UNIVERSITY OF CENTRAL FLORIDA

Transit Users Manual

A RADIATIVE-TRANSFER CODE FOR PLANETARY ATMOSPHERES

Authors:
Patricio CUBILLOS
Jasmina BLECIC

Supervisor:
Dr. Joseph HARRINGTON

June 16, 2015

Contents

1 Team Members

- Patricio Cubillos[1], University of Central Florida (pcubillos@fulbrightmail.org).
- Jasmina Blecic, University of Central Florida (jasmina@physics.ucf.edu).
- Joseph Harrington, University of Central Florida (jh@physics.ucf.edu).
- Madison Stemm, University of Central Florida (email@email.com).
- Andrew S. D. Foster, University of Central Florida (andrew.scott.foster@gmail.com).
- Patricio M. Rojo, Universidad de Chile (pato@oan.uchile.cl).

2 Introduction

This document describes the University of Central Florida's computer program `Transit`. The program calculates the transmission or emission flux spectrum of a planetary atmosphere with application to extrasolar-planet transit and eclipse observations, respectively. `Transit` calculates the spectra by solving for 1-dimensional line-by-line radiative-transfer equation for a plane-parallel atmospheric model. A separate document **(FINDME: point to file)** describes in detail the theory and assumptions adopted by `Transit`.

This manual describes the `Transit` program usage for the user (Sections 3 through 8) and potential developer (Sections 9-10). Section 3 indicates how to obtain the code and the system requirements. Section 6 describes the inputs necessary for execution. Section 7 describes the output files produced by the code. Section 8 show an example execution of the code. Section **??** describes the program design. Sections 9 and 10 details the data structures and routines, respectively.

2.1 Transit Package Overview

`Transit` is a C code, written in a modular, object-oriented style. It was originally developed at Cornell University by Patricio Rojo, a former Ph.D. student of Dr. Joseph Harrington. The code handled the case of transmission spectrum, where stellar light is absorbed as it travels tangentially across the limb of a planetary atmosphere, as observed during an exoplanet transit observation. Subsequently, we —the exoplanet group at the University of Central Florida— have modified the code to add planetary emission spectra calculation (as observed during a secondary-eclipse observation), include multiple opacity line lists, and incorporate it to the Bayesian Atmospheric Radiative Transfer (`BART`[2]) project, which constrains exoplanet atmospheric properties in a Bayesian framework, given a set of eclipse or transit-depth measurements.

`BART` combines `Transit` with the Thermochemical Equilibrium Abundances (`TEA`[3]) module, which calculates abundances of gaseous species, and the Multi-Core Markov-chain Monte Carlo (`MCcubed`[4]) statistical module, which assesses the temperature and molecular-abundances posterior distributions, given a set of observations.

[1] https://github.com/pcubillos/
[2] github.com/joeharr4/BART
[3] github.com/dzesmin/TEA
[4] github.com/pcubillos/MCcubed

The current `Transit` package consists of three main modules: `Transit` **(FINDME: change name to** `RT`**)**, a C module that calculates the emission or transmission spectrum for a planetary atmosphere by solving the radiative-transfer equation. `pu`, a library of C utility functions used by `Transit`. `Pylineread`, a Python module that converts molecular line-opacity information (line transitions and partition functions) from online-available databases into the transit line information (TLI) format for later use by `Transit`.

The Transit module is organized as follow:

```
transit
├── transit
│   ├── src
│   └── include
├── pylineread
│   ├── src
│   ├── inputs
│   └── examples
├── pu
│   ├── src
│   └── include
├── run
├── examples
└── doc
```

2.2 License

Transit, a code to solve for the radiative-transfer equation for planetary atmospheres.

This project was completed with the support of the NASA Planetary Atmospheres Program, grant NNX12AI69G, held by Principal Investigator Joseph Harrington. Principal developers included graduate students Patricio E. Cubillos and Jasmina Blecic, programmer Madison Stemm, and undergraduate Andrew S. D. Foster. The included 'transit' radiative transfer code is based on an earlier program of the same name written by Patricio Rojo (Univ. de Chile, Santiago) when he was a graduate student at Cornell University under Joseph Harrington.

We welcome your feedback, but do not guarantee support. Please send feedback or inquiries to:

Patricio Cubillos <pcubillos[at]fulbrightmail.org>
Jasmina Blecic <jasmina[at]physics.ucf.edu>
Joseph Harrington <jh[at]physics.ucf.edu>

or alternatively,

Joseph Harrington, Patricio Cubillos, and Jasmina Blecic
UCF PSB 441
4111 Libra Drive
Orlando, FL 32816-2385
USA

Thank you for using transit!

3 Installation

3.1 System Requirements

Transit was developed on a Unix/Linux machine using Python 2.7.6, Numpy 1.8.2, and mpi4py 1.3.1. **(FINDME: Ask someone who knows about C and Fortran requirements.)**

- Python.
- NumPy.
- matplotlib.
- mpi4py.
- An MPI distribution (MPICH preferred).
- A C compiler **(FINDME: any preference?)**.
- A Fortran compiler **(FINDME: any preference?)**.
- GSL **(FINDME: give specifics)**.

3.2 Install and Compile

To obtain the `Transit` code download the latest stable version from the releases[5] page. Alternatively, clone the repository to your local machine with the following terminal commands. First, create a working directory to place the code:

```
cd
mkdir tmp/
mkdir tmp/transit_demo
cd tmp/transit_demo
```

Clone the repository to your working directory:

```
git clone https://github.com/pcubillos/transit transit
```

Compile the `pu` and `transit` code (in this order):

```
cd transit/pu
make
cd ../transit
make
```

Compile the `pylineread` fortran code:

```
cd ../pylineread/src/fortran
make
```

To remove the program binaries, execute (from the respective directories):

```
make clean
```

Note that there will be warnings.

