Documentation

Levin Hennicker

May 22, 2023

Contents

1 Introduction

The aim of this code library is to perform radiative transfer calculations for a range of applications, focussing however on the spectral synthesis of 3D hot-star winds. The main features consist of performing detailed radiative transfer calculations accounting for highly supersonic velocity fields and (almost) arbitrary 3D structures. We have developed several main programs to be used for different situations, summarized as follows:

line3D: A code module to calculate synthetic line profiles for a single star within a global *star-in-a-box* setup (see Sect. 3.1).

BOSS-3D: A code module to calculate synthetic line profiles for binary systems (see Sect. 3.3).

cont3Dslab: A code module to calculate the continuum radiation for a single star within a local *star-in-a-box* setup, that might be used for coupling in radiation-hydrodynamic simulations (see Sect. 3.2).

Each of these packages will be described in the corresponding sections in more detail.

1.1 Radiation hydrodynamics

ToDo

1.2 Radiative transfer

To calculate the radiative transfer, we consider the time-independent equation of radiative transfer,

$$\boldsymbol{n}\boldsymbol{\nabla} I_\nu = \eta_\nu - \chi_\nu I_\nu = \chi_\nu (S_\nu - I_\nu)\,, \tag{1}$$

with I_ν the specific intensity, η_ν the emissivity, χ_ν the opacity, and $S_\nu = \eta_\nu / \chi_\nu$ the source function. Further, we define the angular moments of the specific intensity:

$$J_\nu = \frac{1}{4\pi}\int I_\nu \mathrm{d}\Omega = \frac{c}{4\pi} E_\nu\,, \tag{2}$$

$$\boldsymbol{H}_\nu = \frac{1}{4\pi}\int I_\nu \boldsymbol{n} \mathrm{d}\Omega = \frac{1}{4\pi} \boldsymbol{F}_\nu\,, \tag{3}$$

$$\boldsymbol{K}_\nu = \frac{1}{4\pi}\int \underbrace{\boldsymbol{n} I_\nu \boldsymbol{n}}_{\text{dyadic product}} \mathrm{d}\Omega = \frac{c}{4\pi} \boldsymbol{P}_\nu\,, \tag{4}$$

with J_ν the mean intensity, $\boldsymbol{H}_\nu$ the Eddington flux, and K_ν simply the second moment without a specific name. The mean intensity, Eddington flux, and second moment are trivially related to the radiation energy density E_ν, the radiation flux $\boldsymbol{F}_\nu$, and the radiation pressure tensor $\boldsymbol{P}_\nu$.

For all code modules, we are typically considering the radiation quantities I_ν, J_ν, $\boldsymbol{H}_\nu$, $\boldsymbol{K}_\nu$, and not the corresponding 'physical' quantities. In general there are three operating modes for calculating these quantities:

(i) Since the source function in the equation of radiative transfer (EQRT) can in principle depend on the radiation field, Eq. (1) becomes an integro-differential equation. In this case, an iteration scheme is required (main code *line3D/sc3d.eo* and *cont3Dslab/sc3d.eo* for star-in-a-box and box-in-a-star simulations, respectively) based on a non-local accelerated Λ iteration (ALI).

(ii) For known source functions and opacities (e.g., approximated in LTE or pre-calculated in step (i)), we can solve the radiative transfer in a *pz*-type geometry to obtain surface brightnesses or emergent flux profiles (main codes *line3D/modelspec.eo* and *line3D/spec.eo*).

(iii) If we are dealing with binary systems, we rely purely on semi-analytical models thus far (e.g., opacities and source functions in LTE), and solve the radiative transfer in a *pz*-type of geometry (main codes *line3D/modelspec_vbin.eo* and *line3D/spec_vbin.eo*).

1.3 Philosophy

All developed sub-programs are meant to be – at least in principle – a sort of stand-alone packages, that can be used completely independent of each other. Indeed, we consider the full radiative transfer problem as a three-step process:

1. Firstly, we need to create a discretized model of the physical state of the gas (i.e., density ρ, gas temperature T_{gas}, velocity field $\boldsymbol{v}$). In order that our radiative-transfer routines can communicate with such a model, we have developed a user interface to either transform input data from a given hydrodynamic simulation or to set up a semi-analytical model. This step is performed in the code *model.eo* with corresponding source code to be found in *line3D/src_model* and *cont3Dslab/src_model*.

2. Secondly, we need to calculate opacities and source functions. We can follow two branches here:
   - For resonance lines within a two-level-approximation and/or a two-component continuum source consisting of thermal and scattering terms, we can calculate continuum and line source functions consistently with the radiation field. This is performed by the code *sc3d.eo*. Since the iteration scheme is computationally very expensive, the source function will be calculated on a relatively low-resolution grid, and then interpolated back onto the original model within the code *modelspec.eo*. The corresponding source codes can be found in *line3D/src_sc3d*, *line3D/src_modelspec*, and *cont3Dslab/src_sc3d*.
   - Alternatively, we can directly use the code *modelspec.eo* to calculate source functions and opacities from semi-analytical calculations (e.g., assuming LTE occupation numbers, see *line3D/src_modelspec*).
3. Finally, the surface brightness or emergent flux profiles are calculated by solving the radiative transfer in a *pz*-type geometry using the code *spec.eo*, with input given from the previous step 2. The corresponding source code can be found in *line3D/src_spec* and *cont3Dslab/src_surfb*.

We emphasize that the binary version (extension *_vbin*) consists only of steps (2) and (3), since we haven't implemented an ALI scheme for binary systems yet.

2 Installation

2.1 Requirements

The code requires the following packages:

FORTRAN compiler: Either the gfortran (version 8+) or ifort compiler is required. Depending which compiler is used, one needs to adapt the source code since the INQUIRE function works differently for both compilers:
gfortran: inquire(file=trim(directory)//'/.', exist=my_boolean)
ifort: inquire(directory=trim(directory), exist=my_boolean)

HDF5: The HDF5 library is required (version 1.10.5 or higher). This library needs to be compiled with the same compiler as used for the main programs (i.e., gfortran or ifort). See Sect. 2.4 on how to install HDF5 on your system

PYTHON, IDL, GDL The code comes with an IDL/GDL or PYTHON library for reading and plotting all output files.

2.2 Getting the code

You can get the code from github:

- Develpopment version of Global star-in-a-box simulations:
```
git clone https://github.com/IvS-KULeuven/line3D_dev
```
- Global star-in-a-box simulations:
```
git clone https://github.com/levin-h/line3D
```
- Local box-in-a-star simulations:
```
git clone https://github.com/levin-h/cont3Dslab
```

2.3 Quick start

2.4 Installing HDF5

I recommend to create a local build for your HDF5 libraries. To this end, please follow the following steps (here for version 1.10.6)

UNIX systems

1. Download the package **hdf5-1.10.6.tar.gz**, unzip it, and change to the corresponding folder:
```
tar -zxvf hdf5-1.10.6.tar.gz
cd hdf5-1.10.6
```

2. Export some required environment variables:

gfortran	**ifort**
`export FC=gfortran`	`export FC=ifort`
`export CC=gcc`	export CC=`icc`
`export F9X=gfortran`	`export F9X=ifort`
`export CXX=g++`	`export CXX=icpc`

3. Configure your installation with a local path where you want to install the package (e.g., `.../hdf5_lib`):
   `./configure --prefix=.../hdf5_lib --enable-fortran` (and if required `--enable-cxx`)

4. Installation
   `make`
   watch for fatal errors
   `make check`
   verify that all tests return a 'pass'
   `make install`
   all done

5. Include the library path in the Makefile of the main code (e.g., `line3D/Makefile`)

6. Add the library path to your LD environment variable, and define a new environment variable that will be used in the Makefiles:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<.../hdf5_lib/lib>
export LIB_HDF5=<.../hdf5_lib>
export COMPILER=<.../gfortran>
export DIR_OPAL=<.../opal_tables>
export DIR_LTE=<.../lte_tables>
```

   This can be either added to `.bashrc` / `.zshrc`, or text file can be created with the list of variables. To do this we create `.env` file with folowing contents:

```
HDF5_PATH=<.../hdf5_lib/lib>
COMPILER=<.../gfortran>
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HDF5_PATH
LIB_HDF5=<.../hdf5_lib>
DIR_OPAL=<.../opal_tables>
DIR_LTE=<.../lte_tables>
```

   Variables from such file can simply be exported by calling

```
export $(grep -v "^#" .env | xargs -d '\n')
```

   for GNU system or

```
export $(grep -v "^#" .env | xargs -0)
```

   BSD systems.

MAC On the MAC, the installation is essentially performed the same way. For the gfortran compiler, you might need

1. Install the CommandLineTools
   `xcode-select --install`

2. Install homebrew https://docs.brew.sh/Installation:

