Py4CAtS — PYthon for Computational ATmospheric Spectroscopy

Franz Schreier, Sebastián Gimeno García, and Jian Xu
DLR — Remote Sensing Technology Institute, Oberpfaffenhofen, GERMANY

Introduction

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 parameterized models

Why Python?

- Rapid prototyping
- Most (?) lbl models kind of “black-box”
- Difficult to see intermediate quantities

InfraRed Radiative Transfer

Schwarzschild equation: $I(\nu)$ radiance/intensity at wavenumber ν

$$I(\nu) \;=\; I_b(\nu)\, e^{-\tau_b(\nu)} \;+\; \int_0^{\tau_b(\nu)} B(\nu, T(\tau'))\, e^{-\tau'}\, d\tau'$$

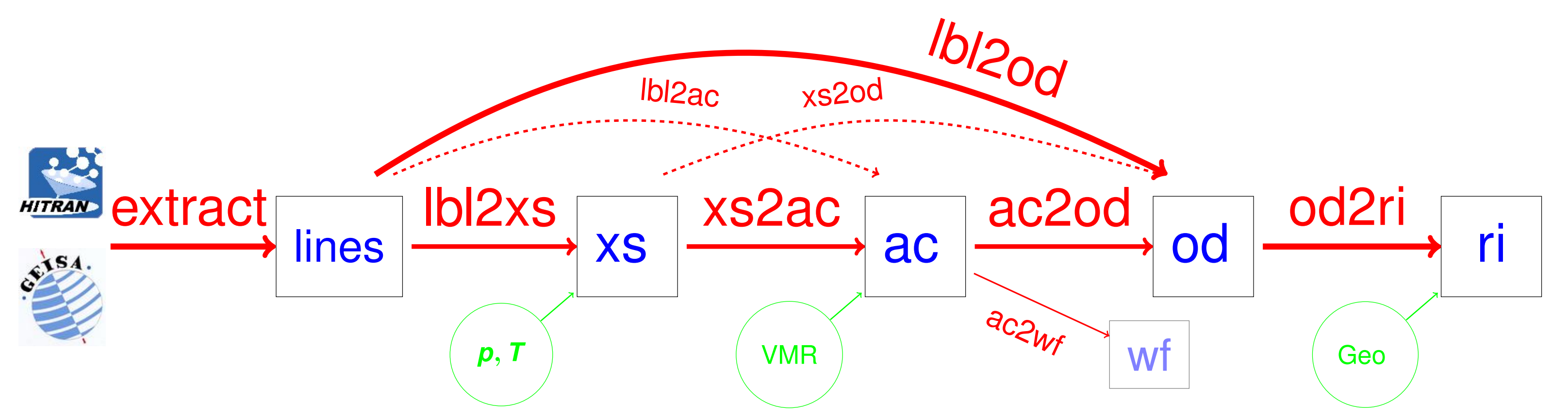
Beer's law: transmission $\mathcal{T}$ and optical depth τ

$$\mathcal{T}(\nu, s) = e^{-\tau} = \exp\left(- \int_0^s ds' \sum_m k_m(\nu, p(s'), T(s'))\; n_m(s') \right)$$

Absorption coefficient α and cross section k: line-by-line

$$k(\nu, p, T) = \sum_l S_l(T)\; g(\nu;\, \hat{\nu}_l, \gamma_l(p, T))$$

Line shape function g: Voigt, Lorentz, VVH, VVW, …
Line parameters: position $\hat{\nu}_l$, strength S_l, width(s) γ_l, …