[5] github.com/pcubillos/transit/releases

4 Quick Example

The following script quickly lets you calculate a methane emssion spectrum from the terminal. To start, follow the instructions in Section 3.2 to install and compile the code. Now, create a working directory to place the files and execute the programs:

```
cd
cd tmp/transit_demo/
mkdir run/
cd run/
```

Download the methane line-transition database from the HITRAN server:

```
wget -user=HITRAN -password=getdata -N https://www.cfa.harvard.edu/HITRAN/HITRAN2008/H
unzip 06_hit08.zip
```

Copy the `pylineread` configuration file to the run directory and execute pylineread to generate the transition-line-information file:

```
cp ../transit/pylineread/examples/demo/pyline_demo.cfg .
../transit/pylineread/src/pylineread.py -c pyline_demo.cfg
```

Copy the Transit configuration file to the run directory and execute Transit to compute the spectrum:

```
cp ../transit/transit/examples/demo/transit_demo.cfg .
../transit/transit/transit -c transit_demo.cfg
```

To check out the results, run this Python script to generate the plot in Figure 1:

```
import matplotlib.pyplot as plt
import sys
sys.path.append("../transit/scripts/")
import readtransit as rt
wlength, flux = rt.readspectrum("CH4_demo_spectrum.dat.—Flux", 0)

plt.figure(0, (8,5))
plt.clf()
plt.title("Methane Emission Spectrum")
plt.plot(wlength, flux, "b")
plt.xlabel("Wavelength  (um)")
plt.ylabel("Flux  (erg s—1 cm—1)")
plt.show()
```

5 Quick Walkthrough

Transit consists of three packages — pylineread, transit, and pu—. The pylineread and transit packages provide the program executables to calculate an atmospheric spectrum, the pu package provides utility functions for the transit package. The pylineread and transit programs allow for command-line arguments or a configuration file (preferred) to configure the calculation specifics

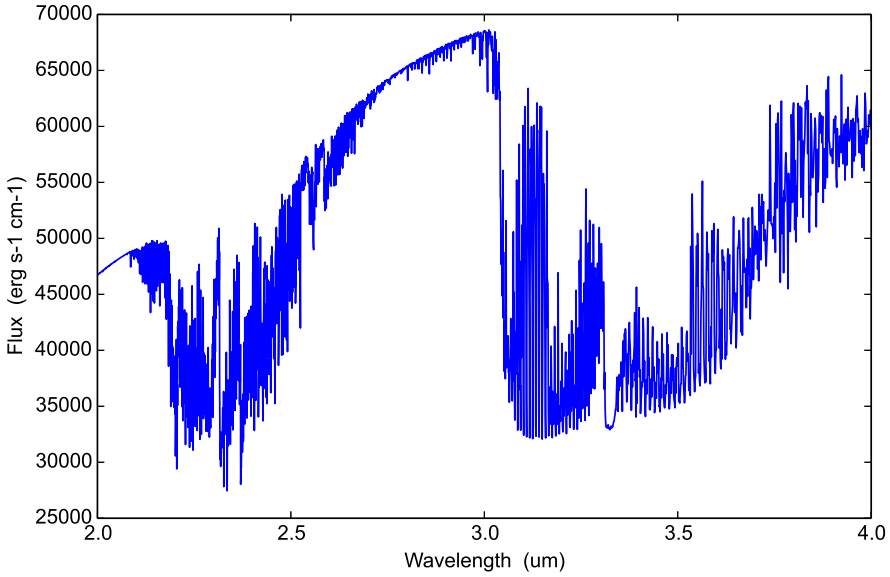

Figure 1: Methane emission spectra.

(e.g, spectrum sampling boundaries, resolution, input and output file locations and names, verbosity, etc).

The spectrum calculation is broken in two steps: first, pylineread processes species line-transition information files (as downloaded from online databases), extract the necessary data, and writes it to a binary transition-line-information (TLI) file. Then the transit program computes the spectra for the given atmospheric model and species opacities.

The inputs for pylineread are (1) the line-transition database files of the species of interest, and (2) their corresponding partition-function files. Pylineread outputs a binary TLI file with the line-transitions' wavelength, lower-state energy, oscillator strength, and isotpe index; and a tabulated list of partition-function values as a function of temperature.

The inputs for transit are (1) an atmospheric file, (2) a TLI file (output from pylineread), (3) a collision-induced-absorption (CIA), (4) a molecular information file, and optionally (5) an opacity file. See section 6.2 for further details on the transit input files and their required formats.

The atmospheric file defines the atmospheric model, indicating the species present, and their physical properties (radius, pressure, temperature, and species mole mixing ratio) given in a layer-by-layer style. The input TLI file is the output from pylineread. The CIA file is a tabulated list of CIA opacities as function of wavenumber and temperature. The molecular information file contains additional species properties, is not expected to change (mass and collision diameter). This file is provided by the Transit module. The optional opacity file provides a tabulated list of line-transition opacities, as a function of temperature, pressure, and wavenumber, for each species. This file can be computed by the transit prgram and further used as input to speed up the calculations (interpolate instead of doing the line-by-line calculation).

The main output of the transit program is the atmospheric spectrum. For transit geometry, transit computes the transmission spectrum, returning the modulation spectrum (Rp/Rs as function of wavelength). For eclipse geometry, transit computes the emission intensity spectrum for a set of incident angles, combining them to return a hemisphere integrated flux spectrum. Additional (optional) outputs are (2) the opacity file (see above), (3) a 'toomuch' file with the layer at which

the optical deth reached a specified maximum threshold, (4) a sampling file with information on the wavelength and layer sampling, and for eclipse geometry, (5) the intensity emission spectra as a function of incident angle. See section 7.2 for further details on the output files.

6 Program Inputs

This Section describes the command-line arguments and input files required by the the `Pylineread` and `Transit` modules, and any additional information needed to properly configure the modules.

6.1 Pylineread

The executable `transit/pylineread/src/pylineread.py` is the main `Pylineread` routine. `Pylineread` runs from the shell taking a set of command-line arguments. Alternatively, the command-line arguments can be specified using a configuration file. Before running `Pylineread`, the user needs to obtain two types of input files: opacity line-list database files and partition-function files.

6.1.1 Pylineread Command-Line Arguments

To display the complete list of command-line arguments, cd into the `Pylineread` source folder (`transit/pylineread/src`) and execute:
`./pylineread.py --help`

Optional Arguments:

`-h, --help`
Show the help message and exit.

`-c FILE, --config_file FILE`
Configuration filename (string).

`-v VERB, --verbose-level VERB`
Verbosity level (Integer) [default: 2].

`-q, --quiet`
Set verbosity level to 0.

Database Arguments:

`-o OUTPUT, --output OUTPUT`
Output TLI filename (string) [default: 'output.tli'].

`-d DB_LIST, --database DB_LIST`
Path (string) to the input line-transition database file(s).

`-p PART_LIST, --partition PART_LIST`
Path (string) to the auxiliary partition-function file(s).

`-t DBTYPE, --dbtype DBTYPE`
Database type (string). 'ps' for Partridge & Schwenke's H2O; 'hit' for HITRAN and HITEMP; or 'ts' for Schwenke's TiO.

Wavelength Arguments:

`-i IWAV, --wav-init IWAV`
Initial wavelength (microns) [default: 1.0].

