The Journal of Open Source Software

pythonradex: a fast Python re-implementation of RADEX with extended functionality

Gianni Cataldi [1¶]

1 National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka, Tokyo 181-8588, Japan ¶ Corresponding author

DOI: 10.21105/joss.09233

Software

- Review
- Repository
- Archive

Editor: Warrick Ball

Reviewers:

- @gijsvermarien
- @andresmegias

Submitted: 04 March 2025
Published: 24 February 2026

Summary

A common task in astronomical research is to estimate the physical parameters (temperature, mass, density etc.) of a gas by using observed line emission. This often requires a calculation of how the radiation propagates via emission and absorption ("radiative transfer"). In radio and infrared astronomy, the Fortran code RADEX (van der Tak et al., 2007) is a popular tool to solve the non-LTE radiative transfer of a uniform medium in a simplified geometry. I present a Python re-implementation of RADEX: pythonradex. Written in Python, it provides an easy and intuitive user interface, improved performance as well as additional functionality not included in RADEX (continuum effects and overlapping lines).

Statement of need

Modern astronomical facilities such as the Atacama Large Millimeter/submillimeter Array (ALMA) or the James Webb Space Telescope (JWST) are providing a wealth of line emission data at radio and infrared wavelengths. These data are crucial to constrain the physical and chemical properties of various astrophysical environments.

To interpret such data, a radiative transfer calculation is typically used (see Rybicki & Lightman, 1985, for an introduction to radiative transfer). For a given set of input parameters describing the source (temperature, density, geometry, etc.), one calculates the amount of radiation reaching the telescope. The input parameters are then adjusted such that the predicted flux matches the observations.

If the medium is dense enough, local thermodynamic equilibrium (LTE) can be assumed. This considerably simplifies the radiative transfer calculation. A non-LTE calculation is considerably more complex and computationally expensive because the fractional population of the molecular energy levels needs to be solved for numerically (e.g. van der Tak et al., 2007). Typically, an iterative approach is used: from a first guess of the level populations, the radiation field is calculated. This radiation field is then used to solve for updated level populations. The iterations continue until convergence is reached.

Various codes are available to solve the radiative transfer. Codes solving the radiative transfer in 3D are used for detailed calculations of sources with well-known geometries. Examples include RADMC-3D (Dullemond et al., 2012) and LIME (Brinch & Hogerheijde, 2010). However, a full 3D calculation is often too computationally expensive if a large parameter space needs to be explored, in particular in non-LTE. 1D codes that quickly provide an approximate solution are a commonly used alternative. In this respect, the 1D non-LTE code RADEX (van der Tak et al., 2007) has gained considerable popularity: as of February 10, 2026, the paper presenting RADEX (van der Tak et al., 2007) has 1463 citations. The Fortran code RADEX solves the radiative transfer of a uniform medium using an escape probability formalism.

The Python programming language is now widely used in astronomy. Still, no Python version of RADEX is available, although some Python wrappers such as SpectralRadex (Holdship et al., 2023) or ndradex (Taniguchi, 2019), and even a Julia version (Jadex, Svoboda, 2022), exist. Furthermore, RADEX cannot take into account the effects of an internal continuum field (typically arising from dust that is mixed with the gas), nor cross-excitation effects arising when transitions overlap in frequency. The pythonradex code addresses these concerns.

Implementation

pythonradex is written in Python and implements the accelerated lambda iteration (ALI) scheme presented by Rybicki & Hummer (1992). Like RADEX, an escape probability equation is used to calculate the radiation field for a given level population. This allows solving the radiative transfer iteratively. To speed up the convergence, Ng acceleration (Ng, 1974) is employed. To improve performance, critical parts of the code are just-in-time compiled using Numba (Lam et al., 2015).

pythonradex supports four geometries: two static geometries (slab and sphere), and two large-velocity-gradient (LVG) geometries (again slab and sphere). In the LVG approximation, it is assumed that all regions of the source are Doppler shifted with respect to each other due to a velocity gradient. This means that all photons escape the source, unless absorbed locally (i.e. close to the emission location, e.g. Elitzur, 1992).

