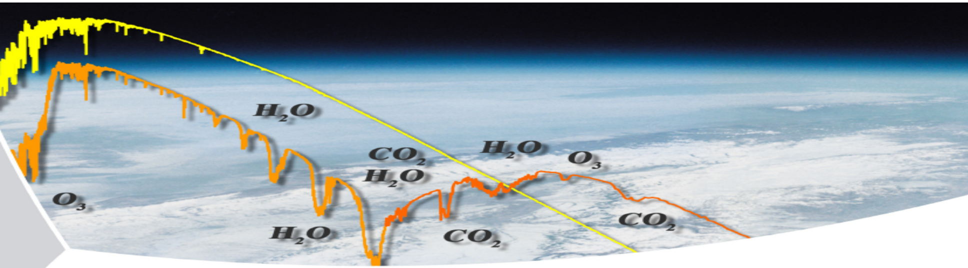


Py4CAtS
PYthon for Computational ATmospheric Spectroscopy

Franz Schreier
DLR — Remote Sensing Technology Institute
Oberpfaffenhofen, GERMANY

Outline

Why?¿? — Motivation & History

Why radiative transfer?

- Sensitivity studies: detectability of molecules, etc.
- Forward model for atmospheric inverse problems

Why line-by-line (lbl)?

- Modeling and analysis of high resolution spectra
- "Training set" and benchmark for fast models

Why a new lbl code?

- Early 1990s: not "happy" with FASCODE
- Maybe GenLn2 or RFM or ... ???
  ARTS not yet alive

$\Longrightarrow$ MIRART / SQuIRRl (Fortran 77) $\longrightarrow$ GARLIC

Why?¿? — Motivation & History

Why Python?

- “Discovered” mid 1990s
- Rapid prototyping
- “Computational steering”
  number crunching in Fortran (or . . .), rest in Python

Why a second lbl code?

- Most (?) lbl models kind of “black-box”
- Difficult to see intermediate quantities
- Wrappers (pyfort, f2py) not that easy

Outline

Infrared Radiative Transfer in the Atmosphere

- Radiative transfer equation

$$\frac{\mathrm{d}I(\nu,s)}{k(\nu,s)\;n(s)\;\mathrm{d}s} \;=\; -I(\nu,s) \;+\; B(\nu,s)$$

- Schwarzschild — Monochromatic Intensity / Radiance

$$I(\nu,s) \;=\; I(\nu,s_0)\;\mathrm{e}^{-\tau(\nu;s_0,s)} \;+\; \int_0^\tau \mathrm{d}\tau'\; B\big(\nu,\, T(\tau')\big)\; \mathrm{e}^{-\tau'}$$

- Beer: Transmission $\mathcal{T}$ and optical depth τ

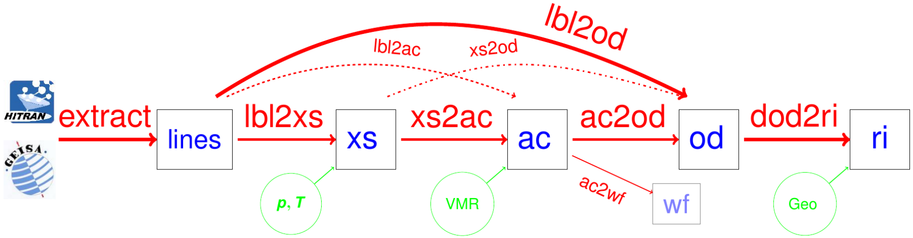
$$\mathcal{T}(\nu) \;=\; \mathrm{e}^{-\tau} \;=\; \exp\big(-k(\nu)\,n\,s\big)$$

. . . in an inhomogeneous atmosphere with some molecules . . .

$$\tau(\nu) = \int_{\text{path}} \mathrm{d}s \sum_m \sum_l S_l\big(T(s)\big)\; g_{\mathrm{L}}\big(\nu;\hat{\nu}_l,\gamma_l^{\mathrm{L}}(p(s),T(s)\big)\otimes g_{\mathrm{G}}\big(\nu;\hat{\nu}_l,\gamma_l^{\mathrm{G}}(T(s)\big)\,n_m(s)$$

