radmc3dPy

v0.24

Attila Juhász

juhasz@strw.leidenuniv.nl

Generated by Doxygen 1.8.3

March 27, 2013

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Namespace Documentation

3.1 radmc3dPy Namespace Reference

Namespaces

- namespace analyze
- namespace crd_trans
- namespace image
- namespace model_lines_nlte_lvg_1d_1
- namespace model_ppdisk
- namespace model_simple_1
- namespace model_spher1d_1
- namespace model_spher2d_1
- namespace model_template
- namespace model_test_scattering_1
- namespace natconst
- namespace setup

Variables

- string **__version__** "0.24"
- list **__all__** ["analyze", "setup", "image", "crd_trans", "natconst"]

3.1.1 Detailed Description

```
RADMC-3D Python module
(c) Attila Juhasz, Leiden, 2011,2012,2013
```

3.2 radmc3dPy.analyze Namespace Reference

Classes

- class radmc3dGrid
- class radmc3dData
- class radmc3dStars
- class radmc3dDustOpac
- class radmc3dPar

Functions

- def readopac
- def read_data
- def read_grid
- def readparams
- def write_default_parfile
- def read_spectrum

3.2.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz 2011,2012,2013

This sub-module contains classes and functions to read and write input/output data to/from RADMC3D

CLASSES:
--------
   radmc3dData
   radmc3dDustOpac
   radmc3dGrid
   radmc3dStars

FUNCTIONS:
----------
    read_data()
    read_grid()
    read_masteropac()
    write_masteropac()
    readopac()
    readparams()
```

3.2.2 Function Documentation

3.2.2.1 def radmc3dPy.analyze.read_data (*ddens* = `False`, *dtemp* = `False`, *gdens* = `False`, *gtemp* = `False`, *gvel* = `False`, *ispec* = `None`, *vturb* = `False`, *binary* = `True`)

```
Function to read the model data (e.g. density, velocity, temperature)

INPUT:
------
    ddens - If True dust density will be read (all dust species and grain sizes)
    dtemp - If True dust temperature will be read (all dust species and grain sizes)
    gdens - If True gas density will be read (NOTE: the gas density will be number density in 1/cm^3)
    gtemp - If True gas temperature will be read (all dust species and grain sizes)
    gvel  - If True the velocity field will be read
    ispec - Name of the molecule in the 'molecule_ispec.inp' filename

OUTPUT:
------
    Returns an instance of the radmc3dData class with the following attributes:
        rhodust   - Dust density in g/cm^3
        dusttemp  - Dust temperature in K
        rhogas    - Gas density in molecule/cm^3
        gasvel    - Gas velocity in cm/s
        gastemp   - Gas temperature in K
        vturb     - Mictroturbulence in cm/s
        taux      - Optical depth along the x (cartesian) / r (cylindrical) / r (spherical) dimension
        tauy      - Optical depth along the y (cartesian) / theta (cylindrical) / theta (spherical) dimension
        tauz      - Optical depth along the z (cartesian) / z (cylindrical) / phi (spherical) dimension
        sigmadust - Dust surface density in g/cm^2
        sigmagas  - Gas surface density in molecule/cm^2 (or g/cm^2 depending on the dimension of rhogas)
```

3.2.2.2 def radmc3dPy.analyze.read_grid ( )

```
Function to read the spatial and frequency grid

OUTPUT
------

    Returns an instance of the radmc3dGrid class with the following attributes:

    crd_sys    - 'car'/'cyl'/'sph' coordinate system of the spatial grid
    act_dim    - A three element vector the i-th element is 1 if the i-th dimension is active, otherwize the i
    nx         - Number of grid points in the x (cartesian) / r (cylindrical) / r (spherical) dimension
    ny         - Number of grid points in the y (cartesian) / theta (cylindrical) / theta (spherical) dimensic
    nz         - Number of grid points in the z (cartesian) / z (cylindrical) / phi (spherical) dimension
    nxi        - Number of cell interfaces in the x (cartesian) / r (cylindrical) / r (spherical) dimension
    nyi        - Number of cell interfaces in the y (cartesian) / theta (cylindrical) / theta (spherical) dime
    nzi        - Number of cell interfaces in the z (cartesian) / z (cylindrical) / phi (spherical) dimension
    nwav       - Number of wavelengths in the wavelength grid
    freq       - Number of frequencies in the grid (equal to nwav)
    x          - Cell centered x (cartesian) / r (cylindrical) / r (spherical)  grid points
    y          - Cell centered y (cartesian) / theta (cylindrical) / theta (spherical)  grid points
    z          - Cell centered z (cartesian) / z (cylindrical) / phi (spherical)  grid points
    xi         - Cell interfaces in the x (cartesian) / r (cylindrical) / r (spherical)  dimension
    yi         - Cell interfaces in the y (cartesian) / theta (cylindrical) / theta (spherical)  dimension
    zi         - Cell interfaces in the z (cartesian) / z (cylindrical) / phi (spherical)  dimension
    wav        - Wavelengh  grid
    freq       - Frequency  grid
```

3.2.2.3 def radmc3dPy.analyze.read_spectrum (*fname* = '')

```
Function to read the spectrum / SED


OPTIONS:
--------
    fname - Name of the file to be read


OUTPUT:
-------
    Returns a two dimensional Numpy array with [Nwavelength, 2] dimensions
    [Nwavelength,0] is the wavelength / velocity and
    [Nwavelength,1] is the flux density
```

3.2.2.4 def radmc3dPy.analyze.readopac (*ext* = [''], *idust* = None, *used* = False)

```
Function to read the dust opacity files
This function is an interface to radmc3dDustOpac.readopac()

INPUT:
------
    ext  : file name extension (file names should look like 'dustkappa_ext.inp')
    idust: index of the dust species in the master opacity file (dustopac.inp')
    used : if set to True the used dust opacity file ('dustkappa_ext.inp.used') file is read,
            which is interpolated to the frequency grid of the RADMC3D run

OUTPUT:
-------
    Returns an instance of the radmc3dDustOpac class with the following attributes:

    wav     - wavelength grid
    freq    - frequency grid
    nwav    - number of wavelengths
    kabs    - absorption coefficient per unit mass
    ksca    - scattering coefficient per unit mass
    phase_g - phase function
    ext     - if set it contains the file name extension of the duskappa_ext.Kappa file
    therm   - if False the dust grains are quantum-heated (default: True)
```

```
    idust   - index of the dust species in the dust density distribution array
```

3.2.2.5 def radmc3dPy.analyze.readparams ( )

```
Function to read the problem_params.inp file (interface function to radmc3dPar.readparams())

OUTPUT:
-------
    Returns an instance of the radmc3dPar class with the following attributes:

    ppar   : Dictionary containing parameter values with parameter names as keys
    pdesc  : Disctionary containing parameter description (comments in the parameter file) with parameter name
    pblock : Dictionary containing the block names in the parameter file and parameter names as values
    pvalstr: Dictionary containing parameter values as strings with parameter names as keys
```

3.2.2.6 def radmc3dPy.analyze.write_default_parfile (*model* = ' ', *fname* = ' ')

```
Function to write a parameter file (problem_params.inp) with default parameters for a given model

INPUT:
------
    model - Name of the model whose parameter should be written to the file

OPTIONS:
--------
    fname - Name of the parameter file to be written (if omitted problem_params.inp will be used)
```

3.3 radmc3dPy.crd_trans Namespace Reference

Functions

- def ctrans_sph2cyl
- def ctrans_sph2cart
- def vtrans_sph2cart
- def csrot
- def vrot

3.3.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz 2011,2012,2013

This sub-module contains functions for coordinate transformations (e.g. rotation)
```