Currently, effects of internal continuum and overlapping lines can only be included for the static geometries. Another limitation is that only a single molecule can be considered at a time. Thus, solving the radiative transfer of overlapping lines of different molecules is not supported yet. Also, treating overlapping lines adds considerable computational cost because averages over line profiles need to be calculated.

Like RADEX, pythonradex needs a file in LAMDA-format as input to read the molecular data. Such files can for example be downloaded from the LAMDA (Schöier et al., 2005) or EMAA (*EMAA*, 2021) databases.

Benchmarking

pythonradex was benchmarked against RADEX for a number of example problems, generally with excellent agreement (see Figure 1 for an example). To test the treatment of overlapping lines, pythonradex was tested against the MOLPOP-CEP code (Asensio Ramos & Elitzur, 2018), again showing good agreement, as illustrated in Figure 2.

Performance advantage

Both pythonradex and RADEX are single-threaded. To compare their performance, we consider the calculation of a grid of models over a parameter space spanning 20 values in each of kinetic temperature, column density and H_2 density (i.e. a total of 8000 models). We consider a few different molecules: C (small number of levels and transitions), SO (large number of levels and transitions) as well as CO and HCO^+ (intermediate). On a laptop with i7-7700HQ cores running on Ubuntu 22.04, pythonradex computed the model grid faster than RADEX by factors of approximately 1.5 (C), 6 (SO), 7 (CO) and 3 (HCO^+). Running the same test on the Multi-wavelength Data Analysis System (MDAS) operated by the National Astronomical Observatory of Japan (Rocky Linux 8.9 with AMD EPYC 7543 CPUs) resulted in an even larger performance advantage: pythonradex calculated the grid faster by factors of 13 (C), 10 (SO), 13 (CO) and 12 (HCO^+)[1].

[1] Jadex (Svoboda, 2022) claims a performance advantage of a factor ~110 over RADEX.