`-f FWAV, --wav-final FWAV`
Final wavelength (microns) [default: 2.0].

6.1.2 Configuration File

A configuration file presents an alternative to the command-line arguments to specify the input variables. The examples folder (`transit/pylineread/examples/`) provides a configuration-file sample, `pyline_example.cfg`, and is further explained below:

```
[Parameters]
# Multiple-value arguments can be set separated either by a white space or
# a line break.

# This is the list (and source) of the currently supported Line Lists:
# Partridge and Schwenke (H2O):
#   - http://kurucz.harvard.edu/molecules/h2o/h2ofastfix.bin
# HITRAN and HITEMP:
#   - http://www.cfa.harvard.edu/hitran/
# Schwenke (TiO):
#   - http://kurucz.harvard.edu/molecules/tio/tioschwenke.bin

# With the correspoding partition functions:
#   - http://kurucz.harvard.edu/molecules/h2o/h2opartfn.dat
#   - Total Internal Partition Sums (TIPS) *
#   - http://kurucz.harvard.edu/molecules/tio/tiopart.dat
# (*): Transit incorporates a modified version of the TIPS code

# Path(s) to the database line lists:
# (assuming that the user put the files in a databases folder)
db_list  = .../databases/Schwenke-H2O/h2ofastfix.bin
           .../databases/HITEMP/CO2/02_2500-3000_HITEMP2010.par

# Path(s) to the partition function files:
part_list= .../databases/Schwenke-H2O/h2opartfn.dat
           .../transit/pylineread/src/fortran/lib/TIPS_BART

# Type of each input database:
#   hit:  HITRAN and HITEMP
#   ps:   Partridge and Schwenke H2O
#   ts:   Schwenke TiO
dbtype   = ps hit

# The output filename
```

```
output    = PSandHIT_2.0-4.0um.tli

# Initial wavelength in microns:
iwav      = 2.0

# Final wavelength in microns:
fwav      = 4.0

# Verbosity level (max number 20)
verb      = 10
```

`Pylineread`'s configuration files follow the ConfigParser[6] format. The `[Parameters]` line defines the section read by ConfigParser. Multiple arguments can be separated either with blank spaces or put in separated lines. The db_list, part_list, and dbtype arguments must have the same number of values, and should be input in the corresponding order. We recommend that the output name carry a `.tli` extension.

6.1.3 Opacity Line-List Files

Table 1 lists the line-transition databases that are currently supported by pylineread and source URLs so the user can obtain such files.

Table 1: Line-Transition Databases for Pylineread

Database	Molecule	URL
Partridge & Schwenke	H_2O	kurucz.harvard.edu/molecules/h2o/h2ofastfix.bin
HITRAN	H_2O, CO_2, CH_4, + others	cfa.harvard.edu/hitran
HITEMP	H_2O, CO_2, CO	cfa.harvard.edu/hitran
Schwenke	TiO	kurucz.harvard.edu/molecules/tio/tioschwenke.bin

6.1.4 Partition-Function Files

Table 2 lists the partition-function files and source URLs for the databases listed in Table 1. For the HITRAN and HITEMP databases, `Pylineread` provides a a modified version of the Total Internal Partition Sums (TIPS, Laraia et al., 2011) code[7], to calculate the partition functions.

Table 2: Partition-Function Data for Pylineread

Database	Temperature range (K)	URL
Partridge & Schwenke	10-6000	kurucz.harvard.edu/molecules/h2o/h2opartfn.dat
HITRAN & HITEMP	70-3000	transit/pylineread/src/fortran/lib/TIPS_BART
Schwenke	10-6000	kurucz.harvard.edu/molecules/tio/tiopart.dat

[6] docs.python.org/2/library/configparser.html
[7] faculty.uml.edu/robert_gamache/software/index.htm

6.1.5 HITRAN Data File

The HITRAN data file, `transit/pylineread/inputs/hitran.dat`, is an additional file used by `Pylineread` to obtain known physical properties of the species from the HTRAN and HITEMP databases. This file is automatically read by `Pylineread` and, thus, does not need to be given as an argument. There is also no need to modify this file by the user, as the information included does not vary. The layout of the `hitran.dat` is given below:

```
# ID:          HITRAN molecule ID
# Molecule:    Molecule name
# Iso:         Isotope AFGL code
# gi:          State-independent Statistical weight
# Iso ratio:   Isotopic ratio
# Iso mass:    Isotopic mass (amu)

# ID Molecule Iso        gi         Iso ratio    Iso mass
 1   H2O      161         1         9.973e-01    18.0105646
 1   H2O      181         1         1.999e-03    20.014811
 1   H2O      171         6         3.719e-04    19.014781
 1   H2O      162         6         3.107e-04    19.016841
 1   H2O      182         6         6.230e-07    21.021088
 1   H2O      172        36         1.158e-07    20.021058
 2   CO2      626         1         9.842e-01    43.98982920
 2   CO2      636         2         1.106e-02    44.99318400
...  ...      ...        ...           ...           ...
```

The first, second, and third columns are the molecular ID, molecule names, and isotope ID, respectively, as given by the HITRAN database. The fourth column (g_i) are the state-independent statistical weights (from Fischer et al., 2003), the fifth column are the isotopic ratios (from Šimečková et al., 2006), and the last column are the isotopic masses (calculated from Lide, 2008).

6.2 Transit

The executable `transit/transit/transit` is the main `Transit` program. `Transit` runs from the shell taking a set of command-line arguments. Alternatively, the command-line arguments can be specified using a configuration file. Before running `Transit`, the user needs to obtain three types of input files, an atmospheric file, transit-line information (TLI) files (created by `Pylineread`), and collision-induced absorption (CIA) files. Additionally, `Transit` uses a molecules data file (included in the code). Lastly, an optional opacity file can be used to provide a pre-calculated table of opacities (this file can be created by `Transit` itself).

6.2.1 Transit Command-Line Arguments

To display the complete list of command-line arguments, cd into the `Transit` source folder (`transit/transit`) and execute:

`./transit --help`

General Options:

`-h, --help`
Display the list of command-line arguments.

`-V, --version`
Display Transit's version number.

`-q, --quiet`
Set the verbosity level to the minimum.

`-v, --verb=<verb>`
Set the verbosity level (integer) to <verb>. [default: 2].

`-c, --config_file=<file>`
Read command-line arguments from <file>.

Input/Output Options:

`-o, --output=<outfile>`
Output file to store the model spectrum. [default: Print to standard output].

`--atm=<atmfile>`
Input atmospheric info file.