3.3.2 Function Documentation

3.3.2.1 def radmc3dPy.crd_trans.csrot (*crd* = None, *ang* = None, *xang* = 0.0, *yang* = 0.0, *zang* = 0.0, *deg* = False)

```
Function to make coordinate system rotation

INPUT :
-------
       crd  : three element vector containing the coordinates of a
              given point in a cartesian system

       ang  : three element array, angles of rotation around the x,y,z axes
```

```
OPTIONS :
---------
       deg=True : if this keyword is set angles should be given in
              angles instead of radians (as by default)

Rotation matrices :
-------------------
X-axis

 |      1              0            0        |
 |      0          cos(alpha)    -sin(alpha)  |
 |      0          sin(alpha)     cos(alpha)  |

Y-axis

 |   cos(beta)          0          sin(beta)   |
 |      0               1             0        |
 |   -sin(beta)         0          cos(beta)   |

Z-axis

 |   cos(gamma)     -sin(gamma)       0         |
 |  sin(gamma)       cos(gamma)       0         |
 |      0                0            1         |
```

3.3.2.2 def radmc3dPy.crd_trans.ctrans_sph2cart (*crd* = [0, *reverse* = False)

```
Function to transform coordinates between spherical to cartesian systems

INPUT :
-------
        crd      : Three element array containing the input
                   coordinates [x,y,z] or [r,phi,theta] by default
                   the coordinates assumed to be in the cartesian system
OPTIONS :
---------
        reverse=False : Calculates the inverse trasnformation
                   (cartesian -> spherical). In this case crd should be [r,phi,theta]

OUTPUT :
--------
        result   : A three element array containg the output
                   coordinates [r,phi,theta] or [x,y,z]
```

3.3.2.3 def radmc3dPy.crd_trans.ctrans_sph2cyl (*crd* = None, *theta* = None, *reverse* = False)

```
Function to transform coordinates between spherical to cylindrical systems

INPUT :
-------
        r,phi,theta : numpy arrays containing the spherical coordinates

OPTIONS :
----------
        reverse=False : Calculates the inverse trasnformation
                   (cartesian -> spherical). In this case crd should be [r,phi,theta]

OUTPUT :
--------
        result   : a numpy array of [Nr,Nphi,Ntheta,3] dimensions containing the cylindrical
                   coordinates [rcyl, z, phi]
```

3.3.2.4 def radmc3dPy.crd_trans.vrot (*crd* = None, *v* = None, *ang* = None)

```
Function to rotate a vector in spherical coordinate system
```

```
First transform the vector to cartesian coordinate system do the rotation then make the
 inverse transformation

INPUT :
-------
       crd  : three element vector containing the coordinates of a
              given point in a cartesian system

       v    : three element array, angles of rotation around the x,y,z axes

       ang  : angle around the x, y, z, axes with which the vector should be rotated
```

3.3.2.5 def radmc3dPy.crd_trans.vtrans_sph2cart (*crd* = [0, *v* = [0, *reverse* = False)

```
Function to transform velocities between spherical to cartesian systems

INPUT :
-------
        crd      : Three element array containing the input
                   coordinates [x,y,z] or [r,phi,theta] by default
                   the coordinates assumed to be in the cartesian system

        v        : Three element array containing the input
                   velocities in the same coordinate system as crd


OPTIONS :
---------
        reverse=False : Calculates the inverse trasnformation (cartesian -> spherical)

OUTPUT :
--------
        result   : A three element array containg the output
                   velocities [vr,vphi,vtheta] or [vx,vy,vz]


NOTE!!!!! The velocities in the spherical system are not angular velocities!!!!
v[1] = dphi/dt * r
v[2] = dtheta/dt * r
```

3.4 radmc3dPy.image Namespace Reference

Classes

- class radmc3dImage

Functions

- def get_psf
- def readimage
- def plotimage
- def makeimage
- def cmask

3.4.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz 2011,2012,2013

This sub-module contains classes/functions to create and read images with radmc3d and to calculate
```

```
interferometric visibilities and write fits files
For help on the syntax or functionality of each function see the help of the individual functions

CLASSES:
--------
radmc3dImage - RADMC3D image class
radmc3dVisibility - Class of interferometric visibilities