```
mkdir homebrew && curl -L https://github.com/Homebrew/brew/tarball/master \
| tar xz --strip 1 -C homebrew

eval "$(homebrew/bin/brew shellenv)"
brew update --force --quiet
chmod -R go-w "$(brew --prefix)/share/zsh"
export PATH=\$HOME/homebrew/bin:\$PATH
```

3. Install gfortran:
   ```
   brew install gcc
   ```

4. Install HDF5 as for UNIX systems. When updating the Makefile of the main code (e.g., `line3D/Makefile`), replace all *.so libraries with the MAC *.dylib extension (depending on the version of the Makefile, not required anymore).
   NOTE. For new MAC on M1 chip gfortran provided by Xcode and by Homebrow are incompatible. As such HDF5 should be compiled with the same fortran compiler as the main code will be. Best will be si install HDF5 library from Homebrew
   ```
   brew install hdf5
   ```

2.5 Importing the plotting routines

All plotting routines are stored in the directory *line3D/plotFILES* or *cont3Dslab/plotFILES*, with corresponding libraries named *levpy*, *lib_gdl* and *lib_idl* for PYTHON, GDL, and IDL, respectively.

GDL/IDL The required libraries can be simply read in by typing in your IDL/GDL terminal:
@idl_lib/startup.pro
@gdl_lib/startup.pro

We emphasize that some of the routines might be outdated (for instance, variable names stored within the individual *.h5 files might have changed). It should be straight forward though to adapt the corresponding reading routines (in *idl_lib/proLEV/getall*).

PYTHON The following python packages are required:
h5py, imageio, matplotlib, numpy, os, scipy, sys

3 Main code modules

3.1 line3D

This folder contains code modules for running global (star-in-a-box) radiative transfer simulations. There are essentially four different code modules further described in the following. Each code module requires a specific input file organized by namelists.

model.eo Prepares the data to be used for the actual radiative transfer calculations, by transforming any input data or calculating semi-analytic models. You can simply add new models in src_model. The corresponding namelist file is described in Table 9. Essentially, we set up the state of the gas in 1d, 2d, or 3d, described by the density ρ, the velocity field $\boldsymbol{v}$, the gas and radiation temperatures, $T_{\rm gas}$ and $T_{\rm rad}$, the thermal velocities (becomes obsolete at some point), $v_{\rm th}$, and the thermalization parameter $\epsilon_{\rm C}$. Currently hardcoded, the model will be saved in inputFILES/modelXd.h5.

sc3d.eo Performs the radiative transfer for the continuum and/or line transition with certain opacity laws using an iterative ALI scheme. The corresponding namelist file is described in Table 9.

modelspec.eo Prepares the data to be used for the line-profile calculations of single stars. Here, we can read in the source functions and opacities calculated by the *sc3d.eo* program, or implement other semi-analytic models. The corresponding namelist file is described in Table 2.

spec.eo Calculates line profiles for a specific input file. The corresponding namelist files is described in Table 3.

3.1.1 Program model.eo

All source files for this program are stored in the directory *src_model*, and the corresponding namelist file is summarized in Table 9. Within this namelist, there are many input parameters that are actually not required. For consistency reasons and to avoid potential error sources, we decided to use the same namelist file also for the program *sc3d.eo*.

To register a new model, we recommend to follow the following steps:

1. In *src_model/model.f90*, add a new model identifier as a case for the variable input_mod. This identifier should be used also in the namelist file to call this particular model. The subroutine to create the model still needs to be developed by the user, e.g., *calc_my_model*, and needs to be called within the case of the new model identifier. Further, depending on the dimensionality of the new model, we need to save it as an h5 file by calling the (already existing) subroutines *output_mod*d*.

2. In *src_model/model*d.f90*, we create our new subroutine *calc_my_model*. Depending on the dimension of the new model, different global variables have to be set (see *src_model/output_model.f90* for more details). For a 3D model in spherical coordinates (r, Θ, Φ), we would require the following:

   nr_modext describes the number of radial grid points for our model.

   ntheta_modext describes the number of Θ grid points for our model.

   nphi_modext describes the number of Φ grid points for our model.

   r_modext3d describes the radial grid (array of length nr_modext) in cgs.

   theta_modext3d describes the Θ grid (array of length ntheta_modext) from $[0, \pi]$.

   phi_modext3d describes the Φ grid (array of length nphi_modext) from $[0, 2\pi]$.

   velr_modext3d describes the radial velocity component (array of length nr_modext, ntheta_modext, nphi_modext) in cgs.

   velth_modext3d describes the Θ velocity component (array of length nr_modext, ntheta_modext, nphi_modext) in cgs.

   velphi_modext3d describes the Φ velocity component (array of length nr_modext, ntheta_modext, nphi_modext) in cgs.

   rho_modext3d describes the density (array of length nr_modext, ntheta_modext, nphi_modext) in cgs.

   t_modext3d describes the gas temperature (array of length nr_modext, ntheta_modext, nphi_modext) in cgs.

   trad_modext3d describes the radiation temperature (array of length nr_modext, ntheta_modext, nphi_modext) in cgs. Often used only as a dummy array.

   vth_modext3d describes the thermal velocity (array of length nr_modext, ntheta_modext, nphi_modext) in cgs. Often used only as a dummy array.

   eps_cont_modext3d describes the thermalization parameter (array of length nr_modext, ntheta_modext, nphi_modext).

To plot the resulting model, you can use the programs *plotFILES/model*d.py* or *plotFILES/model*d.pro* for PYTHON or IDL/GDL, respectively.

Table 1: Input namelist for the programs *model.eo* and *sc3d.eo*. Some of the inputs are meanwhile obsolete. We use one indat file for both programs to avoid inconsistency of the data used within *model.eo* and *sc3d.eo*. If only running the *model.eo* many of the paramneters are not required and should be assigned with an arbitrary value.

Example	Data type	Description
&input_options		Options for the models
model_dir = 'inputFILES'	string	Directory of the model that will be read in
output_file = 'output_model00.h5'	string	All calculations stored in output_file (.h5 extension to be included)
input_mod = 12	integer	Identifier of the model to be calculated; only required for model.eo (where the hydro model is specified)
input_mod_dim = 3	integer	Dimension of input model; input_mod_dim $\in [1, 2, 3]$
spatial_grid1d = 5	integer	Identifier to calculate a 1D radial grid from a beta-velocity law. Depending ong the option spatial_grid3d, this input is obsolete. spatial_grid1d=0 if equidistant radial grid is used (subroutine grid1d_r_equi) spatial_grid1d=1 if equidistant velocity grid is used (subroutine grid1d_vel_equi) spatial_grid1d=2 if equidistant tau_thomson grid is used (subroutine grid1d_tau_equi) spatial_grid1d=3 if equidistant log(tau_thomson) grid is used (subroutine grid1d_tau_log) spatial_grid1d=4 if combination is used (see subroutine grid1d_final for details) spatial_grid1d=5 if combination is used (see subroutine grid1d_final_2 for details) spatial_grid1d=6 if grid is calucalted equidistant in log-space (subroutine grid1d_r_log)
spatial_grid3d = 2	integer	Identifier to calculate the 3D Cartesian grid. spatial_grid3d=0 if 3d grid is calculated from 1d grid with equidistant core points spatial_grid3d=1 if 3d grid is calculated from a mean-value approach (minimizing distance of subsequent coordinates from 1d-grid)

		spatial_grid3d=2 if 3d grid is calculated from a mean-value approach (minimizing distance of subsequent coordinates from original input-grid) spatial_grid3d=3 if 3d grid is calculated completely equidistant spatial_grid3d=4 if 3d grid is calculated from a 1d radial grid and setting up angular grid equidistantly spatial_grid3d=5 if 3d grid is calculated from a 3d spherical grid (optimized)
opt_opac = 0	integer	Identifier to decide on the continuum opacity model opt_opac=0 if Thomson opacities opt_opac=1 if OPAL opacities
opt_opal = 0	integer	Identifier to decide on the line opacity model opt_opal=0 if line-strength parameter opt_opal=1 if Hamann (1980) parameterization
opt_angint_method = 9	integer	Identifier to decide on the angular-integration technique to be used opt_angint_method=0 if angular integration is used with trapezoidal rule (nodes equidistant in θ and ϕ) opt_angint_method=1 if angular integration is used with trapezoidal rule (nodes from Lobel & Blomme (2008)) opt_angint_method=2 if angular integration is used with simpsons rule (nodes equidistant in θ and ϕ, note: μ-grid and ϕ-grid will be made equidistant for three subsequent points) opt_angint_method=3 if angular integration is used with simpson rule corrected for the error from a grid with half resolution (also known as boole's rule) opt_angint_method=4 if angular integration is used with cubic splines (catmull-rom-spline, nodes equidistant in θ and ϕ) opt_angint_method=5 if angular integration is used with gauss-legendre-integration (for each octant) opt_angint_method=6 if angular integration is used with gauss-chebyshev-integration (for each octant) opt_angint_method=7 if angular integration is used with triangulation (linear integrals) opt_angint_method=8 if angular integration is used with triangulation ('pseudo'-gauss integrals per triangle) opt_angint_method=9 if angular integration is used with lebedev interpolation (optimized nodes on the sphere)
