


# pythonradex: a python implementation of the non-LTE radiative transfer code RADEX with additional functionality

Gianni Cataldi <sup>1</sup>✉

<sup>1</sup> National Astronomical Observatory of Japan  ✉ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Warrick Ball](#) 

## Reviewers:

- [@psicluna](#)
- [@gjsvermarien](#)

Submitted: 04 March 2025

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## 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 implementation of RADEX: pythonradex. Written in pure python, it provides an easy and intuitive user interface as well as additional functionality not included in RADEX (continuum effects and overlapping lines).

## Statement of need

Modern 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) maybe 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 calculated explicitly.

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 28, 2025, the RADEX paper by van der Tak et al. (2007) has 1361 citations. The Fortran code RADEX solves the radiative transfer of a uniform medium using an escape probability formalism.

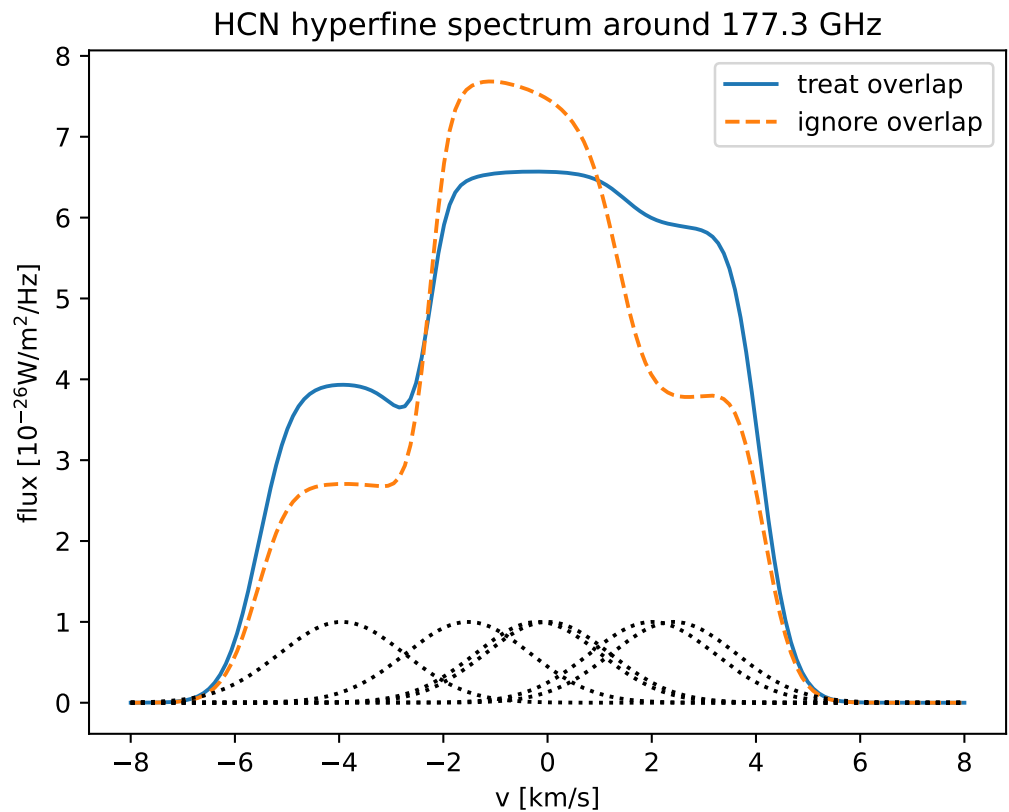
The python programming language is now very widely used in astronomy. Still, no python version of RADEX is available, although some python wrappers (for example SpectralRadex

(Holdship et al., 2023) or ndradex (Taniguchi, 2019)) and even a Julia version (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 pure 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. Critical parts of the code are just-in-time compiled using numba (Lam et al., 2015). pythonradex supports four geometries: static sphere, large velocity gradient (LVG) sphere, static slab and LVG slab. Effects of internal continuum and overlapping lines can be included for the static geometries.

Figure 1 illustrates the capability of pythonradex to solve the radiative transfer for overlapping lines. Note that treating overlapping lines adds considerable computational cost because averages over line profiles need to be calculated.



**Figure 1:** Spectrum of HCN around 177.3 GHz computed with pythonradex. The blue solid and orange dashed lines show the spectrum calculated with cross-excitation effects turned on and off, respectively. The positions and widths of the individual hyperfine transitions are illustrated by the black dotted lines.

## Benchmarking

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

## Performance

pythonradex is optimised for the use case of a parameter space exploration. On a laptop with four i7-7700HQ cores (eight virtual cores), running a grid of CO models was three times faster with pythonradex compared to RADEX. However, the CPU usage of pythonradex was much higher than RADEX during this test. Still, it demonstrates that pythonradex might be faster than RADEX depending on the setup.

## Additional differences between RADEX and pythonradex

### Output flux

RADEX computes line fluxes based on a “background subtracted” intensity given by  $(B_\nu(T_{\text{ex}}) - I_{\text{bg}})(1 - e^{-\tau_\nu})$ , where  $B_\nu$  is the Planck function,  $T_{\text{ex}}$  the excitation temperature,  $I_{\text{bg}}$  the external background and  $\tau_\nu$  the 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 the computed fluxes are compared to observations.

### Flux for spherical geometry

Regardless of the adopted geometry, RADEX always uses the flux formula for a slab geometry, resulting in inconsistencies. Consider the optically thin limit where the total flux (in  $[\text{W}/\text{m}^2]$ ) for a sphere is simply given by

$$F_{\text{thin}} = V_{\text{sphere}} n_2 A_{21} \Delta E \frac{1}{4\pi d^2} \quad (1)$$

with  $V_{\text{sphere}} = \frac{4}{3}R^3\pi$  the volume of the sphere,  $n$  the number density,  $x_2$  the fractional level population of the upper level,  $A_{21}$  the Einstein coefficient,  $\Delta E$  the energy of the transition and  $d$  the distance. pythonradex correctly reproduces this limiting case by using the formula by Osterbrock (1974), while RADEX overestimates the optically thin flux by a factor 1.5.

## 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 Simon Bruderer for his helpful clarifications about the ALI method, and Andrés Asensio Ramos for helpful discussions about the LVG geometry and the MOLPOP-CEP code.

## 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>
- 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>
- 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  $\Gamma=7000$ . *Journal of Chemical Physics*, 61(7), 2680–2689. <https://doi.org/10.1063/1.1682399>
- Osterbrock, D. E. (1974). *Astrophysics of gaseous nebulae*. W. H. Freeman.
- 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>
- 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>