`--linedb=<linedb>`
Input line information (TLI) file(s) (as given by 'pylineread').

`--cia=<filenames>`
Input collision-induced absorption (CIA) opacities file(s) (comma-separated list if more than one file).

`--outtoomuch=<filename>`
Ouput file to store the depth (as a function of wavelength) where the optical depth reached 'toomuch'.

`--outsample=<filename>`
Output file to store the layer and wavenumber sampling information. A dash (-) indicates standard input. [Default: NULL].
(FINDME: Standard input?)

`--molfile=<filename>`
Input file with the molecular information. [default: ../inputs/molecules.dat].

Radius Options:

`--raddelt=<spacing>`
Radius spacing. If set, resample the atmospheric layers to an equidistant sampling array. [default: -1].

`--radlow=<radius>`
Lower radius. If 0, use atmospheric-file minimum. [default: 0].

`--radhigh=<radius>`
Higher radius. If 0, use atmospheric-file maximum. [default: 0].

--radfct=<factor>
Radius units conversion factor to cm. E.g., if the radii are given in m, then radfct=100. If 0, use the atmospheric-file factor. [default: 0].

Atmospheric Options:

--allowq=<value>
Maximum allowed cumulative-abundance departure from 1.0. [default: 0.00001].

--refpress=<value>
Atmospheric pressure at the planet's reference 'surface'.

--refradius=<value>
Atmospheric radius at the planet's reference 'surface'.

--gsurf=<value>
Planetary surface (bulk) gravity (in cm/s^2).

--qmol=<NULL>
List of molecule names to modify their abundace with qscale.

--qscale=<NULL>
log10-abundance scale factors for qmol molecules.

Wavelength-Array Options (in wlfct units):

--wllow=<wavel>
Lower wavelength boundary. [default: Minimum in TLI file].

--wlhigh=<wavel>
Upper wavelength boundary. [default: Maximum in TLI file].

--wlfct=<factor>
Wavelength units conversion factor to cm. E.g., for wavelengths given in microns, wlfct=1e-4. [default: 1.0].

Wavenumber-Array Options (in wnfct units):

--wnlow=<waven>
Lower wavenumber boundary. [default: wavelength maximum boundary].

--wnhigh=<waven>
Upper wavenumber boundary. [default: wavelength minimum boundary].

--wndelt=<spacing>
Wavenumber array spacing. [default: 1].

--wnosamp=<Integer>
Wavenumber oversampling factor. [default: $2160 = 2^4 \times 3^3 \times 5$].

--wnfct=<factor>
Wavenumber units conversion factor to cm^{-1}. [default: 1.0].

Voigt Profile Calculation Options:

--ndop=<integer>
Number of Doppler-broadening width samples [default: 40].

--nlor=<integer>
Number of Doppler-broadening width samples [default: 40].

--dmin=<float>
Minimum Doppler-broadening width (in cm-1) [default: 1e-3].

--dmax=<float>
Maximum Doppler-broadening width (in cm-1) [default: 0.25].

--lmin=<float>
Minimum Lorentz-broadening width (in cm-1) [default: 1e-4].

--lmax=<float>
Maximum Lorentz-broadening width (in cm-1) [default: 10.0].

--nwidth=<number>
Number of the max-widths (the greater of Voigt or Doppler widths) that needs to be contained in a calculated profile. [default: 20].

Extinction-Coeficcient Calculation Options:

--ethresh=<threshold>
Minimum extinction-coefficient ratio (w.r.t. maximum in a given layer) to consider in the calculation. [default: 1e-8].

--cloudrad=<radup,raddown>
If set (in conjunction with cloudext), define a cloud layer (gray opacity component) where a gray opacity component linearly increases from radup to raddown, where the opacity will reach cloudext. From raddown below keep constant opacity. Use '--cloudfct' units; if not defined, use radfct.

--cloudext=<extinction>
Maximum extinction of the cloud, which opacity will linearly increase from 'radup' to 'raddown'.

--cloudfct=<factor>
Radius units conversion factor to cgs for --cloudrad.

--detailext=<filename:wn1,wn2,...>
Save extinction at specified wavenumbers in filename.

--detailcia=<filename:wn1,wn2,...>
Save CIA extinction at specified wavenumbers in filename.

--saveext=<filename>
Save extinction array in this file which won't need to be recomputed if only the radius scale (scale height) changes.

Opacity-Grid Options:

--opacityfile=<filename>
Filename to read/save the opacity grid.

--tlow=<temperature>

Lower temperature boundary for the opacity grid (in kelvin). [default: 500].

`--thigh=<temperature>`
Upper temperature boundary for the opacity grid [default: 3000].

`--tempdelt=<spacing>`
Temperature sample spacing (in Kelvin degrees). [default: 100.0].

`--justOpacity=<boolean>`
If set, end execution after the opacity-grid calculation.

Optical-Depth Options:

`-s, --solution=<sol_name>`
Name of the kind of output solution (eclipse or transit). [default: eclipse].

`--toomuch=<optdepth>`
Maximum optical depth to calculate (at each wavenumber). [default: 20].

`--taulevel=<integer>`
Calculate the lightray path with a constant (1) or variable (2) index of refraction. [default: 1].
(FINDME: taulevel=1 is the only working option.)

`--modlevel=<integer>`
On the modulation calculation, If modlevel=1, do not consider limb darkening, if modlevel=-1 do not consider limb darkening and additionally only returns the modulated radius at which extinction becomes one. (default: 1).

`--detailtau=<filename:wn1,wn2,..>`
Save optical depth at specified wavenumbers in filename.

Geometry Options:
`--starrad=<radius_sun>`
Stellar radius in solar radius. (default: 1.125).

`--gorbpar=<smaxis,time,incl,ecc,long_node,arg_per>`
Orbital parameters. Use the above order. Default: 1, 0, 0, 0, 0, 0.

`--gorbparfct=<unitsof:smaxis,time,incl,ecc,long_node,arg_per>`
Units convertion factors to the cgs system of the orbital parameters. Same order of g-orbpar. Default: AU, hours, deg, 1, deg, deg.

`--transparent`
If selected, the planet will have a maximum optical depth given by toomuch, it will never be totally opaque.

`--raygrid=<(null)>`
List of incident angles to calculate the emission intensity spectrum (default: 0 20 40 60 80).

(FINDME: re-word this:)
The next sub section describe how the CLA relate to the radiative transfer aspect/case/problem.