opt_method = 1	integer	Identifier to decide on the radiative-transfer solution method opt_method=0 if finite volume method shall be used opt_method=1 if linear short characteristics method shall be used opt_method=2 if quadratic bezier short characteristics method shall be used
opt_sol2d = f	logical	Logical to decide whether 2D solution scheme shall be applied
opt_ltec = 0	integer	Identifier to decide on the continuum wavelength/frequency model opt_ltec = 0 if single continuum frequency opt_ltec = 1 if grey approximation for continuum (frequency integrated). If this option is set, the temperature will be updated after the radiation-transfer calculations assuming radiative equilibrium (i.e., from $J = S = B = \sigma_{\rm B}/\pi T^4$).
opt_incl_cont = t	logical	Set to true (false) if continuum shall be included (or not)
opt_start_cont = t	logical	Set to true (false) if continuum iteration shall start from the beginning (or from intermediate steps)
opt_ng_cont = t	logical	Set to true (false) if Ng-extrapolation for continuum iteration shall be included or not
opt_ait_cont = f	logical	Set to true (false) if Aitkens-extrapolation for continuum iteration shall be included or not
opt_incl_line = f	logical	Set to true (false) if line shall be included (or not)
opt_start_line = t	logical	Set to true (false) if line iteration shall start from the beginning (or from intermediate steps)
opt_ng_line = t	logical	Set to true (false) if Ng-extrapolation for line iteration shall be included or not
opt_ait_line = f	logical	Set to true (false) if Aitkens-extrapolation for line iteration shall be included or not
opt_alo_cont = 3	integer	Identifier to define the approximate Λ-operator for continuum iteration opt_alo_cont = 0 if classical Λ iteration opt_alo_cont = 1 if diagonal approximate Λ operator opt_alo_cont = 2 if direct-neighbour approximate Λ operator (7 elements) opt_alo_cont = 3 if nearest-neighbour approximate Λ operator (27 elements)
opt_alo_line = 3	integer	Identifier to define the approximate Λ-operator for line iteration opt_alo_line = 0 if classical Λ iteration opt_alo_line = 1 if diagonal approximate Λ operator opt_alo_line = 2 if direct-neighbour approximate Λ operator (7 elements) opt_alo_line = 3 if nearest-neighbour approximate Λ operator (27 elements)
opt_incl_gdark = f	logical	Set to true (false) if gravity darkening by von Zeipel (1924) shall be included (or not)
opt_incl_sdist = f	logical	Set to true (false) if surface distortion due to rotation shall be included (or not)
&input_mod_1d		Input parameters of the considered star (some not required anymore)
teff = 40.d3	float	Effective temperature of the star in [K]
trad = 40.d3	float	Radiation temperature of the star (used as the inner boundary condition for the specific intensity) in [K]
xlogg = 3.5d0	float	$\log g$ of the star
rstar = 8.d0	float	R_* in $R_\odot$
lstar = 1.d6	float	L_* in $L_\odot$
rmax = 12.d0	float	Maximum radius of the computational domain in [R_*] along each x, y, z axis
tmin = .8d0	float	Minimum temperature in the wind in [$T_{\rm rad}$]
xmloss = 5.d-6	float	mass-loss rate $\dot{M}$ in $M_\odot {\rm yr}^{-1}$; only required for 1D benchmarking
vmin = 1.d1	float	minimum velocity of β-velocity law $v_{\rm min}$ in ${\rm km\,s^{-1}}$; only required for 1D benchmarking
vmax = 2.d3	float	terminal velocity of β-velocity law v_∞ in ${\rm km\,s^{-1}}$; only required for 1D benchmarking
vmicro = 1.d2	float	micro-turbulent velocity for the line-profile function $v_{\rm turb}$ in [${\rm km\,s^{-1}}$]
vth_fiducial= 1.d2	float	fiducial thermal velocity $v^*_{\rm th}$ in [${\rm km\,s^{-1}}$]
vrot = 0.d0	float	rotational velocity $v_{\rm rot}$ in [${\rm km\,s^{-1}}$]
beta = 1.d0	float	β parameter for β-velocity law; only required for 1D benchmarking models

yhe = .1d0	float	Helium abundance by number, Y_{He}
hei = 2.d0	float	Helium ionization fraction (number of free electrons per Helium-atom)
xnue0 = 1.93798d15	float	Frequency of the line transition
na = 12	integer	mass number A for the line transition
&input_infreg		Input parameters to define the computational domain (information region)
rmin = 1.d0	float	Minimum radius of the computational domain in R_*
rlim = 13.2d0	float	Maximum radius of the computational domain in R_*
&input_cont		Parameters for the continuum transport
eps_cont = 0.d0	float	Thermalization parameter ϵ_{C}
kcont = 1.d0	float	k_{C} parameter (linear scaling factor for the continuum opacity)
&input_line		Parameters of the line transport
eps_line = 0.d0	float	Line-scattering parameter ϵ_{L}
kline = 1.d0	float	line-strength parameter k_{L}
kappa0 = 1.d-1	float	Hamann (1980) parameterization
alpha = 0.5d0	float	Hamann (1980) parameterization
&dimensions_1d		Dimension parameters to set up 1D radial grid
n1d = 17	integer	number of radial grid points (used to distribute z-axis in [R_{min},R_{max}])
n1d_t = 81	integer	number of 1D grid points to set up equidistant τ-grid
n1d_r = 22	integer	number of 1D grid points to set up equidsitant v_r-grid
delv = 0.33d0	float	Preferred velocity steps Δv_r in v^*_{th}
&dimensions_3d		Dimension parameters to set up the 3D grid
ncx=19	integer	Preferred number of core-points for x-axis
ncy=19	integer	Preferred number of core-points for y-axis
ncz=19	integer	Preferred number of core-points for z-axis
delx_max=.7d0	float	Maximum allowed Δx in R_*
dely_max=.7d0	float	Maximum allowed Δy in R_*
delz_max=.7d0	float	Maximum allowed Δz in R_*
&dimensions_freq		Dimension parameters to set up the frequency grid
deltax = 0.333d0	float	Δx_{obs} steps
xcmf_max = 3.d0	float	Maximum frequency width of the line-profile function, $x^{(\mathrm{max})}_{\mathrm{cmf}}$
&dimensions_angles		Dimension parameters to set up the angular grid
n_theta = 11	integer	Number of θ angles in first octant; ϕ angles are calculated based on that
&benchmark		Parameters for setting up a benchmark
benchmark_mod = 0	integer	Identifier to define the benchmark model (set to 0 if no benchmark shall be performed)
im_source = 3	integer	see benchmark subroutines
im_opacity = 2	integer	see benchmark subroutines
im_vel = 0	integer	see benchmark subroutines
tau_min = 0.d0	float	see benchmark subroutines
tau_max = 5.d0	float	see benchmark subroutines
source_min = 0.1d0	float	see benchmark subroutines
source_max = 1.d-6	float	see benchmark subroutines
n_y = 0.d0	float	see benchmark subroutines
n_z = 0.707107d0	float	see benchmark subroutines

3.1.2 Program sc3d.eo

This program solves the non-linear coupling of the radiative transfer equation with the source function of the form:

$$
\begin{aligned}
S_{\mathrm{C}} &= (1-\epsilon_{\mathrm{C}})\, J_\nu + \epsilon_{\mathrm{C}} B_\nu \qquad (5)\\
S_{\mathrm{L}} &= (1-\epsilon_{\mathrm{L}})\, \bar{J} + \epsilon_{\mathrm{L}} B_{\nu_0}\,, \qquad (6)
\end{aligned}
$$

i.e., for a continuum consisting of thermal and scattering terms, and for a resonance-line transition approximated as a two-level atom. To this end, we are discretizing the equation of radiative transfer in Cartesian coordinates, and rely on the accelerated Λ-iteration (ALI) using non-local approximate Λ operators (ALO). The corresponding source files can be found in *src_sc3d*.

There are various different methods for solving the radiative transfer equation (e.g., via the finite-volume method or the short-characteristics method), as well as for performing the source-function updates (using different ALO's). All available options required for the input namelist are summarized in Table 9. As output and depending on the chosen options, the *.h5 file generated by *sc3d.eo* provides among other data:

scont3d The continuum source function S_{C} in cgs (3d array with dimensions (nx,ny,nz)).

mint3d The mean intensity J_ν in cgs (3d array with dimensions (nx,ny,nz)).

fcontx3d, fconty3d, fcontz3d The Eddington flux components in Cartesian coordinates, $\boldsymbol{H}_\nu = (H_x, H_y, H_z)$, in cgs (3d array with dimensions (nx,ny,nz)).

kcontxx3d, kcontyy3d, kcontzz3d, kcontxy3d, kcontxz3d, kcontyz3d The tensor components of the $\boldsymbol{K}_\nu$-tensor (3d arrays with dimensions (nx,ny,nz)). We emphasize that this is a symmetric tensor, and only six components need to be saved to deduce the complete tensor.

mintbar3d The frequency integrated and profile weighted mean intensity, $\bar{J}$ in cgs (3d array with dimensions (nx,ny,nz)).

sline3d The line source function, S_L in cgs (3d array with dimensions (nx,ny,nz)).

To plot the resulting model, we provide the PYTHON and GDL/IDL programs *plotFILES/plot_sc3d.py* and *plotFILES/plot_sc3d.pro*.

3.1.3 Program modelspec.eo

This program prepares data to be used for calculating spectral features and/or surface brightnesses of our simulations. We can either calculate a line profile from the output (i.e., source functions) of the *sc3d.eo* program, or create a completely new semi-analytic model. All source files can be found in the *src_modelspec/* directory, with the available namelist options summarized in Table 2.

To create a new model here, we essentially follow the same philosophy as for the *model.eo* program, and recommend the following two steps:

1. In *src_modelspec/modelspec.f90*, we can add a new model identifier as a case for the namelist variable inuput_mod. As before, we can then create and call a new subroutine describing our model.

2. In *src_modelspec/modelspec.f90*, we also create the new subroutine, e.g., *subroutine my_model*. Within this subroutine (or in the input namelist), we need to specify the following global variables (here for a standard 3D model in spherical coordinates).

   nr Number of radial grid points.

   ntheta Number of Θ grid points.

