Documentation

Levin Hennicker

February 10, 2022

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. 3.2).

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. 3.3).

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 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

2.2 Getting the code

You can get the code from github:

- 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/cont3D

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., $HOME/hdf5_gfortran_build):
   ./configure --prefix=$HOME/hdf5_gfortran_build --enable-fortran (and if required --enable-cxx)
4. Installation
   make (and watch for fatal errors)
   make check (and verify that all tests return a 'pass')
   make install
5. Include the library path in the Makefile of the main code (e.g., line3D/Makefile)
6. Update your .bashrc file to add the library path to your LD environment variable:
   HDF5_PATH=$HOME/hdf5_gfortran_build/hdf5_lib/lib
   LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ADD_LIB_PATH
   export LD_LIBRARY_PATH

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

1. Install the CommandLineTools
2. Install homebrew:
   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.

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 1. 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 1.

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 The namelist files

The namelist options for the programs *model.eo* and *sc3d.eo* are summarized in Table 1, and the namelist options for the programs *modelspec.eo*, *spec.eo* are shown in Tables Tables 2, 3, 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)
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, $Y_{\rm He}$
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 $\epsilon_{\rm C}$
kcont = 1.d0	float	$k_{\rm C}$ parameter (linear scaling factor for the continuum opacity)
&input_line		Parameters of the line transport
eps_line = 0.d0	float	Line-scattering parameter $\epsilon_{\rm L}$
kline = 1.d0	float	line-strength parameter $k_{\rm L}$
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 $[R_{\rm min}, R_{\rm max}]$)
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 $v^*_{\rm th}$
&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	$\Delta x_{\rm obs}$ steps
xcmf_max = 3.d0	float	Maximum frequency width of the line-profile function, $x^{(\rm max)}_{\rm cmf}$
&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

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).
&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 $\upsilon_{\rm min}$ of a β-velocity law in [${\rm km\,s^{-1}}$]. Only used for very specific test routines
vmax = 4.d3	float	Terminal velocity υ_∞ of a β-velocity law in [${\rm km\,s^{-1}}$]. **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 [${\rm km\,s^{-1}}$]. Only used for very specific test routines
vmicro = 1.0d2	float	Microturbulent velocity $\upsilon_{\rm turb}$in [${\rm km\,s^{-1}}$].
vth_fiducial=1.d2	float	Fiducial thermal velocity to be used in [${\rm km\,s^{-1}}$].
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$_\alpha$
		iline=2 - 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 or arbitrary scaling factor to increase/decrease the line opacity
kappa0 = 1.d0	float	Hamann (1980) parameterization
alpha = 0.d0	float	Hamann (1980) parameterization

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	integer	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 [${\rm km\,s^{-1}}$] (at the equator).
vth_fiducial = 1.d2	float	Fiducial thermal velocity $\upsilon_{\rm th}^*$ in[${\rm km\,s^{-1}}$].
vmicro = 1.0d2	float	Microturbulent velocity $\upsilon_{\rm turb}$ in[${\rm km\,s^{-1}}$].

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 cont3Dslab

3.3 boss3D

The BOSS-3D package (also within the *line3D* folder) contains modules for running global (star-in-a-box) radiative transfer simulations of binary systems. 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.

3.3.1 The namelist files

The namelist options for the programs *modelspec_vbin.eo*, and *spec_vbin.eo* are shown in Tables 4 and 5, respectively.

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 - H$_\alpha$
		iline=2 - 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 $v_{\rm th}^*$.

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	integer	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.

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 **?** (ToDo Luka).

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.

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

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.

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

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

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

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