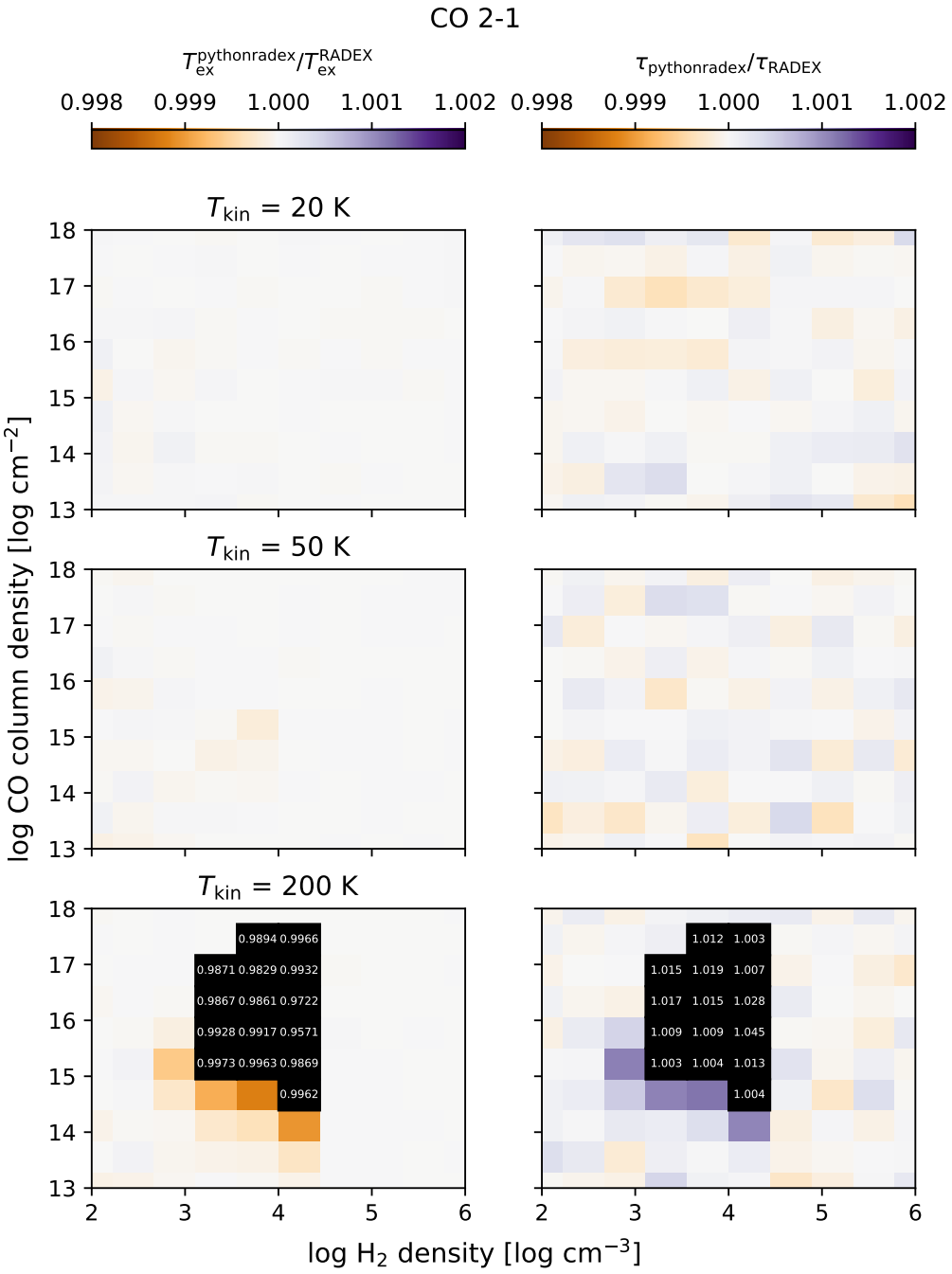

Figure 1: The ratio of CO 2-1 excitation temperature (left column) and optical depth (right column) computed with `pythonradex` and `RADEX` for a static sphere. Each panel shows a parameter space of H_2 density and column density for a fixed kinetic temperature. Values exceeding the colorbar range are shown in black with the corresponding value in white text.