   nphi Number of Φ grid points.

   r The radial grid in R_*.

   theta The Θ grid in the range $[0,\pi]$.

   phi The ϕ grid in the range $[0,2\pi]$.

   sline3d The line source function in cgs (3d array with dimensions (nr,ntheta,nphi)).

   scont3d The continuum source function in cgs (3d array with dimensions (nr,ntheta,nphi)).

   t3d The gas temperature in cgs (3d array with dimensions (nr,ntheta,nphi)).

   opac3d The continuum opacity in $[1/R_*]$ (3d array with dimensions (nr,ntheta,nphi)).

   oplbar3d The frequency integrated line opacity in $[1/sR_*]$ (3d array with dimensions (nr,ntheta,nphi)).

   velx3d The x-component of the velocity field in cgs (3d array with dimensions (nr,ntheta,nphi)).

   vely3d The y-component of the velocity field in cgs (3d array with dimensions (nr,ntheta,nphi)).

   velz3d The z-component of the velocity field in cgs (3d array with dimensions (nr,ntheta,nphi)).

   xic1, xic2 The anchor for the inner boundary condition of the specific intensity, which should follow the form for core rays:

   $$I_\nu = xic1 \cdot q_1 - xic2 \cdot q_2 \,, \tag{7}$$

   where q_1 and q_2 are scaling factors to be calculated during the formal solution (e.g., q_1 can be set to account for gravity darkening). A reasonable choice, for instance might be:

   $$xic1 = B_\nu(T_\mathrm{eff}) \qquad\qquad xic2 = \frac{\mathrm{d}B\nu}{\chi_\nu \mathrm{d}z} \,. \tag{8}$$

Again, we can display the resulting model by using the programs *plotFILES/modelspec3d.py* or *plotFILES/modelspec3d.pro*.

Table 2: Input namelist for the program *modelspec.eo*

Example	Data type	Description
&input_options		Main options
input_file = './outputFILES/output_model00.h5'	string	Name of the input file generated by *sc3d.eo*, if source funtions and opacities are to be read in from the solution of *sc3d.eo*
input_file2 = './inputFILES/model3d.h5'	string	Name of the input model file generated by *model.eo*. Depending on the input_mod options, all opacities and source functions are either interpolated from the *sc3d.eo* output onto this grid, or calculated from a semi-analytical model. This procedure allows us to use a low-resolution grid for the computationally challenging ALI iteration, while still using a high-resolution grid of the wind's density and velocity structure.
output_file = './outputFILES/modspec_model00.h5'	string	Output file
input_mod = 19	integer	Identifier for the model to be calculated (see in ./src_modelspec/modelspec.f90). There are a few standard options to communicate with the output from the program *sc3d.eo*, such as:

		input_mod=11 3d model: standard ouput from sc3c.eo (3d cartesian model)
		input_mod=12 3d model: standard ouput from sc3c.eo (3d cartesian model) interpolated onto the spherical grid from the model.eo output
&input_model		Parameters of the input model
teff = 258390.7d0	float	Effective temperature of the star. Only required to get the correct photospheric line profile later on.
trad = 258390.7d0	float	Radiation temperature of the star. Only used to set the inner boundary condition for the specific intensity.
xlogg = 3.6d0	float	$\log g$ of the star. Only used to get the correct photospheric line profile later on.
rstar = 1.d0	float	R_* in $R_\odot$
rmax = 11.d0	float	$R_{\rm max}$ in R_*, used to define the computational domain
tmin = 1.d0	float	Minimum temperature of the wind in [$T_{\rm eff}$]. Only used for very specific test routines.
xmloss = 1.d-6	float	Mass-loss rate $\dot{M}$ in [$M_\odot {\rm yr}^{-1}$]. Only used for very specific test routines.
vmin = 10.d0	float	Minimum velocity $v_{\rm min}$ of a β-velocity law in [km s^{-1}]. Only used for very specific test routines
vmax = 4.d3	float	Terminal velocity v_∞ of a β-velocity law in [km s^{-1}]. **If not overwritten within the specific model routines, this sets also the range of velocities/frequencies for which the line-profiles are calculated**
beta = 1.d0	float	β parameter of a β-velocity law in [km s^{-1}]. Only used for very specific test routines
vmicro = 1.0d2	float	Microturbulent velocity $v_{\rm turb}$in [km s^{-1}].
vth_fiducial=1.d2	float	Fiducial thermal velocity to be used in [km s^{-1}].
yhe = 0.1d0	float	Helium number abundance, $Y_{\rm He} = n_{\rm He}/n_{\rm H}$ (e.g., $Y_{\rm He} = 12.25$ corresponds to mass-fraction 0.98).
hei = 2.d0	float	Number of free electrons per helium atom
&input_line		Line parameters
iline = 0	integer	Identifier for the line (as defined in src/mod_iline.f90) to get all line data (ν_0, g_l, g_u, etc) iline=0 - read atomic charge Z, element i, lower level l and upper level u from file 'in_linelist.dat' iline=1 - H_α iline=2 - H_β iline=10 - C IV resonance line iline=11 - C III 5696 line
eps_line = 0.d0	float	Line scattering parameter $\epsilon_{\rm L}$. Only used for specific test routines (Sobolev solution)
kline = 1.d0	float	Line-strength parameter or arbitrary scaling factor to increase/decrease the line opacity
kappa0 = 1.d0	float	Hamann (1980) parameterization
alpha = 0.d0	float	Hamann (1980) parameterization

3.1.4 Program spec.eo

This program calculates synthetic line profiles and surface brigthnesses for a given model obtained by the program *modelspec.eo*. To this end, we rely on a cylindric coordiante system (p, ζ, z) (see also Hennicker et al. (2021)). When calculating surface brightnesses the output will be stored as *.h5 file giving:

p The array of impact parameters.

zeta The array of angles of the cylindrical coordinate system

iem_surface The (total) emergent intensity at each p, ζ in cgs.

iemi_surface The emission part of the total intensity at each p, ζ in cgs.

iabs_surface The absorption part of the total intensity at each p, ζ in cgs.

icont_surface The continuum intensity only (if there was no line) at each p, ζ in cgs.

When calculating emergent flux profiles, the output will be stored as ASCII files in *FLUXEM_*.dat*. The output is organized in columns giving:

xobs The frequency shift from line center in units of the fiducial velocity $v_{\rm th}^*$.

flux_tot The total emergent flux-like (or rather luminosity-like) quantity at this frequency. Following, e.g., Hennicker et al. (2020, Sect. 3.7), the flux is given by:

$$F_\nu = \frac{1}{d^2} \underbrace{\int_0^{2\pi} \int_0^{R_{\rm max}} I_\nu(p, \zeta, z = R_{\rm max})\, p {\rm d}p {\rm d}\zeta}_{=:\text{flux\_tot}} . \tag{9}$$

Since we have been integrating over the impact parameter p (which internally is measured in R_*), we can translate the quantity flux_tot to a luminosity in cgs:

$$L_\nu = \text{flux\_tot} \cdot R_*^2 \cdot 4\pi . \tag{10}$$

The namelist options for the program *spec.eo* are summarized in Table 3.

Table 3: Input namelist for the program *spec.eo*

Example	Data type	Description
&input_options		Main options
input_mod = 2	integer	Type of the input model input_mod = 0 – 1D model on radial grid input_mod = 1 – 3D model on Cartesian grid input_mod = 2 – 3D model on spherical grid
input_file = './outputFILES/modspec_model00.h5'	string	Name of the input file generated by *modelspec.eo*
output_dir = './outputFILES'	string	Output directory
opt_photprof = 0	integer	Identifier for defining the photospheric line profile opt_photprof = 0 - no photospheric line profile (flat illumination) opt_photprof = 1 - from A. Herrero files opt_photprof = 2 - from Kurucz (not active at the moment) opt_photprof = 3 - from own FASTWIND compilation (only active in the binary version at the moment) opt_photprof = 4 - from Coelho et al. (2005) (only active in the binary version at the moment) opt_photprof = 5 - from Coelho (2014) (only active in the binary version at the moment)
opt_obsdir_read = t	logical	Logical to decide whether observer's direction shall be read in or calculated. opt_obsdir_read = t – read in angles $\alpha \in [0,180]$ (measured from the z-axis, inclination) and $\gamma \in [0,360]$ (measured from the x-axis, phase) from files in_alpha.dat and in_gamma.dat opt_obsdir_read = f – Equidistant α, γ grid will be calculated based on input options nalpha and ngamma.
opt_surface = t	logical	Logical to decide if surface brightness shall be calculated instead of emergent flux profiles.
opt_int2d = f	logical	Logical to decide if the propagation of intensity along a 2D slice trough the computational domain shall be calculated instead of emergent flux profiles
opt_incl_gdark = f	logical	Logical to decide if von Zeipel (1924) gravity darkening shall be included
opt_incl_sdist = f	logical	Logical to decide if surface distortion shall be accounted for
nalpha = 1	integer	Number of α angles to define the directions to the observer
ngamma = 1	integter	Number of γ angles to define the directions to the observer
&input_model		Input parameters for the model
vrot = 0.d0	float	Surface rotation of the star in [$\mathrm{km\,s^{-1}}$] (at the equator).
vth_fiducial = 1.d2	float	Fiducial thermal velocity $v^*_{\rm th}$ in[$\mathrm{km\,s^{-1}}$].
vmicro = 1.0d2	float	Microturbulent velocity $v_{\rm turb}$ in[$\mathrm{km\,s^{-1}}$].
rmin = 1.d0	float	Minimum radius of the computational domain (as used for *modelspec.eo*).
rmax = 10.97d0	float	Maximum radius of the computational domain (typically a bit smaller than used for *modelspec.eo* to avoid extrapolation errors/interpolations to zero).
&input_surface		Input parameters for surface brightness calculations and calculating intensities along a 2d slice. Will be used only if either opt_surface or opt_int2d is set to true
nsurfb = 2	integer	Number of surface brightnesses to be calculated.
alpha_surface = 1.570796d0, 1.570796d0	float	The α angles towards the observer (number of elements needs to be equal to nsurfb).
gamma_surface = 0.d0, 0.d0	float	The γ angles towards the observer (number of elements needs to be equal to nsurfb).