6.2.1.1 Spectrum Wavenumber Sampling

Internally, `transit` samples the spectrum in a equi-spaced wavenumber space, in CGS units. However, the user can specify the spectrum boundaries either in wavenumber or wavelength. The '**`wnfct`**' and '**`wlfct`**' arguments allow the user to specify the units of the input wavenumber and wavelength variables, respectively.

`Transit` implements a 'dynamic' wavenumber sampling, adjusting the sampling at each layer to avoid oversampling or undersampling the line Voigt profiles, allowing for an efficient extinction coefficient calculation at all layers (see Figure 2).

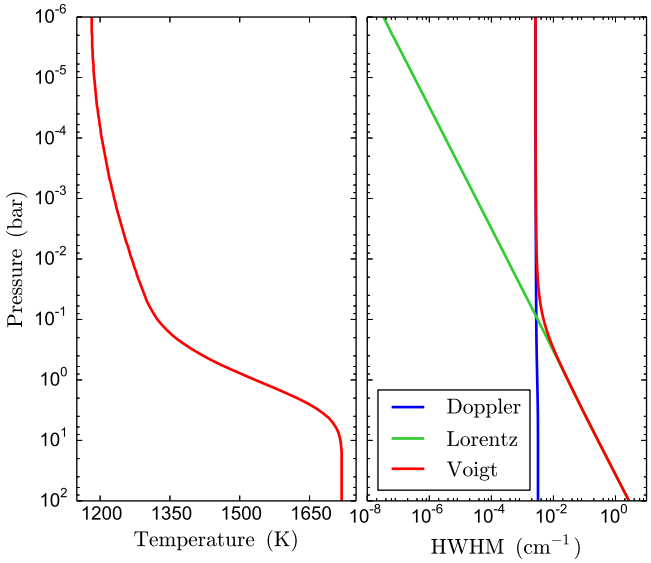

Figure 2: Doppler and Lorentz half-width at half maximum (HWHM) line broadening for the water molecule at 11 μm, for a H_2/He-dominated atmosphere. Since the Lorentz profile width is proportional to the pressure, the Voigt profile HWHM varies over several orders of magnitude between the top and bottom of the atmosphere.

The '**`wndelt`**' argument defines the 'coarse' sampling interval ($\Delta\nu_c$), which is the sampling of the output spectra. Together with the oversampling-factor argument, '**`wnosamp`**', `Transit` defines the 'fine' sampling interval as: $\Delta\nu_f$ = **`wndelt / wnosamp`**. The dynamic sampling interval ($\Delta\nu_d$) lies between the fine and coarse sampling interval.

Initially, Transit pre-computes Voigt profiles, for a set of Lorentz and Doppler widths, over the fine sampling. Then, these profiles are decimated to the dynamic sampling to calculate the extinction coefficient. Lastly, the extinction coefficient is downsampled to the coarse sampling. This requires that the dynamic sampling be an integer factor of the fine sampling ($\Delta\nu_d = f_1\Delta\nu_f$, with $f_1 \in \mathbb{N}$), and the coarse sampling be an integer factor of the dynamical sampling ($\Delta\nu_c = f_2\Delta\nu_d$, with $f_2 \in \mathbb{N}$). Noting that **`wnosamp`** $= f_1 f_2$, favorable values of **`wnosamp`** are thus numbers with a large number of integer divisors.

The '**`resolv`**' argument **(FINDME: not implement yet)** sets the minimum number of spectral points to sample a profile's HWHM, determining the dynamic sampling interval for each layer. Given the wavenumber range and the layer's temperature, pressure, and composition, `Transit` calculates the thinnest HWHM ($HWHM_{min}$), then `Transit` selects the largest integer divisor of

'wnosamp' ($d_{\rm max}$) that resolves HWHM$_{\rm min}$:

$$\Delta\nu_{\rm d} = d_{\rm max}\,\Delta\nu_{\rm f} \leq \frac{\rm HWHM_{min}}{\rm resolv}. \quad (1)$$

6.2.1.2 Atmospheric-Layer Sampling

By default, Transit uses the atmospheric file's layer sampling. If the user sets the '**raddelt**' argument, the layers will be resampled to an equi-spaced radius sampling. The radius boundaries can also be redefined with the '**radhigh**' and '**radlow**' arguments.

6.2.1.3 Extinction-Coefficient Calculation

Transit has an option to calculate the extinction coefficient on the spot (line-by-line calculation) for each layer, or interpolate from a pre-calculated table of opacities.

By setting the '**opacityfile**' argument, the user chooses to interpolate from the specified opacity table. If the file does not exist, Transit will compute (and save) the extinction-coefficient table over a grid of wavenumber, pressure, temperature, and species arrays. The wavenumber is taken from the coarse wavenumber sampling. The pressure array is taken from the atmospheric layers. The list of species will be taken from the TLI file. The temperature array will be computed as a linear sample from **'tlow'** to **'thigh'** with sampling interval **'tempdelt'**. If the opacity file exists, Transit will check that the table's wavenumber, pressure, and species list match those specified by the command-line arguments.

Typical runtimes (for a 100-layers atmosphere, 5 million lines, from 3 to 11 μm) to calculate a spectrum with an existing opacity file (interpolation), on the spot (line-by-line calculation), or to generate the opacity file requires a few seconds, minutes, and hours, respectively.

6.2.1.4 Line-by-Line Calculation

The line-by-line calculation of the extinction coefficient (see the theory document) will only consider the contribution from the lines which strength is larger than $S_{\rm max}\times$ **ethresh**, with $S_{\rm max}$ the maximum line-strength in a given layer.

6.2.1.5 Voigt-Profile Calculation

The Voigt profiles used in the line-by-line extinction-coefficient calculation are pre-calculated in a 2D table for a range of Doppler and Lorentz widths. The Doppler range is a log-spaced sample of '**ndop**' widths from '**dmin**' to '**dmax**'. Likewise, the Lorentz range is a log-spaced sample of '**nlor**' widths from '**lmin**' to '**lmax**'. The 'nwidth' argument indicates how far from the central wavenumber (in number of profile half-widths) to calculate the Voigt profile.

6.2.1.6 Cloud Opacity

Transit allows for a basic gray-opacity (cloud) layer. The two values of 'cloudrad' define a the top and bottom radii of a layer where the opacity linearly increasing from zero (at **radup**) up to **cloudext** at **raddown**. Below this level, the opacity remains constant at **cloudext**.