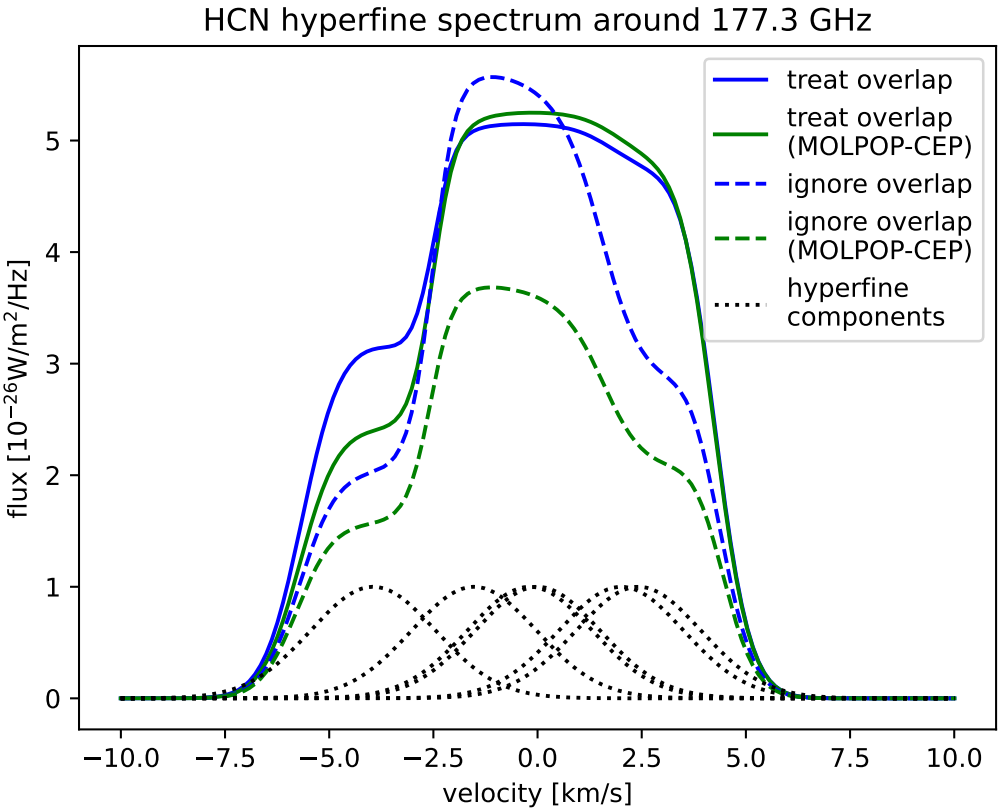

Figure 2: Spectrum of HCN around 177.3 GHz computed with `pythonradex` and `MOLPOP-CEP` for a static slab. Good agreement is found when treating line overlap. Interestingly, the spectra differ somewhat when ignoring overlap. The positions and widths of the individual hyperfine components are illustrated by the black dotted lines.

Additional differences between `RADEX` and `pythonradex`

Output flux

`RADEX` computes line fluxes based on a "background subtracted" intensity given by $(B_\nu(T_{\rm ex}) - I_{\rm bg})(1 - e^{-\tau})$, where B_ν is the Planck function, $T_{\rm ex}$ the excitation temperature, $I_{\rm bg}$ the external background and τ the (frequency-dependent) optical depth. This may or may not be the right quantity to be compared to observations. For example, it is not appropriate when considering data from interferometers like ALMA. `pythonradex` does not apply any observational correction, giving the user the flexibility to decide how to compare the computed fluxes to observations.

Flux for spherical geometry

To calculate line fluxes, `pythonradex` uses different formulae depending on the geometry (see the `pythonradex` documentation for more details). On the other hand, `RADEX` always uses the formula for a slab. Figure 3 illustrates the consequences. For a static sphere, by using the slab formula, the flux is overestimated by a factor 1.5[2] in the optically thin limit. In the optically thick case, only the surface of the static sphere is visible, so the different formulae give the same result. On the other hand, for the LVG sphere, the difference is always a factor 1.5 regardless of optical depth. This is a consequence of the LVG assumption that photons

[2]The factor 1.5 corresponds to the volume ratio of a "spherical slab" (i.e. a cylinder) to a sphere.

always escape unless absorbed locally. By computing the optically thin flux directly[3], it can be confirmed that the formulae used by `pythonradex` are correct.

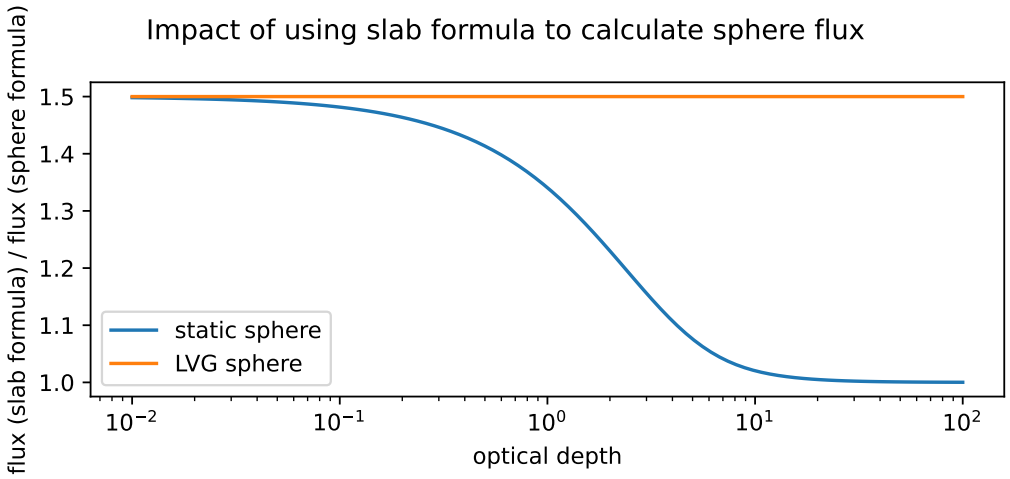

Figure 3: For the two spherical geometries, fluxes using the slab formula (as done by `RADEX`) and formulae appropriate for a sphere (as done by `pythonradex`) were calculated. The figure shows the ratio of the fluxes as function of optical depth.