Py4CAtS — Implementation

- (Numeric and Scientific) *Python* version of Fortran 2008 “Generic Atmospheric Radiation Lbl Ir Code” GARLIC [4]
- Series of scripts for IR & μWave radiative transfer, e.g.,
  - `extract` lines of relevant molecules in the spectral range of interest
  - `lbl2xs` line-by-line cross sections for given pressure(s) & 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 Schwarzschild eq. along line-of-sight to radiance/intensity
  - `ac2wf` compute weighting functions $\partial\mathcal{T}/\partial z \propto \alpha\mathcal{T}$
  - …… and some shortcuts, e.g., `lbl2ac` or `lbl2od`
- *New:* functions accessible within (I)Python shell
- 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`)
- Numerics:
  - Complex error function: Humlíček [1] – Weideman [5] combination [3]
  - Multigrid line-by-line (fine grid near line center only) [2]
  - 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)

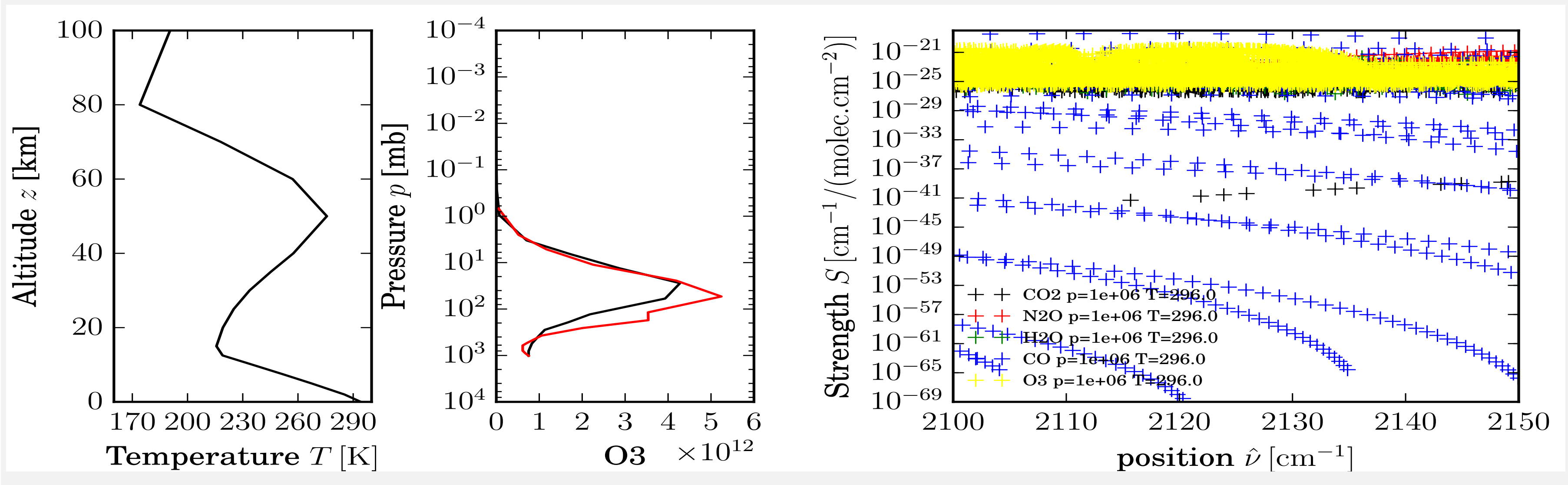
References:

[1] J. Humlíček. Optimized computation of the Voigt and complex probability function. *JQSRT*, 27:437–444, 1982.
[2] F. Schreier. Optimized evaluation of a large sum of functions using a three-grid approach. *Comp. Phys. Comm.*, 174:783–802, 2006.
[3] F. Schreier. Optimized implementations of rational approximations for the Voigt and complex error function. *JQSRT*, 112:1010, 2011.
[4] F. Schreier, S. Gimeno García, P. Hedelt, M. Hess, J. Mendrok, M. Vasquez, and J. Xu. GARLIC – a general purpose atmospheric radiative transfer line-by-line infrared-microwave code: Implementation and evaluation. *JQSRT*, 137:29–50, 2014.
[5] J.A.C. Weideman. Computation of the complex error function. *SIAM J. Num. Anal.*, 31:1497–1518, 1994.