Py4CAtS

- (Numeric and Scientific) *Python* version of Fortran 2008 "Generic Atmospheric Radiation Lbl Ir Code" GARLIC
- Series of scripts/functions for IR & μWave radiative transfer



- *Old:* scripts to execute from Unix/Linux (or Windows) shell
- *New:* functions accessible within (I)Python shell

Outline

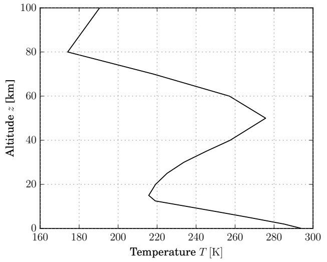
Atmospheric Data

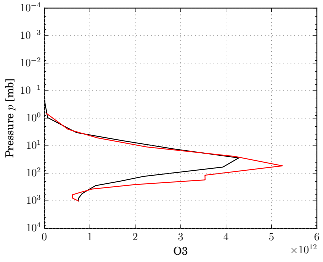
```
Python 2.7.3 (default, Apr 14 2012, 08:58:41) [GCC]
IPython 2.0.0 -- An enhanced Interactive Python.

In[1]: # read mid latitude atmospheres and return 2 "structured arrays"
  ...: mls = atmos1D('/data/atmos/20/mls.xy')
  ...: mlw = atmos1D('/data/atmos/20/mlw.xy', zToA=50)

Atmos1d: got p, T, air and 7 gases at 20 levels
Atmos1d: got p, T, air and 7 gases at 16 levels

In [2]: atmPlot(mls);  atmPlot([mls,mlw], 'O3', 'mb')
```

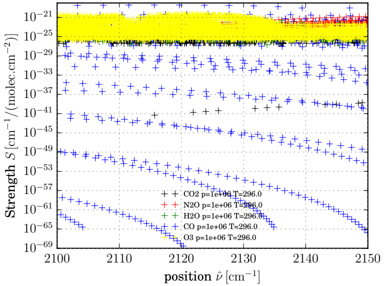


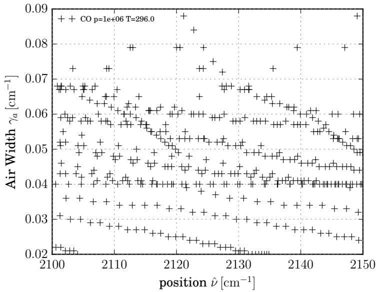


Spectroscopic Data

```
In [3]: # IASI microwindow for CO: HItran-GeiSa-exTRACT
   ...: dictLineLists = higstract('/data/geisa/97/lines',
                                  (2100,2150), molecule='main')
9771  lines of  5  molecule(s), returning a dictionary

In [4]: atlas(dictLineLists)              # plot lines (default S)
  ...:  atlas(dictLineLists['CO'],'a')   # air broadening
```





a line list is also a "Structured Array"!

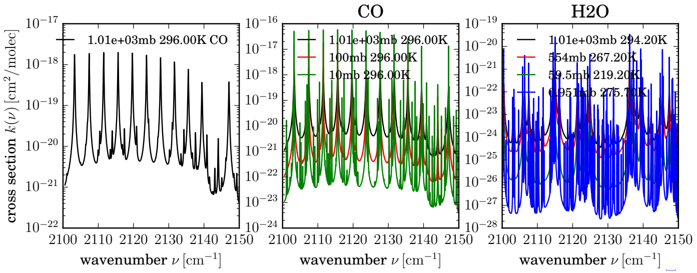
Molecular Absorption Cross Sections

```
In [5]: # CO cross section at database pressure and temperature
   ...: xs  = lbl2xs(dictLineLists['CO'])

   ...: # a list of cross sections for three pressures
   ...: xss = lbl2xs(dictLineLists['CO'], [1013,100,10,'mb'])

   ...: # a dictionary of x-section lists (for all p, T, gases)
   ...: xssDict = lbl2xs(dictLineLists, mls['p'], mls['T'])

   ...: # ... and some plots (not all are shown here)
   ...: xsPlot(xs);  xsPlot(xss);  xsPlot(xssDict)
```