Dependencies

`pythonradex` depends on the following packages:

- NumPy (Harris et al., 2020)
- SciPy (Virtanen et al., 2020)
- Numba (Lam et al., 2015)

Acknowledgements

I thank @andresmegias and @GijsVermarien for their thorough review which led to important improvements of the code and documentation. I would also like to thank Simon Bruderer for his helpful clarifications about the ALI method, Andrés Asensio Ramos for helpful discussions about the LVG approximation and the `MOLPOP-CEP` code, and @pscicluna for helpful comments on the code and paper draft. Performance testing was in part carried out on the Multi-wavelength Data Analysis System operated by the Astronomy Data Center (ADC), National Astronomical Observatory of Japan.

References

Asensio Ramos, A., & Elitzur, M. (2018). MOLPOP-CEP: An exact, fast code for multi-level systems. *Astronomy and Astrophysics*, *616*, A131. https://doi.org/10.1051/0004-6361/201731943

Brinch, C., & Hogerheijde, M. R. (2010). LIME - a flexible, non-LTE line excitation and radiation transfer method for millimeter and far-infrared wavelengths. *Astronomy and Astrophysics*, *523*, A25. https://doi.org/10.1051/0004-6361/201015333

[3] In units of W/m^2, $F_{\mathrm{thin}} = V_{\mathrm{sphere}} n_2 A_{21} \Delta E \frac{1}{4\pi d^2}$ with $V_{\mathrm{sphere}} = \frac{4}{3} R^3 \pi$ the volume of the sphere, n the number density, x_2 the fractional level population of the upper level, A_{21} the Einstein coefficient, ΔE the energy of the transition and d the distance.

Dullemond, C. P., Juhasz, A., Pohl, A., Sereshti, F., Shetty, R., Peters, T., Commercon, B., & Flock, M. (2012). *RADMC-3D: A multi-purpose radiative transfer tool*. Astrophysics Source Code Library, record ascl:1202.015. https://ui.adsabs.harvard.edu/abs/2012ascl.soft02015D

Elitzur, M. (1992). Basic background concepts. In *Astronomical masers* (pp. 4–46). Springer Netherlands. https://doi.org/10.1007/978-94-011-2394-5_2

EMAA. (2021). UGA, CNRS, CNRS-INSU, OSUG. https://doi.org/10.17178/EMAA

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Holdship, J., Vermariën, G., Keil, M., & James, T. (2023). SpectralRadex. In *GitHub repository*. GitHub. https://github.com/uclchem/SpectralRadex

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based Python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. https://doi.org/10.1145/2833157.2833162

Ng, K.-C. (1974). Hypernetted chain solutions for the classical one-component plasma up to Γ=7000. *Journal of Chemical Physics*, *61*(7), 2680–2689. https://doi.org/10.1063/1.1682399

Rybicki, G. B., & Hummer, D. G. (1992). An accelerated lambda iteration method for multilevel radiative transfer. II. Overlapping transitions with full continuum. *Astronomy and Astrophysics*, *262*, 209–215. https://ui.adsabs.harvard.edu/abs/1992A&A...262..209R

Rybicki, G. B., & Lightman, A. P. (1985). Fundamentals of radiative transfer. In *Radiative processes in astrophysics* (pp. 1–50). John Wiley & Sons, Ltd. https://doi.org/10.1002/9783527618170.ch1

Schöier, F. L., van der Tak, F. F. S., van Dishoeck, E. F., & Black, J. H. (2005). An atomic and molecular database for analysis of submillimetre line observations. *Astronomy and Astrophysics*, *432*(1), 369–379. https://doi.org/10.1051/0004-6361:20041729

Svoboda, B. (2022). Jadex. In *GitHub repository*. GitHub. https://github.com/autocorr/Jadex.jl

Taniguchi, A. (2019). ndRADEX. In *GitHub repository*. GitHub. https://github.com/astropenguin/ndradex

van der Tak, F. F. S., Black, J. H., Schöier, F. L., Jansen, D. J., & van Dishoeck, E. F. (2007). A computer program for fast non-LTE analysis of interstellar line spectra. With diagnostic plots to interpret observed line intensity ratios. *Astronomy and Astrophysics*, *468*(2), 627–635. https://doi.org/10.1051/0004-6361:20066820

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2