AIP | Conference Proceedings

Py4CAtS – Python tools for line-by-line modelling of infrared atmospheric radiative transfer

Franz Schreier and Sebastián Gimeno García

Citation: AIP Conf. Proc. **1531**, 123 (2013); doi: 10.1063/1.4804723

View online: http://dx.doi.org/10.1063/1.4804723

View Table of Contents: http://proceedings.aip.org/dbt/dbt.jsp?KEY=APCPCS&Volume=1531&Issue=1

Published by the American Institute of Physics.

Py4CAtS – Python Tools for Line-by-Line Modelling of Infrared Atmospheric Radiative Transfer

Franz Schreier and Sebastián Gimeno García

DLR — German Aerospace Center, Remote Sensing Technology Institute,
Oberpfaffenhofen, 82234 Wessling, Germany

Abstract. Py4CAtS — Python scripts for Computational ATmospheric Spectroscopy is a Python re-implementation of the Fortran infrared radiative transfer code GARLIC, where compute-intensive code sections utilize the Numeric/Scientific Python modules for highly optimized array-processing. The individual steps of an infrared or microwave radiative transfer computation are implemented in separate scripts to extract lines of relevant molecules in the spectral range of interest, to compute line-by-line cross sections for given pressure(s) and temperature(s), to combine cross sections to absorption coefficients and optical depths, and to integrate along the line-of-sight to transmission and radiance/intensity. The basic design of the package, numerical and computational aspects relevant for optimization, and a sketch of the typical workflow are presented.

Keywords: Line-by-line, Infrared, Voigt function, Multigrid.

INTRODUCTION

Radiative transfer modeling is an important aspect of atmospheric physics and chemistry, both for Earth and planetary atmospheres. In particular, for the analysis of atmospheric remote sensing data as well as for theoretical investigations such as retrieval assessments a flexible, yet efficient and reliable radiative transfer code is mandatory. In view of the increasing number of high resolution instruments and thanks to the increasing computational power, line-by-line modeling of high resolution infrared and microwave spectra is widely used today. Although highly optimized codes written in compiled languages such as Fortran or C/C++ are indispensable for operational processing, radiative transfer tools developed in script/interpreter languages such as Python, IDL/GDL, or MatLab/Octave/SciLab are an interesting alternative. Despite the reduced execution speed, script based tools are attractive because they allow for "rapid prototyping", can be executed on a large variety of platforms, and provide easy access to intermediate quantities, hence facilitating visualization and better understanding of the physics.

THE BASICS

For a pure gaseous atmosphere without scattering the transmission $\mathcal{T} = \exp(-\tau(\nu))$ describes attenuation due to absorption according to Beer's law, with the optical depth τ defined as

$$\tau(\nu, s) = \int_0^s \mathrm{d}s' \sum_m n_m(s')\, k_m\left(\nu, p(s'), T(s')\right), \tag{1}$$

where n_m and k_m are the number density and the absorption cross section of molecule m, respectively. In infrared line–by–line models, k_m is obtained by summing over the contributions from many lines,

$$k_m(\nu, p, T) = \sum_l S_l^m(T)\, g\left(\nu; \hat{\nu}_l^m, \gamma_l^m(p, T)\right), \tag{2}$$

where each individual line (at position $\hat{\nu}_l^m$) is described by the product of the temperature–dependent line strength S_l^m and a normalized line shape function g describing the broadening mechanism(s). The combined effect of pressure broadening (corresponding to a Lorentzian line shape) and Doppler broadening (Gaussian line shape) is represented by a Voigt line profile with half width γ_{V} that depends on the half width of the Lorentzian and Gaussian line shapes: γ_{L} and γ_{G}.

Radiation Processes in the Atmosphere and Ocean (IRS2012)
AIP Conf. Proc. 1531, 123-126 (2013); doi: 10.1063/1.4804723

Py4CAtS — IMPLEMENTATION AND USAGE

Py4CAtS is a Python re-implementation of a large part of the Fortran90/2003 Generic Atmospheric Radiation Line-by-line Infrared Code — GARLIC (*a.k.a.* MIRART, [1]) and makes extensive use of Numeric and Scientific Python modules (numpy, scipy) for compute intensive code sections. Individual steps of an infrared or microwave radiative transfer are implemented in separate scripts (Fig. 1), e.g.,

extract	(select) lines of relevant molecules in the spectral range of interest,
lbl2xs	line-by-line cross sections for given pressure(s) and temperature(s),
xs2ac	multiply cross sections with densities and sum over all molecules,
ac2od	integrate absorption coefficients along line-of-sight to optical depth,
od2ri	integrate optical depth along line-of-sight to radiance (Schwarzschild equation).

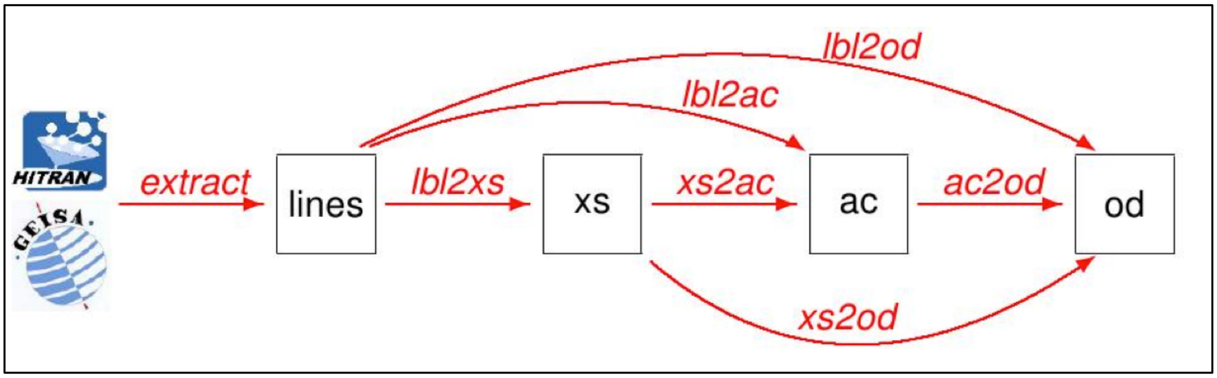
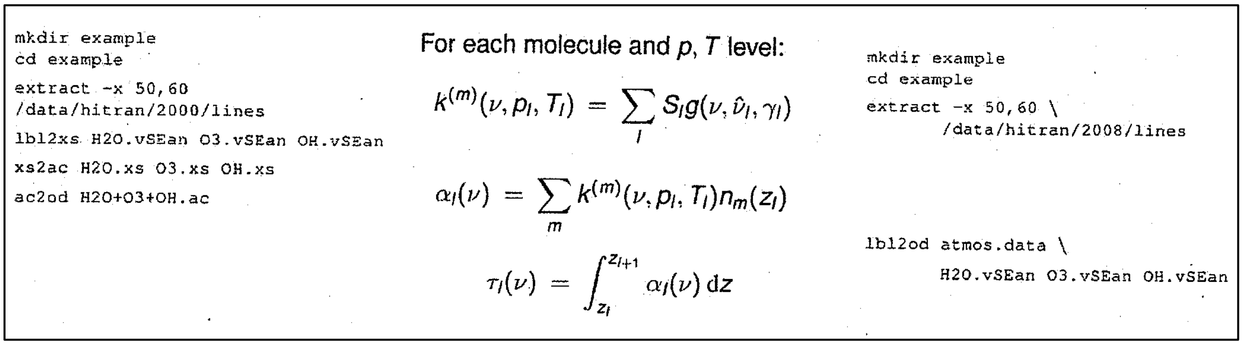


FIGURE 1. Overview of the “main” scripts available in the Py4CAtS package. In addition the package contains “auxiliary” modules, e.g., for plotting.

A typical workflow is shown in Fig. 2 (left). In general all scripts require one (or several) input files as mandatory argument(s), and support a variety of further options where appropriate default values are used if unspecified (e.g., standard pressure and temperature). Output is typically written to stdout if not redirected with the -o option. Basic usage and explanation of all options are available via the -h option.



```
mkdir example
cd example
extract -x 50,60
/data/hitran/2000/lines
lbl2xs H2O.vSEan O3.vSEan OH.vSEan
xs2ac H2O.xs O3.xs OH.xs
ac2od H2O+O3+OH.ac
```

For each molecule and p, T level:

$$k^{(m)}(\nu, p_l, T_l) = \sum_l S_l g(\nu, \hat{\nu}_l, \gamma_l)$$

$$\alpha_l(\nu) = \sum_m k^{(m)}(\nu, p_l, T_l) n_m(z_l)$$

$$\tau_l(\nu) = \int_{z_l}^{z_{l+1}} \alpha_l(\nu)\,\mathrm{d}z$$

```
mkdir example
cd example
extract -x 50,60 \
        /data/hitran/2008/lines

lbl2od atmos.data \
       H2O.vSEan O3.vSEan OH.vSEan
```

FIGURE 2. Typical workflow. Left: Usage of the individual scripts (Note: specification of the atmospheric data file for xs2ac is not indicated.) Right: Usage of the lbl2od script.

Clearly usage of a series of scripts results in extensive file I/O (note that even for small wavenumber (wavelength) intervals the number of spectral grid points can become extremely large). Furthermore the scripts have to carefully test the validity and consistency of the input data, e.g., the **xs2ac** script has to check that all cross sections are covering the same spectral interval and are defined on the same pressure/temperature levels. Accordingly some “shortcut” scripts are available, notably **lbl2od** allowing to compute layer optical depths (important for radiative transfer solvers such as DISORT) directly from the line parameter extracts, see Fig. 2 (right).

NUMERICS

The Voigt Function

The convolution of Lorentz and Gauss profiles leads to the Voigt profile $g = g_\mathrm{L} \otimes g_\mathrm{G}$ and to the Voigt function defined as

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

where x is essentially the distance to the line center and y the ratio of Lorentz to Gauss half width. There is no closed analytical solution for this integral, and in the past century some dozen algorithms have been developed. In most codes the x, y plane is divided in several subregions, and appropriate approximations are used, e.g. series expansion for small arguments, or asymptotic expansions for large x. In recent years rational approximations of the closely related complex error function

$$w(z) = K(x,y) + \mathrm{i}L(x,y) = \frac{\mathrm{i}}{\pi}\int_{-\infty}^{\infty} \frac{e^{-t^2}}{z-t}\,\mathrm{d}t \quad \text{with } z = x + \mathrm{i}y \tag{4}$$

have gained increasing attention. The widely used Humliček code [2] exploits four different x, y regions, and accordingly an efficient implementation in (Numeric) Python is difficult. On the other hand, rational approximations without any division of the x, y plane are problematic: the Hui et al. [3] code exhibits large errors for small y $\ll$1, and the Weideman [4] code is relatively time consuming.

However, for a reasonable fine sampling of the wavenumber grid (typically $\delta\nu \approx \gamma_\mathrm{V}/4$) there are only relatively few grid points in the three "central regions" of the Humliček code, i.e., optimization of the approximation for small x is not critical. Accordingly, Py4CAtS utilizes a combination of the Weideman approximation for small x, y with Humliček's approximation for large arguments [5]:

$$w(z) = \begin{cases} \dfrac{\mathrm{i}z/\sqrt{\pi}}{z^2-\frac{1}{2}} & \text{for } |x|+y \geq 15 \\[2ex] \dfrac{\pi^{-1/2}}{L-\mathrm{i}z} + \dfrac{2}{(L-\mathrm{i}z)^2}\displaystyle\sum_{n=0}^{N-1} a_{n+1} Z^n & \text{else} \quad \begin{array}{l} Z=(L+\mathrm{i}z)/(L-\mathrm{i}z) \\ L=2^{-1/4}N^{1/2} \end{array} \end{cases} \tag{5}$$

Benchmark tests using cross sections evaluation in the microwave regime for atmospheric pressures covering troposphere to mesosphere indicate that this combination is superior to the original Hui, Humlicek, and Weideman algorithms, both for Fortran and Python (Fig. 3) implementations.

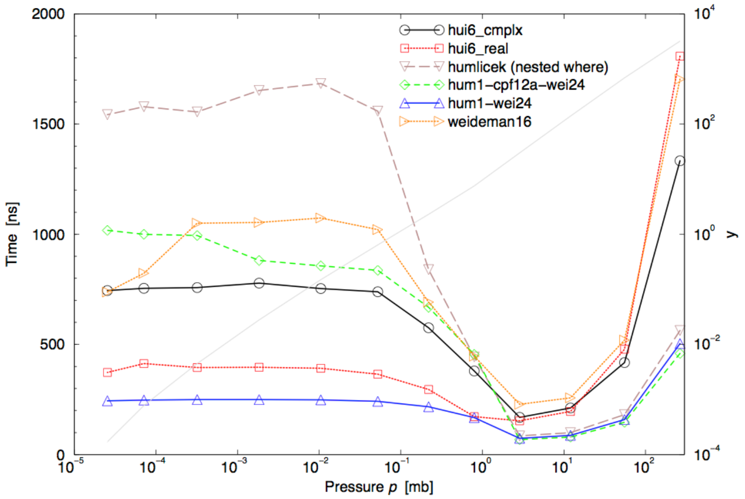


FIGURE 3. CPU time (per evaluation of a single function value) of various Voigt function implementations for computation of HNO_3 cross sections in the microwave regime; (hui6 indicates the Hui et al. approximation with a 6th order nominator polynomial, cpf12 the Humlicek (1979) approximation, and hum1-wei24 the combination (5) with Weideman's approximation for N = 24.)

Multigrid

Even for small wavenumber intervals as typically considered in the microwave, or for operational processing of thermal infrared spectra (microwindows) the number of spectral data points can become extremely large (up to hundred thousands or a million points) especially for the low pressures in the upper atmosphere. For these small pressures (and corresponding small y) the line center is narrow, and the majority of evaluations have to be performed in the line wings, where the Voigt function is essentially a Lorentzian, i.e. numerous time consuming divisions have to be evaluated. To alleviate this problem, lbl codes typically exploit some kind of two- or multigrid scheme.

Py4CAtS uses two- or three-grid approximations, where the line profile (not necessarily Voigt or symmetric) is evaluated on a fine grid only near the line center. In the two-grid scheme, the profile is computed on a coarse grid covering the entire wavenumber interval under consideration. (In order not to model the line center twice, the "fine grid profile" is actually the line profile with an interpolated coarse grid contribution subtracted). The extension to a three-grid scheme is straightforward.

In the original implementations [6] Lagrange interpolation has been used: Two-point interpolation is fast and robust, however in the line wings this approach leads to an overestimate of the interpolated profile. On the other hand, three- or four-point interpolation can lead to wiggles, and in rare cases this can even give rise to some negative cross section values. Moreover, line wing truncation becomes difficult in the case of higher order Lagrange interpolation. To overcome these problems, a cubic Hermite interpolation scheme was developed recently.

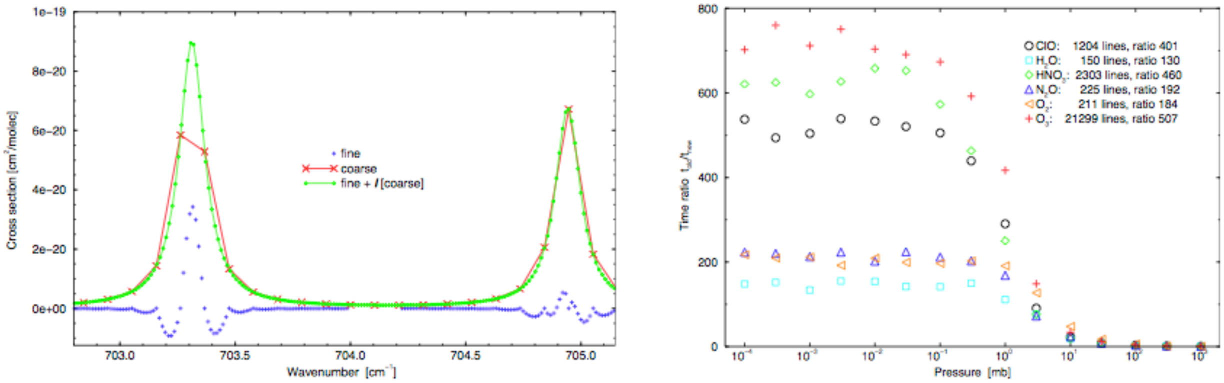


FIGURE 4. a) Schematics of the two-grid scheme. b) Relative speed-up (CPU time) for evaluation of microwave cross sections by three-grid scheme relative to "brute force".

SUMMARY

Py4CAtS is "work in progress"! Future developments include utilization of Fortran wrappers to interface directly to Python, or further optimization of the cross section computation.

ACKNOWLEDGMENTS

The development of the tools, nb. **lbl2od**, has been supported by ESA study ESAS-Light, see http://esaslight.libradtran.org/. Furthermore we would like to thank the Helmholtz Research Alliance "Planetary Evolution and Life" and the World Data Center for Remote Sensing of the Atmosphere.

REFERENCES

1. F. Schreier and B. Schimpf, in W. L. Smith and Y. Timofeyev, editors, *IRS 2000: Current Problems in Atmospheric Radiation*, pages 381–384, A. Deepak Publishing (2001).
2. J. Humliček, *J. Quant. Spect. & Rad. Transfer* **27**, 437–444, doi:10.1016/0022-4073(82)90078-4 (1982).
3. A. K. Hui, B. H. Armstrong and A. A. Wray, *J. Quant. Spect. & Rad. Trans.* **19**, 509–516, doi: 10.1016/0022-4073(78)90019-5 (1978).
4. J. A. C. Weideman, *SIAM J. Num. Anal.* **31**, 1497–1518, doi: 10.1137/0731077 (1994).
5. F. Schreier, *J. Quant. Spect. & Rad. Trans.* **112**(6), 1010–1025, doi:10.1016/j.jqsrt.2010.12.010 (2011).
6. F. Schreier, *Comp. Phys. Comm.* **174**, 783–802, doi: 10.1016/j.cpc.2005.12.015 (2006).