6.2.2 Configuration File

A configuration file presents an alternative to the command-line arguments to specify the input variables. The following example (**transit/run/config_sample.cfg**) shows the format of a

basic transit configuration file, which is further explained below:

```
# Transit Configuration File Example:
# Comment (#) and empty lines are allowed.
# To set an argument, write the argument name, followed by the
#  argument value (white-space separated).  No need for the 'equal'
#  sign, nor quotes for string values.
# For the full list of arguments see Transit User Guide or type: transit -h

# ::::::::::  Input files  ::::::::::::::::::::::::::::::::::::::::::
# Path to atmospheric info file:
atm    /home/.../HD209458b_atm.tea
# Path to transit line information (TLI) file:
linedb /home/.../HITRAN_CH4.tli
# Path to collision induced absorption (CIA) file:
cia    /home/.../h2h2.dat

# ::::::::::  Spectrum sampling  ::::::::::::::::::::::::::::::::::::
# Lowest wavelength boundary (also can be set as the wavenumber
#   highest boundary with wnhigh):
wllow   2.8
# Highest wavelength boundary (also can be set as the wavenumber
#   lowest boundary with wnlow):
wlhigh 11.0
# Conversion factor from wavelength units to cm (default: 1e-4, microns):
wlfct  1e-4
# Wavenumber sampling for plotting the final output (default: 1.0)
wndelt  1.0
# Wavenumber over-sampling factor for internal calculations, see the User Guide
#   (default: 2160)
wnosamp 2160
# Conversion factor from wavenumber units to cm-1 (default: 1.0):
wnfct   1.0

# ::::::::::  Geometry  ::::::::::::::::::::::::::::::::::::::::::::::
# Eclipse or transit (default: eclipse)
solution eclipse
# Planetary grid for intensity calculation in degrees
#   (default: 0 20 40 60 80)
raygrid 0 20 40 60 80

# ::::::::::  Optical Depth ::::::::::::::::::::::::::::::::::::::::::
# Maximum optical depth (default: 20)
toomuch 10
# Opacity threshold, see the User Guide (default: 1e-6)
ethresh 1e-6
```

```
# ::::::::::  Broadening Function Calculation  ::::::::::::::::::
# Number of HWHM left and right from the Voight profile centre
#    (default: 20)
nwidth 20

# ::::::::::  Opacity Grid Calculation ::::::::::::::::::::::::::::
# Lowest temperature boundary in K (default: 500)
tlow        500
# Highest temperature boundary in K (default: 3000)
thigh      3000
# Temperature sampling in K (default: 100)
tempdelt    100
# Path to the opacity file
opacityfile  ./opacity_CH4.dat

# ::::::::::  Verbosity Level  ::::::::::::::::::::::::::::::::::::
# Level of on-screen verbosity from 1-20 (default: 2)
verb 11

# ::::::::::  Output Files  ::::::::::::::::::::::::::::::::::::::::
# Wavelength vs. radius where the optical depth reaches toomuch
outtoomuch ./eclipse_toomuch.dat
# Various sampling information
outsample  ./eclipse_sampling.dat
# Final output spectrum, wavelength vs. flux
output     ./eclipse_spectrum.dat
```

`Transit`'s configuration file has its own format. Comment lines start with the # character. When setting an argument–value pair, do not include the 'equal' sign in between. String values do not need the quotation marks.

6.2.3 Atmospheric File

The atmospheric file is a plain text file that determines the species present in the atmosphere and the physical properties of the atmospheric layers (pressure, radius, temperature, and mole mixing ratio of the species) assuming a 1D plane-parallel model. See below (and `transit/examples/example01/samp` **(FINDME: add file to repo)** for an atmospheric file sample:

```
# This is a sample atmospheric file for transit.
# HD209458b

# Values units: radius (km), pressure (bar), temperature (K),
#               abundances (unitless).
ur 1e5
up 1e6
q number
```

```
#SPECIES
H He C N O H2 CO CO2 CH4 H2O NH3 C2H2 C2H4

#TEADATA
# Radius     Pressure     Temp      H               C               ...   C2H4
 93102.145   1.0000e+02   1887      9.9917414e-01   2.6893119e-04   ...   9.5068e-09
 96092.954   1.7783e+00   1839.85   9.9917414e-01   2.6893119e-04   ...   2.0976e-08
 99008.753   3.1623e-02   1791.37   9.9917414e-01   2.6893119e-04   ...   2.2020e-08
       ...   ...          ...       ...             ...             ...   ...
101851.331   5.6234e-04   1744.45   9.9917414e-01   2.6893119e-04   ...   2.3203e-08
104638.705   1.0000e-05   1701.75   9.9917414e-01   2.6893119e-04   ...   2.4418e-08
```

The atmospheric file follows the following format: Blank and comment lines (starting with the '#' character) are allowed and ignored by transit. The 'ur' and 'up' keywords set the radius and pressure units (conversion factors to CGS), respectively. For example, if the radii are given in km ($= 10^5$ cm), set 'ur 1e5'. The 'q' keyword indicate if the abundances are given by mass or by number. The line after '#SPECIES' lists the species in the atmosphere. The '#TEADATA' line indicates where the atmospheric data per layer starts. The file contains one layer's information per line (in descending pressure order). Each line contains the radius, pressure, temperature, and mole mixing ratio for each layer. The file can be produced using the BART routines (see github.com/joeharr4/BART and documentation therein).

6.2.4 Transit Line Information (TLI) File

The TLI file is a binary file that provides the species' line-transition information (central wavelength, lower-state energy, and weighted oscillator strength), mass, isotopic ratio, and a tabulated partition-function array as function of temperature for each one. The TLI file is the output of the `Pylineread` module.

6.2.5 CIA File

The CIA file provides tabulated collision-induced absorption data for Transit. CIA data can be obtained from Aleksandra Borysow's webpage (astro.ku.dk/∼aborysow/programs/index.html). See below (and `transit/inputs/CIA_H2H2_400-7000K.dat`) for a CIA file sample:

```
# CIA Header for H2-H2:
i H2 H2
t          1000       2000       3000       4000       5000       6000       7000

# Wavenumber in cm-1, CIA coefficients in cm-1 amagat-2:
   20.00   0.467E-08 0.321E-08 0.283E-08 0.291E-08 0.319E-08 0.351E-08 0.386E-08
   40.00   0.184E-07 0.127E-07 0.113E-07 0.116E-07 0.127E-07 0.140E-07 0.154E-07
   60.00   0.402E-07 0.285E-07 0.253E-07 0.261E-07 0.286E-07 0.315E-07 0.347E-07
     ...
20000.00   0.269E-14 0.567E-12 0.597E-11 0.205E-10 0.382E-10 0.103E-09 0.144E-09
```

A valid CIA file is a plain text file that contains a header and a main body. Comment lines (starting with the '#' character) and blank lines are allowed and ignored by `Transit`. The header

must contain two keyword arguments. A line starting with the '`i`' keyword contains the name of the species separated by blank spaces (names should match those from the atmospheric file). A line starting with the '`t`' keyword contains the list of temperatures (blank-space separated, in Kelvin degrees) sampled in the CIA file.