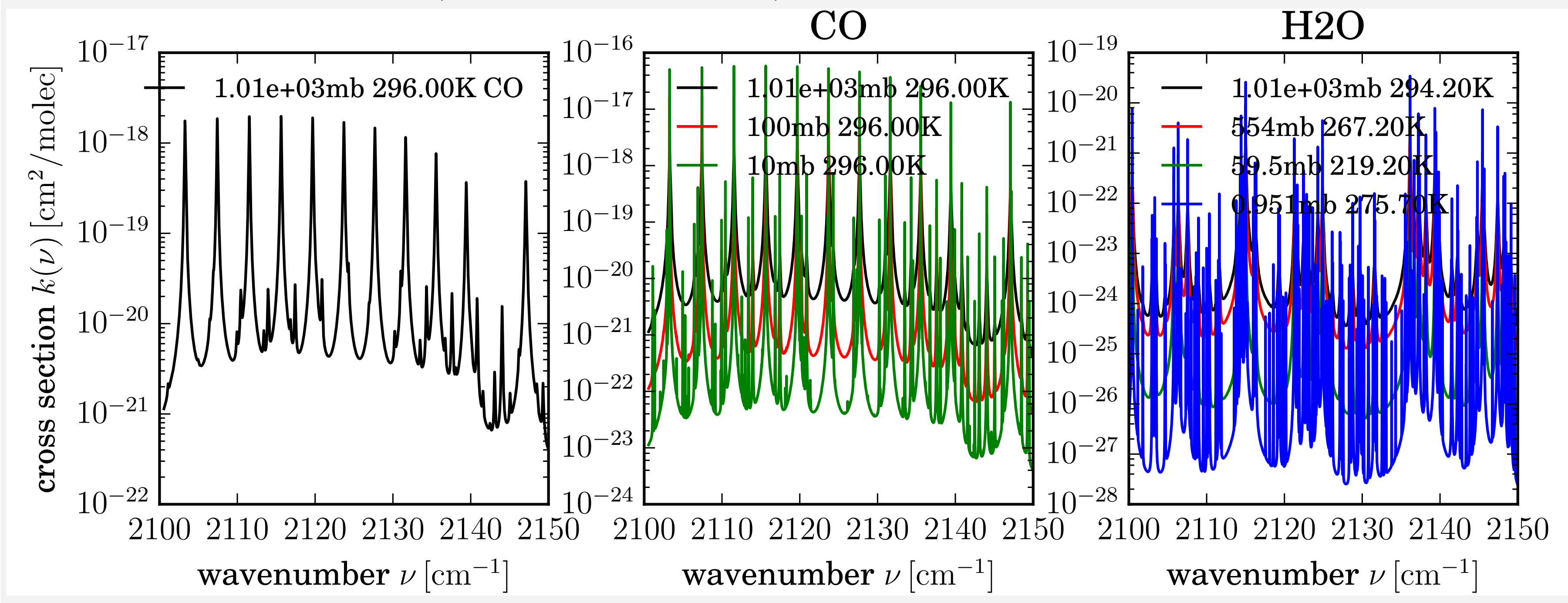
`http://atmos.eoc.dlr.de/tools/Py4CAtS/`

IPython Demo (output largely deleted)