xobs_surface = 0.d0, 10.d0	float	The shift from line center in units of $v^*_{\rm th}$ (number of elements needs to be equal to nsurfb) For this example, two surface brightnesses will be calculated with directions and frequencies taken from (i) the first elements of the arrays and (ii) the second elements of the arrays.

3.2 boss3D

The BOSS-3D package (also within the *line3D* folder with corresponding programs *line3D/modelspec_vbin.eo* and *line3D/spec_vbin.eo*, the extension *vbin* abbreviating 'version binary') contains modules for running global (star-in-a-box) radiative transfer simulations of binary systems (see also Hennicker et al. 2021). There are two different code modules further described in the following. Each code module requires a specific input file organized by namelists.

modelspec_vbin.eo Prepares the data to be used for the line-profile calculations of binary systems. Here, we define the model in terms of density, temperature and velocity fields, as well as opacities and source functions. The corresponding namelist file is described in Table 4.

spec_vbin.eo Calculates line profiles for a specific input file. The corresponding namelist file is described in Table 5.

3.2.1 Program modelspec_vbin.eo

All source files for this program are stored in the directory *line3D/src_modelspec_vbin*. The namelist options are summarized in Table 4. Similar to the single-star code *modelspec.eo*, we recommend the following two-step approach to register a new model:

1. In *src_modelspec_vbin/modelspec.f90*, we can add a new model identifier as a case for the namelist variable inuput_mod. We can then create and call a new subroutine describing our model.

2. In *src_modelspec_vbin/modelspec.f90*, we also create the new subroutine, e.g., *subroutine my_model*. Within this subroutine, we need to specify the following global variables

   cs1_nr Number of radial grid points for the primary object's coordinate system $\Sigma^{(1)}_{\rm spc}$.

   cs1_ntheta Number of Θ grid points for the primary object's coordinate system $\Sigma^{(1)}_{\rm spc}$.

   cs1_nphi Number of Φ grid points for the primary object's coordinate system $\Sigma^{(1)}_{\rm spc}$.

   cs1_r Radial grid for the primary object in $R^{(1)}_*$.

   cs1_theta Θ grid for the primary object in the range $[0,\pi]$.

   cs1_phi Φ grid for the primary object in the range $[0,2\pi]$.

   cs1_sline3d Line source function, $S_{\rm L}$, for the primary object in cgs (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_scont3d Continuum source function, $S_{\rm C}$, for the primary object in cgs (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_rho3d Density, ρ, for the primary object in cgs (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_t3d Gas temperature, $T_{\rm gas}$, for the primary object in cgs (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_opac3d Continuum opacity, $\chi_{\rm C}$ for the primary object in $1/R^{(1)}_*$ (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_opalbar3d Frequency integrated line opacity, $\bar{\chi}$ for the primary object in Hz$/R^{(1)}_*$ (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_velx3d x-component of the velocity field for the primary object in cgs, measured in the rest-frame of the primary object (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_vely3d y-component of the velocity field for the primary object in cgs, measured in the rest-frame of the primary object (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs1_velz3d z-component of the velocity field for the primary object in cgs, measured in the rest-frame of the primary object (dimensions (nr_cs1, ntheta_cs1, nphi_cs1)).

   cs2_nr Number of radial grid points for the secondary object's coordinate system $\Sigma^{(2)}_{\rm spc}$.

   cs2_ntheta Number of Θ grid points for the secondary object's coordinate system $\Sigma^{(2)}_{\rm spc}$.

   cs2_nphi Number of Φ grid points for the secondary object's coordinate system $\Sigma^{(2)}_{\rm spc}$.

   cs2_r Radial grid for the secondary object in $R^{(2)}_*$.

   cs2_theta Θ grid for the secondary object in the range $[0,\pi]$.

   cs2_phi Φ grid for the secondary object in the range $[0,2\pi]$.

   cs2_sline3d Line source function, $S_{\rm L}$, for the secondary object in cgs (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_scont3d Continuum source function, $S_{\rm C}$, for the secondary object in cgs (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_rho3d Density, ρ, for the secondary object in cgs (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_t3d Gas temperature, $T_{\rm gas}$, for the secondary object in cgs (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_opac3d Continuum opacity, $\chi_{\rm C}$ for the secondary object in $2/R^{(2)}_*$ (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_opalbar3d Frequency integrated line opacity, $\bar{\chi}$ for the secondary object in Hz$/R^{(2)}_*$ (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_velx3d x-component of the velocity field for the secondary object in cgs, measured in the rest-frame of the secondary object (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_vely3d y-component of the velocity field for the secondary object in cgs, measured in the rest-frame of the secondary object (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

   cs2_velz3d z-component of the velocity field for the secondary object in cgs, measured in the rest-frame of the secondary object (dimensions (nr_cs2, ntheta_cs2, nphi_cs2)).

To display the resulting model, we can use the plotting routines *plotFILES/modelspec_vbin.py* or *plotFILES/modelspec_vbin.pro*.

Table 4: Input namelist for the program *modelspec_vbin.eo*, i.e., for the binary version.

Example	Data type	Description
&input_options		Main options
input_file = ”	string	Name of the input file generated by *sc3d.eo* (not used yet in binary version)
input_file2 = ”	string	Name of the input model file generated by *model.eo* (not used yet in binary version)
output_file = './outputFILES/modspec_model00.h5'	string	Output file
input_mod = 9	integer	Identifier for the model to be calculated (see in ./src_modelspec_vbin/modelspec.f90).
&input_model1		Parameters of the input model for the primary object.
rstar1 = 1.d0	float	$R_*^{(1)}$ of the primary object in $R_\odot$, defining the length scale of the coordinate system of the primary.
rmin1 = 1.d0	float	Minimum radius defining the computational domain of the primary object, $R_{\rm min}$ in $R_*^{(1)}$
rmax1 = 10.d0	float	Maximum radius defining the computational domain of the primary object, $R_{\rm max}$ in $R_*^{(1)}$
teff1 = 6.d3	float	Effective temperature of the primary object. Only required to get the correct photospheric line profile later on.
trad1 = 6.d3	float	Radiation temperature of the primary object. Only used to set the inner boundary condition for the specific intensity.
logg1 = 1.d0	float	$\log g$ of the primary object. Only used to get the correct photospheric line profile later on.
yhe1 = 0.1d0	float	Helium number abundance, $Y_{\rm He} = n_{\rm He}/n_{\rm H}$, of the primary object.
fehe1 = -1.d0	float	Fe/He abundance of primary object. Only required for Coelho et al. (2005) and Coelho (2014) photospheric line profiles.
aenh1 = 0.d0	float	α-element enhancement of primary object. Only required for Coelho et al. (2005) and Coelho (2014) photospheric line profiles.
vrot1 = 10.d0	float	Surface rotation of the primary object in [$\rm km\,s^{-1}$] (at its equator).
vmicro1 = 1.0d2	float	Microturbulent velocity of the primary object $v_{\rm turb}$in [$\rm km\,s^{-1}$].
p_object01 = 0.d0, 3.d0, 0.d0	float	x, y, z position of the primary object within the global (center-of-mass) coordinate system in units [unit_length] (see &input_units)
v_object01 = -10.d0, 0.d0, 0.d0	float	v_x, v_y, v_z components of the orbit of the primary object within the global center-of-mass coordinate system in [$\rm km\,s^{-1}$].
ex01 = 1.d0, 0.d0, 0.d0	float	Orientation of the $\boldsymbol{e}_x$ basis vector of the primary object within the global center-of-mass coordinate system.
ey01 = 0.d0, 1.d0, 0.d0	float	Orientation of the $\boldsymbol{e}_y$ basis vector of the primary object within the global center-of-mass coordinate system.
ez01 = 0.d0, 0.d0, 1.d0	float	Orientation of the $\boldsymbol{e}_z$ basis vector of the primary object within the global center-of-mass coordinate system.
rot_axis01 = 0.d0, 0.d0, 1.d0	float	Orientation of the rotation axis of the primary object within the global center-of-mass coordinate system (still to be implemented).
&input_model2		Parameters of the input model for the secondary object. Same as for primary object but interchaning the variable name index 1 with 2.
rstar2 = 3.d0	float	$R_*^{(2)}$ of the secondary object in $R_\odot$, defining the length scale of the coordinate system of the secondary.
rmin2 = 1.d0	float	Minimum radius defining the computational domain of the secondary object, $R_{\rm min}$ in $R_*^{(2)}$
rmax2 = 100.d0	float	Maximum radius defining the computational domain of the secondary object, $R_{\rm max}$ in $R_*^{(2)}$
teff2 = 10.d3	float	Effective temperature of the secondary object. Only required to get the correct photospheric line profile later on.
trad2 = 10.d3	float	Radiation temperature of the secondary object. Only used to set the inner boundary condition for the specific intensity.
logg2 = 3.d0	float	$\log g$ of the secondary object. Only used to get the correct photospheric line profile later on.
yhe2 = 0.1d0	float	Helium number abundance, $Y_{\rm He} = n_{\rm He}/n_{\rm H}$, of the secondary object.
fehe2 = -1.d0	float	Fe/He abundance of secondary object. Only required for Coelho et al. (2005) and Coelho (2014) photospheric line profiles.
aenh2 = 0.d0	float	α-element enhancement of secondary object. Only required for Coelho et al. (2005) and Coelho (2014) photospheric line profiles.
vrot2 = 100.d0	float	Surface rotation of the secondary object in [$\rm km\,s^{-1}$] (at its equator).
vmicro2 = 1.0d1	float	Microturbulent velocity of the secondary object $v_{\rm turb}$in [$\rm km\,s^{-1}$].
p_object02 = 0.d0, -2.d0, 0.d0	float	x, y, z position of the secondary object within the global (center-of-mass) coordinate system in units [unit_length] (see &input_units)
v_object02 = 6.d0, 0.d0, 0.d0	float	v_x, v_y, v_z components of the orbit of the secondary object within the global center-of-mass coordinate system in [$\rm km\,s^{-1}$].
ex02 = 1.d0, 0.d0, 0.d0	float	Orientation of the $\boldsymbol{e}_x$ basis vector of the secondary object within the global center-of-mass coordinate system.
ey02 = 0.d0, 1.d0, 0.d0	float	Orientation of the $\boldsymbol{e}_y$ basis vector of the secondary object within the global center-of-mass coordinate system.
ez02 = 0.d0, 0.d0, 1.d0	float	Orientation of the $\boldsymbol{e}_z$ basis vector of the secondary object within the global center-of-mass coordinate system.
rot_axis01 = 0.d0, 0.d0, 1.d0	float	Orientation of the rotation axis of the secondary object within the global center-of-mass coordinate system (still to be implemented).
&input_line		Line parameters
iline = 0	integer	Identifier for the line (as defined in src/mod_iline.f90) to get all line data (ν_0, g_l, g_u, etc) iline=0 - read atomic charge Z, element i, lower level l and upper level u from file 'in_linelist.dat' iline=1 - $\rm H_\alpha$ iline=2 - $\rm H_\beta$ iline=10 - C IV resonance line iline=11 - C III 5696 line
eps_line = 0.d0	float	Line scattering parameter $\epsilon_{\rm L}$. Only used for specific test routines (Sobolev solution)
kline = 1.d0	float	Line strength parameter
&input_units		Units of the simulation
unit_length = 1.d0	float	Length scale of the global coordinate system in [$R_\odot$]

vth_fiducial = 1.0d2	float	Fiducial thermal velocity $v_{\rm th}^*$.

3.2.2 Program spec_vbin.eo

The source files for this program can be found in the directory *line3D/src_spec_vbin*, with corresponding input namelist options summarized here in Table 5. The output of this program is organized the same way as for *spec.eo* (see Sect. 3.1.4), with surface-brightnesses stored on a triangulated surface though. For plotting the output data, one can use the programs *plotFILES/plot_fluxem.py* or *plotFILES/plot_fluxem.pro* to display line profiles, *plotFILES/plot_dynspec.py* for displaying the dynamical line profiles, and *plotFILES/plot_surfb_vbin.py* or *plotFILES/plot_surfb_vbin.pro* for displaying the surface brightness. Further, one can use the *plotFILES/plot_triangles.py* for displaying the triangulation.

Table 5: Input namelist for the program *spec_vbin.eo*, i.e., for the binary version

Example	Data type	Description
&input_options		Main options
input_mod = 2	integer	Type of the input model input_mod = 2 – 3D model on spherical grid
input_file = './outputFILES/modspec_model00.h5'	string	Name of the input file generated by *modelspec_vbin.eo*
output_dir = './outputFILES'	string	Output directory
opt_photprof1 = 5	integer	Identifier for defining the photospheric line profile of the primary object.
opt_photprof2 = 0	integer	Identifier for defining the photospheric line profile of the secondary object. opt_photprof = 0 - no photospheric line profile (flat illumination) opt_photprof = 1 - from A. Herrero files opt_photprof = 2 - from Kurucz (not active at the moment) opt_photprof = 3 - from own FASTWIND compilation opt_photprof = 4 - from Coelho et al. (2005) opt_photprof = 5 - from Coelho (2014)
opt_obsdir_read = t	logical	Logical to decide whether observer's direction shall be read in or calculated. opt_obsdir_read = t – read in angles $\alpha \in [0, 180]$ (measured from the z-axis of the global center-of-mass coordinate system, inclination) and $\gamma \in [0, 360]$ (measured from the x-axis of the global center-of-mass coordinate system, phase angle) from files in_alpha.dat and in_gamma.dat opt_obsdir_read = f – Equidistant α, γ grid will be calculated based on input options nalpha and ngamma.
opt_surface = t	logical	Logical to decide if surface brightness shall be calculated instead of emergent flux profiles.
opt_int2d = f	logical	Logical to decide if the propagation of intensity along a 2D slice trough the computational domain shall be calculated instead of emergent flux profiles
opt_incl_gdark1 = f	logical	Logical to decide if von Zeipel (1924) gravity darkening shall be included for primary object
opt_incl_sdist1 = f	logical	Logical to decide if surface distortion of primary object shall be accounted for
opt_incl_gdark2 = f	logical	Logical to decide if von Zeipel (1924) gravity darkening shall be included for secondary object
opt_incl_sdist2 = f	logical	Logical to decide if surface distortion of secondary object shall be accounted for
opt_pgrid01 = 'log'	string	Defining the p-grid stratification of the primary object.
opt_rgrid01 = 'log'	string	Defining the r-grid stratification of the primary object.
opt_pgrid02 = 'lin'	string	Defining the p-grid stratification of the secondary object.
opt_rgrid02 = 'lin'	string	Defining the r-grid stratification of the secondary object. 'lin' – linear stratification 'log' – logarithmic stratification 'llog' – log – log stratification
nalpha = 1	integer	Number of α angles to define the directions to the observer
ngamma = 1	integter	Number of γ angles to define the directions to the observer
&input_model		Input parameters for the model
vth_fiducial = 1.d2	float	Fiducial thermal velocity $v_{\rm th}^*$ in[$\rm km\,s^{-1}$].
&input_surface		Input parameters for surface brightness calculations and calculating intensities along a 2d slice. Will be used only if either opt_surface or opt_int2d is set to true
alpha_surface = 1.570796d0	float	The α angle towards the observer.
gamma_surface = 0.d0	float	The γ angle towards the observer.
xobs_surface = 0.d0	float	The shift from line center in units of $v_{\rm th}^*$ Note: In contrast to the single-star version, we here only allow for one surface brightness to be calculated at a time.

3.3 cont3Dslab

This folder contains code modules for running *box-in-a-star* simulations for calculating the energy density, radiation flux, and radiation pressure tensor components to be used in the future within radiation-hydrodynamic simulations. Since the computational domain is defined as a box within the stellar envelope, all code modules assume periodic boundary conditions for the specific intensity at the lateral (xz, yz) planes, the z axis defined from inside to outside the envelope. Thus, a big warning should be stated here: If the computational domain in the z-direction is large, **curvature terms are intrinsically neglected** which might have a large impact particularly for rays propagating almost horizontally. There are five different codes that can be used:

model.eo Prepares the data to be used for the actual radiative transfer calculations by transforming any input data or calculating semi-analytic models. A new model can be simply registered in src_model. Essentially, we set up the state of the gas in 3d Cartesian coordinates, described by the density ρ, the velocity field $\boldsymbol{v} = (v_x, v_y, v_z)$, and the gas and radiation temperatures, T_{gas} and T_{rad}. Currently hardcoded, the model will be saved with the file name model3d.h5.

diff1d.eo Solution of the 1D plane-parallel diffusion equation using a two-stream approximation for a β-velocity model, compared to the solution of the 1D short-characteristics or finite-volume-method solution schemes (see Table 6 for a summary of the namelist file). All output is stored by default in *outputFILES/diff1d*.

sc1d.eo 1D short-characteristics solution scheme for a plane-parallel model β-velocity model. Similar as in *diff1d.eo*, however allowing for more than two rays.

sc2d.eo 2D short-characteristics solution scheme.

sc3d.eo 3D short-characteristics solution scheme.

surfb.eo Calculating surface brightness.

3.3.1 Program diff1d.eo

The source files of this code are stored in *cont3Dslab/src_diff1d*. By default, we are only considering Thomson scattering opacities. As a benchmark for the full scattering problem, we can consider the source function given by

$$S = (1-\epsilon_{\mathrm{C}})J + \epsilon_{\mathrm{C}} B\,, \tag{11}$$

within a plane-parallel atmosphere. Defining $\mathrm{d}\tau = -\chi \mathrm{d}z$, the EQRT reads:

$$\mu \frac{\mathrm{d}I}{\mathrm{d}\tau} = I - S\,. \tag{12}$$

When assuming that the intensity depends only linearly on μ (Eddington approximation), i.e.,:

$$I(\tau,\mu) = a(\tau) + b(\tau)\mu\,, \tag{13}$$

one easily finds $K = J/3$, with $K = 1/2 \int I\mu^2 \mathrm{d}\mu$ the second moment of the specific intensity. The 0th and 1st moment of the EQRT can then be combined with Eq.(11), to obtain the diffusion equation:

$$\frac{1}{3}\frac{\mathrm{d}^2 J}{\mathrm{d}\tau^2} = \epsilon_{\mathrm{C}}(J-B)\,. \tag{14}$$

With appropriate boundary conditions, we solve this equation within this code module.

Now, considering a two-stream approximation with directions $\mu_\pm = \pm 1/\sqrt{3}$, we can calculate the moments of the specific intensity also from a direct solution to the EQRT:

$$J = \frac{1}{2}\left[\int_{-1}^{0} I_- \mathrm{d}\mu + \int_0^1 I_+ \mathrm{d}\mu\right] = \frac{1}{2}(I_+ + I_-) \tag{15}$$

$$K = \frac{1}{2}\left[\int_{-1}^{0} \mu^2 I_- \mathrm{d}\mu + \int_0^1 I_+ \mathrm{d}\mu\right] = \frac{1}{6}(I_+ + I_-) = \frac{J}{3}\,. \tag{16}$$

Thus, the Eddington approximation and the two-stream-approximation are equivalent. As a benchmark of our solution schemes, we can 'simply' calculate $I_\pm$ using finite-volume methods, short-characteristics methods, or finite differences, then iterate the source-functions to convergence, and compare the solution with the corresponding solution of the diffusion equation. The corresponding namelist file is summarized in Table 6.

Table 6: Input namelist for the program *diff1d.eo*

Example	Data type	Description
&input_options		Main options
opt_ng_cont = t	logical	Option for switching Ng extrapolation on or off
opt_ait_cont = f	logical	Option for switching Aitkens extrapolation on or off
opt_alo_cont = 1	integer	Option for the approximate Λ operator to be used.
		opt_alo_cont = 0 – classical Λ iteration
		opt_alo_cont = 1 – ALI with diagonal ALO
		opt_alo_cont = 2 – ALI with tri-diagonal ALO

&input_model		Parameters of the input model
teff = 40.d3	float	Effective temperature of the star. Only required to get the correct photospheric line profile later on.
trad = 40.d3	float	Radiation temperature of the star. Only used to set the inner boundary condition for the specific intensity.
rstar = 20.d0	float	R_* in $R_\odot$
tmin = 0.8d0	float	Minimum temperature of the wind in [$T_{\rm eff}$].
xmloss = 1.d-6	float	Mass-loss rate $\dot{M}$ in [$M_\odot \mathrm{yr}^{-1}$].
vmin = 10.d0	float	Minimum velocity $v_{\rm min}$ of a β-velocity law in [$\mathrm{km\,s^{-1}}$].
vmax = 2.d3	float	Terminal velocity v_∞ of a β-velocity law in [$\mathrm{km\,s^{-1}}$].
beta = 1.d0	float	β parameter of a β-velocity law in [$\mathrm{km\,s^{-1}}$]. Only used for very specific test routines
yhe = 0.1d0	float	Helium number abundance, $Y_{\rm He} = n_{\rm He}/n_{\rm H}$.
hei = 2.d0	float	Number of free electrons per helium atom.
xnue0 = 1.03798d15	float	Frequency at which radiation transfer equation will be solved.
&input_cont		Continuum parameters
eps_cont = 0.d0	float	Thermalization parameter
kcont = 1.d0	float	Scaling factor for continuum opacity
&dimensions_3dz		Dimension and definition of the computational domain (z-axis)
nz = 101	integer	Number of grid points.
zmin = 1.d0	float	Minimum z in R_*.
zmax = 10.d0	float	Maximum z in R_*.
&dimensions_freq		Frequency grid definition
nnue = 1	integer	Number of frequency points (only for future code updates, until now, only one frequency point implemented).
&input_diff1d		Options to set the two-stream solution method
opt_method = 4	integer	Option to set the two-stream solution method to be compared with. opt_method = 0 – 1st order finite-volume method opt_method = 1 – 2nd order finite-volume method opt_method = 2 – 1st order finite-differences method opt_method = 3 – 2nd order finite-differences method opt_method = 4 – 1st order short-characteristics method opt_method = 5 – 2nd order short-characteristics method

3.3.2 Program sc1d.eo

The source files of this code are stored in *cont3Dslab/src_sc1d*. This program essentially considers the same input model as *diff1d.eo*, however relaxing the two-stream approximation to investigate the effects of multiple angles in the RT solution scheme. Thus, the namelist file is organized as summarized in Table 6, with additional inputs described in Table 7.

Table 7: Input namelist for the program *sc1d.eo*, additionally to the one shown in Table 6.

Example	Data type	Description
&input_options		Main options
opt_angint_method = 0	integer	Identifier to decide which angular integration method to be used (currently not used) opt_angint_method = 0 – Simpson's rule in μ.
&dimensions_angles		Angular grid definition
ntheta = 5	integer	Number of θ points for the angular integration
&input_sc1d		Options to set up the short-characteristics method
opt_method = 4	integer	Option to set the order of the SC interpolation scheme. opt_method = 4 – 1st order short-characteristics method opt_method = 5 – 2nd order short-characteristics method

All output data is then saved in the directory *outputFILES/sc1d/*, with plotting routines to display the data (additionally to the data from *diff1d.eo*) in *outputFILES/plot_sc1d.pro*.

3.3.3 Program sc2d.eo

Same as program *sc1d.eo*, however solving the EQRT in two dimensions. The source files of this code are stored in *cont3Dslab/src_sc2d*, and the namelist file is summarized (in addition to the input from Table 7) in Table 8.

Table 8: Input namelist for the program *sc2d.eo*, additionally to the one shown in Table 7.

Example	Data type	Description
&dimensions_3dx		Dimension and definition of the computational domain (x-axis)
nx = 51	integer	Number of grid points.
xmin = -0.25d0	float	Minimum x in R_*.
xmax = +0.25d0	float	Maximum x in R_*.
&input_sc2d		Options to set up the 2D short-characteristics method
opt_method = 4	integer	Option to set the order of the SC interpolation scheme. opt_method = 4 – 1st order short-characteristics method opt_method = 5 – 2nd order short-characteristics method

&benchmark		Options to set up benchmarking models
benchmark_mod = 0	integer	Identifier for the benchmark model.
		benchmark_mod = 1 – Pseudo searchlight-beam test: iterating periodic boundary conditions.
theta = 3.14	float	θ angle for considered searchlight beam.
phi = 0.d0	float	Not used here

All output data is then saved in the directory *outputFILES/sc2d/*, with plotting routines to display the data in *outputFILES/plot_sc2d.pro* and *outputFILES/plot_searchlight2d.pro*.

3.3.4 Program sc3d.eo

This program calculates the radiation quantities for actual hydrodynamical simulations, and thus is not only meant for *pp*-benchmarking as the previous (1d and 2d) programs. As such, the indat file (see Table 9) is organized somewhat differently. Further, this program always needs a model to be specified by running *model.eo* beforehand (or by directly coupling to the hydro code). The source files of this code are stored in *cont3Dslab/src_sc3d*.

Table 9: Input namelist for the program *sc3d.eo*.

Example	Data type	Description
&input_options		Main options
opt_ng_cont = t	logical	Option for switching Ng extrapolation on or off
opt_ait_cont = f	logical	Option for switching Aitkens extrapolation on or off
opt_alo_cont = 1	integer	Option for the approximate Λ operator to be used.
		opt_alo_cont = 0 – classical Λ iteration
		opt_alo_cont = 1 – ALI with diagonal ALO
		opt_alo_cont = 2 – ALI with direct-neighbour ALO (7 elements)
		opt_alo_cont = 3 – ALI with nearest-neighbour ALO (27 elements)
opt_angint_method = 0	integer	Identifier to decide which angular integration method to be used (currently not used)
		opt_angint_method = 0 – Trapezoidal rule with equidistant θ, ϕ spacing.
		opt_angint_method = 1 – Trapezoidal rule with θ, ϕ spacing following Lobel & Blomme (2008).
opt_grey=2	integer	Option to set up the frequency grid
		opt_grey = 0 – Perform RT according to a grid of frequencies (to be implemented)
		opt_grey = 1 – Perform RT at a single frequency bin
		opt_grey = 2 – Perform RT in grey approximation with frequency integrated variables
opt_opac=1	integer	Define the opacity law to be used
		opt_opac = 0 – Use Thomson scattering opacity
		opt_opac = 1 – Use OPAL opacities
opt_epsc = 0		Define the 3D thermalization parameter
		opt_epsc = 0 – 3D thermalization parameter is constant and specified by input below
		opt_epsc = 1 – 3D thermalization parameter calculated from $(\chi_{\rm tot} - \chi_{\rm Thomson})/\chi_{\rm tot}$
opt_gridxyz = 1	integer	Define the grid spacing
		opt_gridxyz = 0 – linear spacing in x, y, z
		opt_gridxyz = 1 – linear spacing in x, y, and logarithmic spacing in x
verbose = t	logical	Set to true/false to print information in the terminal during runtime
model_dir = './inputFILES	string	Directory where the input model is stored
opal_dir = './opal_tables'	string	Directory where the OPAL tables are stored
&input_model		Options to set the input model
input_mod=2	integer	Identifier to decide which model to calculate (in *model.eo*)
yhe = 0.98d0	float	Helium number abundance
hei = 2.d0	float	Number of free electrons per Helium atom
&input_cont		Continuum parameters
eps_cont = 0.d0	float	Thermalization parameter (not used if opt_epsc neq 0)
kcont = 1.d0	float	Scaling factor for continuum opacity
&input_units		Definition of units to be used
unit_length = 1.d0	float	Unit of length in $R_\odot$.
unit_density = 5.d-8	float	Unit of density in cgs.
unit_velocity = 1.d8	float	Unit of velocity in cgs.
unit_temperature = 1.d0	float	Unit of temperature in cgs.
&dimensions_3dx		Dimension and definition of the computational domain (x-axis)
nx = 41	integer	Number of grid points.
xmin = -0.25d0	float	Minimum x in unit_length (see above).
xmax = 0.25d0	float	Maximum x in unit_length (see above).
&dimensions_3dy		Dimension and definition of the computational domain (y-axis)
ny = 41	integer	Number of grid points.
ymin = -0.25d0	float	Minimum y in unit_length (see above).
ymax = 0.25d0	float	Maximum y in unit_length (see above).
&dimensions_3dz		Dimension and definition of the computational domain (z-axis)
nz = 101	integer	Number of grid points.
zmin = 1.d0	float	Minimum z in unit_length (see above).
zmax = 10.d0	float	Maximum z in unit_length (see above).
&dimensions_freq		Frequency grid definition
nnue = 1	integer	Number of frequency points (only for future code updates. Thus far, only set to 1).

xnue0 = 1.93798d15	float	Frequency to be used for single-bin and non-grey RT model
&dimensions_angles		Angular grid definition
ntheta = 1	integer	Number of θ points for the angular integration for each quadrant/octant.
&input_sc3d		Options to set the solution method