The body of the CIA file contains a tabulated list with the absorption coefficients (in units of cm^{-1} $amagat^{-2}$) evaluated as a function of wavenumber (in units of cm^{-1}) and temperature. Each line contains the wavenumber (first column) and the coefficient corresponding to the temperatures specified in the header (values blank-space separated).

6.2.6 Opacity File

The opacity file is a binary file that contains a tabulated grid of extinction coefficients as function of wavenumber, for each species with a line list, and for a range of pressures and temperatures. If specified, `Transit` will use this table to interpolate the extinction coefficient at the temperatures given by the atmospheric profile. The opacity- and atmospheric-file's pressure arrays must coincide. Likewise, the `transit` and the opacity-file's wavenumber arrays must also coincide. Similarly, all species in the TLI file must be contained in the opacity file. See Section 6.2.1.3.

6.2.7 Molecules Data File

The molecules data file (`transit/develop/inputs/molecules.dat`) defines a universal ID, the mass, and the (collision) diameter for the species in the atmosphere. The layout of the `molecules.dat` file is given below:

```
# Molecular info:
# Radii sources:
# 01  http://chem.chem.rochester.edu/~nvd/molecularsieves.html
# 02  Mateucci et al. (2006)
# 03  http://www.webelements.com/compounds (from density and mass calculation)
# 04  Wolfram Alpha + Wikipedia
# 05  http://www.chemspider.com/Molecular-Formula/C2H2
# Mass source: http://www.webqc.org/mmcalc.php

# ID    Molecule  Mass         Diameter  Diameter   Long
#       Name      g/mol        Angstrom  source     name
 101    H2O       18.01528     3.2       01         Water
 102    CH4       16.0425      4.0       01         Methane
 103    CO        28.0101      2.8       01         Carbon Monoxide
 104    CO2       44.0095      2.8       01         Carbon Dioxide
 105    H2         2.01588     2.89      02         Molecular Helium
 106    NH3       17.03052     3.6       01         Ammonia
 107    TiO       63.8664      3.45      03         Titanium Monoxide
 108    VO        66.94090     3.32      03         Vanadium Monoxide
 109    O2        31.99880     3.46      02         Molecular Oxygen
 110    N2        28.01340     3.64      02         Molecular Nitrogen
 111    C2H2      26.0373      5.26      05         Acetylene
 112    C2H4      28.0532      5.69      05         Ethylene
```

```
 113    HCN        27.02534     5.0        05         Hydrogen Cyanide
   1    H           1.007940    2.4        01         Hydrogen
   2    He          4.0026020   2.0        01         Helium
   6    C          12.0107      1.7        04         Carbon
   7    N          14.0067      1.55       04         Nitrogen
   8    O          15.9994      1.52       04         Oxygen
  11    Na         22.98976928  3.72       04         Sodium
  19    K          39.09830     4.54       04         Potassium
```

7 Program Outputs

7.1 Pylineread Output

The output of the `Pylineread` module is a TLI file, described in Section 6.2.4.

7.2 Transit Output

The `Transit` module produces up to five output files: the opacity, the flux spectrum, the intensity spectrum, the sampling information, and the max optical-depth files.

7.2.1 Opacity File

See Section 6.2.6.

7.2.2 Flux Spectrum File

This file contains the calculated modulation (transit geometry) or hemisphere-integrated emission (eclipse geometry) spectrum. The first column shows the wavelength (in μm) and the second column the spectrum (unitless for transit, erg $s^{-1}cm^{-1}$ for eclipse). The following example shows this file's layout for an eclipse run:

```
#wvl [um]       Flux [erg/s/cm]
11              50317.47
10.9879133      44824.9485
10.97585312     41507.0681
    ...             ...
    ...             ...
```

(FINDME: change -Flux to something more reasonable).

7.2.3 Intensity Spectrum File

This file contains the calculated intensity spectrum at each incident angle (for eclipse geometry). The first column has the wavelength (in μm) and the subsequent columns the planetary emission intensity (in erg $s^{-1}cm^{-1}sr^{-1}$) for each of the angles specified by the '**`raygrid`**' argument. The following example shows the layout of this file:

```
#wvl [um]  I[ 0.0 deg]  I[30.0 deg]  I[60.0 deg]  I[80.0 deg]  [erg/s/cm/sr]
11         16020.5901   16020.4135   16019.3219   16016.2266
10.9879    14731.1128   14682.3188   14512.7771   14116.6959
    ...        ...          ...          ...          ...
    ...        ...          ...          ...          ...
```

7.2.4 Sampling Information File

This file stores the wavenumber, radius, and impact parameter sampling information. Each sampling shows the conversion factor to cgs units, the initial and final sampling values, the spacing interval between samples, the number of elements, the wavenumber sampling's oversample factor, and the radius sampling's list of values for radius. The following example shows the layout of this file:

```
############################
   Wavenumber   Sampling
----------------------------
Factor to cgs units: 1
Initial value: 909.091
Final value: 3571.43
Spacing: 1
Oversample: 1
Number of elements: 2663
############################
   Radius       Sampling
----------------------------
Factor to cgs units: 100000
Initial value: 123453
Final value: 140707
Spacing: 100
Number of elements: 173
Values:     123452.85    123552.85
              ...          ...
              ...          ...
############################
   Impact parameter Sampling
----------------------------
Factor to cgs units: 100000
Initial value: 140653
Final value: 123453
Spacing: -100
Oversample: 1
Number of elements: 173
Values:     140652.85    140552.85
              ...          ...
              ...          ...
```

7.2.5 Max Optical-Depth File

This file contains the atmospheric layer's radius where the optical depth reached `toomuch`, as a function of wavelength. The following example shows the layout of this file:

```
#Wavenumber (cm-1)  Radius at max. calculated depth (cm)
909.0909091         13555285300
910.0909091         13095285300
     ...                 ...
     ...                 ...
```

(FINDME: change wavenumber to wavelength)

8 Running Transit

8.1 Running Pylineread

We strongly suggest the use a configuration file to specify the command-line arguments. In that way, the user has a record of the arguments used to create the TLI files. The example provided in `transit/pylineread/examples/` is further described in Section 6.1.2.