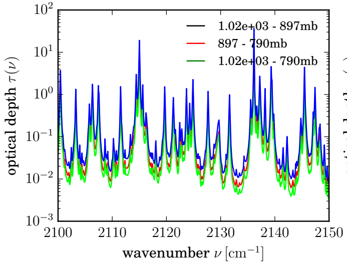


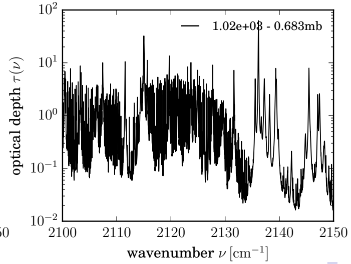
Absorption Coefficients and Optical Depths

```
In [6]: # proceed step-by-step
   ...: acList  = xs2ac(mls, xssDict) # absorption coefficients
   ...: dodList = ac2dod(acList)      # delta optical depths

In [7]: # alternatively bypass intermediate quantities, e.g.
   ...: dodList = lbl2dod(mls,dictLineLists)  # delta opt.depths

In [8]: # sum/combine optical depths and plot
   ...: odPlot([dodList[0], dodList[1]])  # the bottom layers,
   ...: odPlot(dodList[0]+dodList[1])     # ... their sum,
   ...: odPlot(dod2tod(dodList))          # and total opt.depth
```





Radiance / Intensity

```
In [8]: # radiation intensity seen by uplooking observer
   ...: vGrid, radUp = dod2ri(dodList)

   ...: # and downlooking observer at ToA
   ...: # (incl. surface @ 294K)
   ...: vGrid, radNadir = dod2ri(dodList, 180, 294.2)
```

Outline

Py4CAtS — Implementation

- Sub-classed numpy arrays `xsArray`, `acArray`, `odArray`, ... for cross sections, absorption coefficients, optical depths, ... to store "spectra" along with attributes (e.g. `xs.p` and `xs.t`)
- cgs units used internally
- Numerics:
  - Complex error function: Humlicek82 Weideman94 combination
  - Multigrid line-by-line (fine grid near line center only)
  - Schwarzschild integral: B linear or exponential in τ
- Limitations:
  - Plots for quicklook only, not "publication-ready"
  - Plane-parallel atmosphere, no scattering, continua, ...
  - No "package" yet, no distutils etc. (coming soon)

Py4CAtS — Goodies

- Option parser module `command_parser.py`
  - `argparse` only available “recently”
  - no support for range of (real) numbers
- advanced extended input output utilities `aeiou.py`
  - function `awrite(array, file=None, ...)`
    — alternative to numpy’s `savetxt`
    — array can be a list of arrays, no file required
  - support for structured array input, manipulation
- Module `pairTypes.py`
  Example: `xLimits = Interval(10.,20.)`