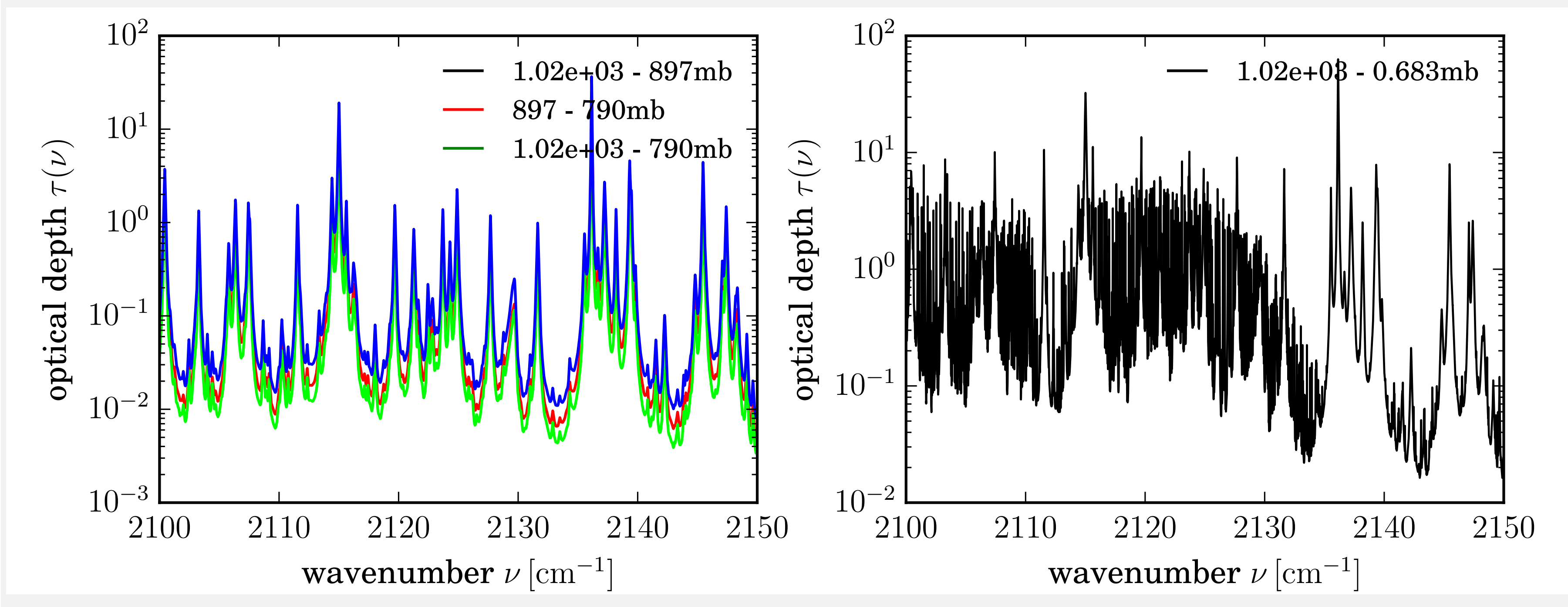
```
Python 2.7.3 (default, Apr 14 2012, 08:58:41) [GCC]
IPython 2.0.0 -- An enhanced Interactive Python.

In[1]: # get two mid latitude atmospheres
  ...: 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]: # IASI microwindow for CO retrieval: HItran-GeiSa-exTRACT
   ...: dictLineLists = higstract('/data/geisa/87/lines',
                                  (2100,2150), molecule='main')
9771  lines of  5  molecule(s), returning a dictionary

In [3]: atmPlot(mls);  atmPlot([mls,mlw], 'O3', 'mb')
   ...: atlas(dictLineLists) # plot line data (default strength)
```



```
In [4]: # 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)
```



```
In [5]: # proceed step-by-step
   ...: acList  = xs2ac(mls, xssDict) # absorption coefficients
   ...: dodList = ac2dod(acList)      # delta optical depths

In [6]: # alternatively bypass intermediate quantities, e.g.
   ...: dodList = lbl2dod(mls,dictLineLists)  # delta opt.depths

In [7]: # 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
```



```
In [8]: # radiation intensity seen by uplooking observer at BoA
   ...: vGrid, radUp = dod2ri(dodList)
   ...: # and downlooking observer at ToA (incl. surface @ 294K)
   ...: vGrid, radNadir = dod2ri(dodList, 180, 294.2)
```

“Main” functions can be used from Unix (Windows?) console, too!

Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft