} - 217801y^{16} + 48153.3y^{18} \\
&\quad -1500.17y^{20} - 198.876y^{22} + 91y^{24} \\
d_8 &= 5.79099e8 - 7.53828e8y^2 + 6.60078e8y^4 + 5.40367e7y^6 + 1.99846e8y^8 \\
&\quad -6.87656e6y^{10} - 6.89002e6y^{12} + 280428y^{14} + 161461y^{16} - 16493.7y^{18} \\
&\quad -567.164y^{20} + 364y^{22} \\
e_8 &= 2.11107e8 - 2.89676e8y^2 + 6.33496e7y^4 - 1.99846e8y^6 - 5.01017e7y^8 \\
&\quad -5.25722e6y^{10} + 1.9547e6y^{12} + 240373y^{14} - 55582y^{16} - 1012.79y^{18} \\
&\quad +1001y^{20}
\end{aligned}
$$

$$
\begin{aligned}
f_8 &= 6.11148e7 - 7.01358y^2 + 4.60396e7y^4 - 6.87656e6y^6 + 5.25722e6y^8 \\
&\quad +3.04316e6y^{10} + 123052y^{12} - 106663y^{14} - 1093.82y^{16} + 2002y^{18} \\
g_8 &= 1.44647e7 - 1.39465e7y^2 + 1.4841e7y^4 + 6.89002e6y^6 + 1.9547e6y^8 \\
&\quad -123052y^{10} - 131337y^{12} - 486.14y^{14} + 3003y^{16} \\
h_8 &= 2.85721e6 - 2.84954e6y^2 + 1.06352e6y^4 + 280428y^6 - 240373y^8 \\
&\quad -106663y^{10} + 486.14y^{12} + 3432y^{14} \\
o_8 &= 483737 - 498334y^2 - 217801y^4 - 161461y^6 - 55582y^8 \\
&\quad +1093.82y^{10} + 3003y^{12} \\
p_8 &= 70946.1 - 55600y^2 - 48153.3y^4 - 16493.7y^6 + 1012.79y^8 \\
&\quad +2002y^{10} \\
q_8 &= 9504.65 - 3058.26y^2 - 1500.17y^4 + 567.164y^6 + 1001y^8 \\
r_8 &= 955.194 + 533.254y^2 + 198.876y^4 + 364y^6 \\
s_8 &= 126.532 + 40.5117y^2 + 91y^4 \\
t_8 &= 3.68288 + 14y^2
\end{aligned}
$$

$$
\begin{aligned}
K(x,y) &= e^{y^2-x^2}\cos(2xy) - \\
&\quad \frac{a_7 + b_7x^2 + c_7x^4 + d_7x^6 + e_7x^8 + f_7x^{10} + g_7x^{12} + h_7x^{14} + \cdots}{a_8 + b_8x^2 + c_8x^4 + d_8x^6 + e_8x^8 + f_8x^{10} + g_8x^{12} + h_8x^{14} + \cdots} \\
&\quad \cdots \frac{+ o_7x^{16} + p_7x^{18} + q_7x^{20} + r_7x^{22} + s_7x^{24} + t_7x^{26}}{+ o_8x^{16} + p_8x^{18} + q_8x^{20} + r_8x^{22} + s_8x^{24} + t_8x^{26} + x^{28}}
\end{aligned}
$$

Bibliography

[1] W. J. Thompson, "Numerous neat algorithms for the Voigt profile function," *Computers in Physics* **7**(6), pp. 627–631, 1993.

[2] J. Humlicek, "Optimized computation of the Voigt and complex probability functions," *J. Quant. Spectrosc. Radiat. Transfer* **27**(4), pp. 437–444, 1982.

[3] F. Schreier, "The Voigt and complex error function: A comparison of computational methods," *J. Quant. Spectrosc. Radiat. Transfer* **48**(5/6), pp. 743–762, 1992.

[4] E. E. Whiting, "An empirical approximation to the Voigt profile," *J. Quant. Spectrosc. Radiat. Transfer* **8**, pp. 1379–1384, 1968.

[5] D. P. Edwards, "Atmospheric transmittance and radiance calculations using line–by–line computer models," in *Modelling of the Atmosphere*, L. S. Rothman, ed., *SPIE proceedings* **928**, pp. 94–116, 1988.