Summary and Outlook

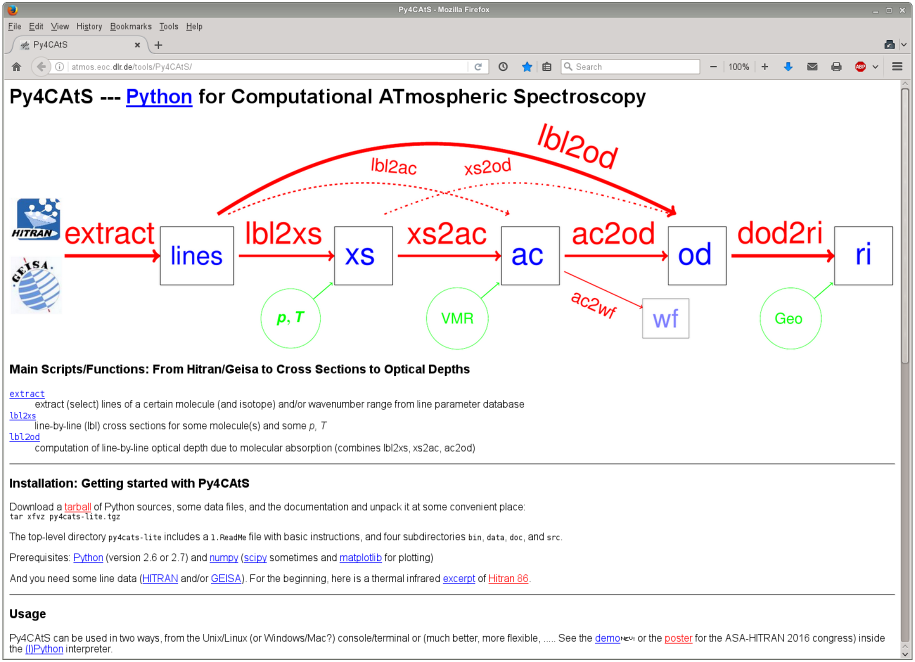
Py4CAtS modules / scripts / functions for Hitran / Geisa to optical depths and radiance via cross sections and absorption coefficients

ToDo's

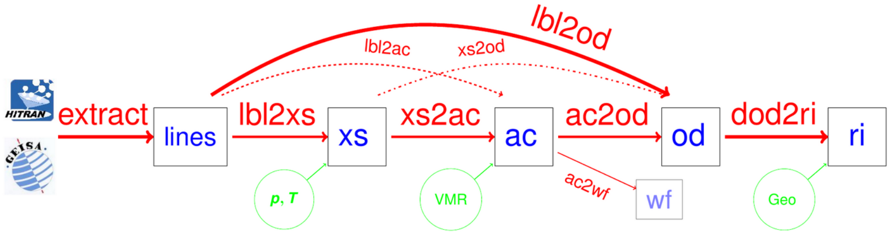
- Packaging with distutils
- Subclassed radiance array `riArray`
- Consolidate the various subclassed numpy arrays
- Python 3

- Spectral response and convolution
- Exploit astropy for unit conversions
- "Simple" scattering solvers**?¿?**

`http://atmos.eoc.dlr.de/tools/Py4CAtS/`


Py4CAtS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Py4CAtS
atmos.eoc.dlr.de/tools/Py4CAtS/
Search
100%

Py4CAtS --- Python for Computational ATmospheric Spectroscopy



Main Scripts/Functions: From Hitran/Geisa to Cross Sections to Optical Depths

`extract`
: extract (select) lines of a certain molecule (and isotope) and/or wavenumber range from line parameter database

`lbl2xs`
: line-by-line (lbl) cross sections for some molecule(s) and some *p*, *T*

`lbl2od`
: computation of line-by-line optical depth due to molecular absorption (combines lbl2xs, xs2ac, ac2od)

Installation: Getting started with Py4CAtS

Download a tarball of Python sources, some data files, and the documentation and unpack it at some convenient place:

```
tar xfvz py4cats-lite.tgz
```

The top-level directory `py4cats-lite` includes a `1.ReadMe` file with basic instructions, and four subdirectories `bin`, `data`, `doc`, and `src`.

Prerequisites: Python (version 2.6 or 2.7) and numpy (scipy sometimes and matplotlib for plotting)

And you need some line data (HITRAN and/or GEISA). For the beginning, here is a thermal infrared excerpt of Hitran 86.

Usage

Py4CAtS can be used in two ways, from the Unix/Linux (or Windows/Mac?) console/terminal or (much better, more flexible, ... See the demo NEW! or the poster for the ASA-HITRAN 2016 congress) inside the (I)Python interpreter.