`Pylineread` can be run from any folder, provided the user gives the path from the working folder to the executable. We recommend to create a folder exclusively to execute `Pylineread` and hold its results. For example, to run `Pylineread` from a folder located in `transit/pylineread/TLI/` using a configuration file, cd into `transit/pylineread/` and execute from the shell:

```
mkdir TLI
cd TLI/
cp ../examples/pyline_example.cfg pyline_run_01.cfg
```

(Edit the configuration file arguments following the format from Section 6.1.2)

```
../src/pylineread.py -c pyline_run_01.cfg
```

8.2 Running Transit

`Transit` runs in the same way as `Pylineread`. Thus, the same recommendations from Section 8.1 apply. Although `Transit` has many command-line arguments, in most cases, most of them can be left as defaults. A minimal example is given in `transit/run/transit_example.cfg`. For example, to run `Transit` using a configuration file from the `transit/run/` folder, cd into that folder and execute from the shell:

```
cp config_sample.cfg transit_run_01.cfg
```

(Edit the configuration file arguments following the format from Section 6.2.2)

```
../transit/transit -c transit_run_01.cfg
```

9 Code Organization

(FINDME: TBD)

9.1 TLI File Format

The data listed in a TLI file, in order as they appear, are listed below:

1. Magic number (endianness)
2. TLI version (short)
3. Line reader version (short)
4. Line reader revision version (short)
5. Initial and final wavelength (doubles)
6. Number of databases (short)
7. For each database:
    - Length (short) and name (string) of the database
    - Length (short) and name (string) of the molecule
    - Number of temperature samples (short)
    - Number of isotopes (short)
    - Temperature array (doubles)
    - For each isotope:
        - Length (short) and name (string) of the isotope
        - Mass of the isotope (double)
        - Isotopic ratio (double)
        - Partition function at each temperature (doubles)
8. Line transition information:
    - Number of transitions (int)
    - Transition's wavelength array (doubles)
    - Transition's isotope ID array (shorts)
    - Transition's lower-state energy array (doubles)
    - Transition's oscillator strength (gf) array (doubles)

9.1.1 Opacity File Format

The opacity file is a binary file that contains a tabulated grid of extinction coefficients as function of wavenumber, for each species with a line list, and for a range of pressures and temperatures.

The data stored in the opacity file are:

- Number of species (integer)
- Number of temperature samples (integer)
- Number of pressure-layer samples (integer)

- Number of wavenumber samples (integer)
- Species ID (array of integers)
- Temperature sample (array of floats)
- Layer's pressure (array of floats)
- Wavenumber array (array of floats)
- Extinction-coefficient (array of floats)

10 Routines

(FINDME: TBD)

10.1 Pylineread Module

(FINDME: TBD)

pylineread routines (`transit/pylineread/src/`):

`pylineread.py:`
(FINDME: Explain me). (*)

`driver.py:`
(FINDME: Explain me).

`db_pands.py:`
(FINDME: Explain me).

`db_voplez.py:`
(FINDME: Explain me).

`db_hitran.py:`
(FINDME: Explain me).

`db_tioschwenke.py:`
(FINDME: Explain me).

`utils.py:`
(FINDME: Explain me).

`constants.py:`
(FINDME: Explain me).

Fortran routines (`transit/pylineread/src/fortran/`):

`BART_BD_ISO_82_TO_85.for:`
(FINDME: Explain me).

`BD_ISO_2011.for:`
(FINDME: Explain me).

`BD_MOL_2011.for:`
(FINDME: Explain me).

`ISOTOPS.CMN:`
(FINDME: Explain me).

`MOLEC.CMN:`

(FINDME: Explain me).

README_TIPS:
(FINDME: Explain me).

SPECIES_2011.CMN:
(FINDME: Explain me).

TIPS_BART.for:
(FINDME: Explain me).

Asterisk (*) indicates files that must be executable.

10.2 Transit Module

(FINDME: TBD)

Transit routines (`transit/transit/src/`, in order of execution):

transit.c:
Main routine that calls the rest of the subroutines.

argum.c:
Read and process the user input parameters.

makesample.c:
Create the spectrum, layer, and temperature arrays.

readatm.c:
Read and process the input atmospheric file.

readlineinfo.c:
Read and process the input TLI (transition-lines information) file.

cia.c:
Read and process the collision-induced-absorption files.

idxrefraction.c:
Calculate the index of refraction of the ray-light path.

opacity.c:
Compute the line-broadening profiles and tabulated grid of opacities.

extinction.c:
Compute the extinction coefficient at a given layer.

tau.c:
Calculate the extinction coefficient at each layer for transit geometry.

slantpath.c:
Calculate the transit ray-path and integrated the extinction to calculate the optical depth.

observable.c:
Integrate the optical depth compute the transit modulation.

eclipse.c:
Compute the extinction coefficient and optical depth. Integrate the optical depth to calculate the planet intensity. Integrate the intensity to obtain the planetary emission spectrum (flux).

transitstd.c:

(FINDME: Explain me).

`geometry.c:`
(FINDME: Explain me).

`MPItransit.c:`
Main transit routine for use with the BART project.

10.3 PU Module

pu routines (`/transit/pu/src/`):

`iomisc.c:`
(FINDME: Explain me).

`numerical.c:`
(FINDME: Explain me).

`procopt.c:`
(FINDME: Explain me).

`voigt.c:`
(FINDME: Explain me).

`messagep.c:`
(FINDME: Explain me).

`sampling.c:`
(FINDME: Explain me).

`xmalloc.c:`
(FINDME: Explain me).

11 Be Kind

Please cite these papers if you found this package useful for your research:

- Cubillos et al. (2015) (in preparation).
- Blecic et al. (2015) (in preparation).
- Harrington et al. (2015) (in preparation).

Thanks!

12 Reproducible Research

(FINDME: TBD)
How to comply with reproducible research.
What to do if you do not edit the code. What to do if you edit the code.

13 Further Reading

(FINDME: TBD, do we need this section?)

References

Fischer, J., Gamache, R. R., Goldman, A., Rothman, L. S., & Perrin, A. 2003, J. Quant. Spec. Radiat. Transf., 82, 401, ADS

Laraia, A. L., Gamache, R. R., Lamouroux, J., Gordon, I. E., & Rothman, L. S. 2011, Icarus, 215, 391, ADS

Lide, D. R. 2008, CRC Handbook of chemistry and physics: a ready-reference book of chemical and physical data, ADS

Šimečková, M., Jacquemart, D., Rothman, L. S., Gamache, R. R., & Goldman, A. 2006, J. Quant. Spec. Radiat. Transf., 98, 130, ADS