opt_method = 4	integer	Option to set the two-stream solution method to be compared with. opt_method = 4 – 1st order short-characteristics method opt_method = 5 – 2nd order short-characteristics method opt_method = 6 – 1st order long-characteristics method opt_method = 7 – 2nd order ling-characteristics method opt_method = 14 – 1st order short-characteristics method without ALI (set S=B) opt_method = 15 – 2nd order short-characteristics method without ALI (set S=B) opt_method = 16 – 1st order long-characteristics method without ALI (set S=B) opt_method = 17 – 2nd order ling-characteristics method without ALI (set S=B)
hline &input_bcondition		Options to set the inner boundary condition for the specific intensity
opt_bcondition = 0	integer	Identifier to decide which inner boundary condition to be used Core intensity will be of the form $I_{\rm core} = {\rm xic1} - \mu \cdot {\rm xic2}$ opt_bcondition=0 – Use xic1 and xic2 as specified in this namelist file opt_bcondition=1 – For future code updates: xic1 = $(1-\epsilon_{\rm C}) \cdot B(T_{\rm rad}) + \epsilon_{\rm C} \cdot B\left(T_{\rm gas}\right)$ and xic2 = $\frac{{\rm d}B}{\chi {\rm d}z}$ opt_bcondition=1 – Read xic1 and xic2 from user specified routine (not implemented yet)
xic1_nue = 1.35d17	float	xic1 parameter
xic2_nue = -6.5395257d15	float	xic2 parameter
&benchmark		Options to set up benchmarking models
benchmark_mod = 0	integer	Identifier for the benchmark model. benchmark_mod = 1 – Pseudo searchlight-beam test: iterating periodic boundary conditions.
theta = 3.14	float	θ angle for considered searchlight beam.
phi = 0.d0	float	ϕ angle for considered searchlight beam.

All output data is then saved in the directory *outputFILES/sc3d/*, with plotting routines to display the data in *outputFILES/plot_sc3d.pro* and *outputFILES/plot_searchlight3d.pro* for pp test problems, and in *plotFILES/sc3d.py* otherwise.

3.4 Organization of directories

In the following, we briefly summarize the organization of all directories and subdirectories.

line3D/ – all files for star-in-a-box simulations
- in_alpha.dat – inclination angles for line-profile calculations
- in_gamma.dat – phase angles for line-profile calculations
- in_linelist.dat – linelist for line-profile calculations
- documentation/ – documentation files
- indatFILES/ – example namelist files
- inputFILES/ – default folder where models from *model.eo* are stored
- lte_tables/ – default folder where tables for LTE level populations are stored
- models/ – default folder where hydrodynamic simulations can be stored
- modules/ – module files from compilation
- objects/ – object files from compilation
- onightFILES/ – example scripts for running a grid of models ('over night')
- opal_tables/ – opacity tables from the OPAL project (Iglesias & Rogers 1996)
- outputFILES/ – default folder where all output is stored
- outputFILES_TEMP/ – default folder for temporary output
- outputFILES_TEST/ – default folder for benchmark models
- phot_flux/ – default folder for photospheric line profiles
  - blend/ – H_α line profiles and continuum levels from A. Herrero including He blend
  - sym/ – H_α line profiles and continuum levels from A. Herrero excluding He blend
  - s_coelho05/ – Coelho et al. (2005) line profiles (not uploaded to github due to storage limits)
  - s_coelho14_hrplc/ Coelho (2014) line profiles (not uploaded to github due to storage limits)
  - s_coelho14_sed/ Coelho (2014) SEDs (not uploaded to github due to storage limits)
- plotFILES/ – plotting routines and libraries
  - ps_files/ – default folder for saving ps/png files
  - animation_files/ – default folder for saving gif files
- src/ – source codes for general modules and routines
- src_lte/ – source code for communication between RT and LTE routines
  - for_levin – source code for calculating LTE level populations (Poniatowski et al. 2022)
- src_model/ – source code for setting up a general single-star model (*model.eo*)
- src_modelspec/ – source code for setting up the model for line-profile calculations (*modelspec.eo*)
- src_modelspec_vbin/ – source code for setting up the model for line-profile calculations of binary systems (*modelspec*
- src_opal/ – – source code for reading OPAL opacities
- src_photprof/ – source code for reading photospheric line profiles
- src_sc3d/ – – source code for the short-characteristics solution and ALI scheme (*sc3d.eo*)
- src_spec/ – source code for calculating synthetic line profiles (*spec.eo*)
- src_spec_vbin/ – source code for calculating synthetic line profiles of binary systems (*spec_vbin.eo*)

4 Getting started

5 Related papers

6 Cite

Depending on the code modules you are using, we would kindly ask you to cite one of the following papers:

- For the ALI scheme using the finite-volume method, please cite Hennicker et al. (2018).
- For the ALI scheme using the short-characteristics method, please cite Hennicker et al. (2020).
- For the formal solution calculating emergent flux profiles or surface brightnesses of single stars, please cite Hennicker et al. (2018) and/or Hennicker et al. (2021).
- For the formal solution calculating emergent flux profiles or surface brightnesses of binary systems, please cite Hennicker et al. (2021).
- For LTE tabulations of occupation numbers, please cite Poniatowski et al. (2022).

Thank you very much.
For further reading on the ALI method, we refer to Hennicker (2020).

7 Developers and contributors

These code modules have been developed in collaboration with: N. Moens, L. Poniatowski., J. Puls, S. Sundqvist. The radiative transfer modules use parts of the GEOMPACK2 library[1] and EISPACK libraries[2] (see Joe (1991)).

8 Known problems and solutions

Below, you can find a list of known problems and (possible) solutions:

OMP is not working: There are (at least) two possibilities that can cause these problems: OMP_FLAG is not set in the Makefile (set it to -fopenmp). Alternatively, you might not have set the enivironment variable on your system (in the terminal, simply use export OMP_NUM_THREADS=N, with N the number of OMP threads to be used, e.g.,, 12).

Segmentation fault when running in OMP mode: By default, the stacksize for each thread can be very low. Particularly when requiring huge arrays in the formal solution with a lot of grid refinement (e.g.,for LDI simulations), the local copies in each thread might run out of stacksize. Probably at the expense of computing efficiency, this problem can be solved by setting the corresponding environment variable: export OMP_STACKSIZE=10M (or larger if required).

Comiplation errors of HDF5 Sometimes, a new fortran compiler is not compatible with an old HDF5 version. Then you might want to switch to a later HDF5 release (version 1.10.7 or higher), or downgrade your fortran compiler.

Mac – Illegal Instruction 4: On the Mac, an Illegal Instruction 4 error can occur when static arrays are not properly initialized. To solve this issue, and to still be able to use OpenMP parallelization, please dynamically allocate static arrays, e.g.:
real(dp), dimension(nd) :: my_array
becomes
real(dp), dimension(:), allocatable :: my_array
allocate(my_array(nd))

9 ToDo

- In src/mod_iline.f90, LTE tables are read in only for 'lte_tables/Y02800'

10 Acknowledgements

We like to thank B. Joe and J. Burkardt for providing the GEOMPACK2 libraries. Further, we thank the developers of the EISPACK library.

The development of all code modules has been funded by the German Research Foundation, DFG, under grant PU 117/9-1, and by the Odysseus program of the Belgian Research Foundation Flanders (FWO) under grant G0H9218N.

References

Coelho, P., Barbuy, B., Meléndez, J., Schiavon, R. P., & Castilho, B. V. 2005: *A library of high resolution synthetic stellar spectra from 300 nm to 1.8 μm with solar and α-enhanced composition*, A&A, 443, 735

Coelho, P. R. T. 2014: *A new library of theoretical stellar spectra with scaled-solar and α-enhanced mixtures*, MNRAS, 440, 1027

Hamann, W.-R. 1980: *The expanding envelope of Zeta Puppis - A detailed UV-line fit*, A&A, 84, 342

Hennicker, L. 2020: *3D radiative transfer – continuum and line formation in hot star winds*, PhD thesis, Ludwig-Maximilians-Universität München

[1] https://people.math.sc.edu/Burkardt/f_src/geompack2/geompack2.html
[2] https://people.sc.fsu.edu/~jburkardt/f77_src/eispack/eispack.html

Hennicker, L., Kee, N. D., Shenar, T., Bodensteiner, J., Abdul-Masih, M., El Mellah, I., Sana, H., & Sundqvist, J. O. 2021: *Binary-object spectral-synthesis in 3D (BOSS-3D) – Modelling H-alpha emission in the enigmatic multiple system LB-1*, arXiv e-prints, arXiv:2111.15345

Hennicker, L., Puls, J., Kee, N. D., & Sundqvist, J. O. 2018: *3D radiative transfer: Continuum and line scattering in non-spherical winds from OB stars*, A&A, 616, A140

Hennicker, L., Puls, J., Kee, N. D., & Sundqvist, J. O. 2020: *A 3D short-characteristics method for continuum and line scattering problems in the winds of hot stars*, A&A, 633, A16

Iglesias, C. A. & Rogers, F. J. 1996: *Updated Opal Opacities*, ApJ, 464, 943

Joe, B. 1991: *GEOMPACK — a software package for the generation of meshes using geometric algorithms*, Advances in Engineering Software and Workstations, 13, 325

Lobel, A. & Blomme, R. 2008: *Modeling Ultraviolet Wind Line Variability in Massive Hot Stars*, ApJ, 678, 408

Poniatowski, L. G., Kee, N. D., Sundqvist, J. O., Driessen, F. A., Owocki, S. P., Gayley, K. G., Decin, L., de Koter, A., & Sana, H. 2022: *Method and new tabulations for flux-weighted line-opacity and radiation line-force in supersonic media*, A&A, submitted

von Zeipel, H. 1924: *The radiative equilibrium of a rotating system of gaseous masses*, MNRAS, 84, 665