FUNCTIONS:
----------

get_psf() - Calculates a Gaussian PSF/beam
get_visibility() - Calculates interferometric visiblities
makeimage() - Runs RADMC3D to calculate images/channel maps
plotimage() - Plots the image
readimage() - Reads RADMC3D image(s)
```

3.4.2 Function Documentation

3.4.2.1 def radmc3dPy.image.cmask (*im* = None, *rad* = 0.0, *au* = False, *arcsec* = False, *dpc* = None)

```
Function to simulate a coronographic mask by
setting the image values to zero within circle of a given radius around the
image center

INPUT:
------
    im     : a radmc3dImage class
    rad    : radius of the mask
    au     : if true the radius is taken to have a unit of AU
    arcsec : if true the radius is taken to have a unit of arcsec (dpc
              should also be set)
    dpc    : distance of the source (required if arcsec = True)

    NOTE: if arcsec=False and au=False rad is taken to have a unit of pixel

OUTPUT:
-------
    res    : a radmc3dImage class containing the masked image
```

3.4.2.2 def radmc3dPy.image.get_psf (*nx* = None, *ny* = None, *fwhm* = None, *pa* = None, *pscale* = None)

```
Function to generate a two dimensional Gaussian PSF

INPUT:
------
      nx      : image size in the first dimension
      ny      : image size in the second dimension
      fwhm    : full width at half maximum of the psf in each dimension [fwhm_x, fwhm_y]
      pa      : position angle of the gaussian if the gaussian is not symmetric
      pscale  : pixelscale of the image, if set fwhm should be in the same unit, if not set unit of fwhm is pi

OUTPUT:
-------
      result  : dictionary containing the following keys
      'psf'   : two dimensional numpy array containing the normalized psf
      'x'     : first coordinate axis of the psf
      'y'     : seonc coordinate axis of the psf
```

3.4.2.3 def radmc3dPy.image.makeimage (*npix* = None, *incl* = None, *wav* = None, *sizeau* = None, *phi* = None, *posang* = None, *pointau* = None, *fluxcons* = True, *nostar* = False, *noscat* = False, *widthkms* = None, *linenlam* = None, *vkms* = None, *iline* = None)

```
Function to call RADMC3D to calculate a rectangular image
```

```
SYNTAX:
-------
       makeimage(npix=100, incl=60.0, wav=10.0, sizeau=300., phi=0., posang=15.,
                 pointau=[0., 0.,0.], fluxcons=True, nostar=False, noscat=False)

INPUT:
------
       npix    : number of pixels on the rectangular images
       sizeau  : diameter of the image in au
       incl    : inclination angle of the source
       dpc     : distance of the source in parsec
       phi     : azimuthal rotation angle of the source in the model space
       posang  : position angle of the source in the image plane
       pointau : three elements list of the cartesian coordinates of the image center
       widthkms: width of the frequency axis of the channel maps
       linenlam: number of wavelengths to calculate images at
       vkms    : a single velocity value at which a channel map should be calculated
       iline   : line transition index

KEYWORDS:
---------
       fluxcons : this should not even be a keyword argument, it ensures flux conservation
       (adaptive subpixeling) in the rectangular images
       nostar   : if True the calculated images will not contain stellar emission
       noscat   : if True, scattered emission will be neglected in the source function, however,
                      extinction will contain scattering if kappa_scat is not zero.
```

3.4.2.4 def radmc3dPy.image.plotimage (*image* = None, *arcsec* = False, *au* = False, *log* = False, *dpc* = None, *maxlog* = None, *saturate* = None, *bunit* = None, *ifreq* = None, *cmask_rad* = None, *interpolation* = 'nearest', *cmap* = cm.gist_gray, *kwargs*)

```
Function to plot a radmc3d image

SYNTAX:
-------
      result = plotimage(image='image.out', arcsec=True, au=False, log=True, dpc=140, maxlog=-6.,
                         saturate=0.1, bunit='Jy')

INPUT:
------
      image     : A radmc3dImage class returned by readimage
      arcsec    : If True image axis will have the unit arcsec (NOTE: dpc keyword should also be set!)
      au        : If True image axis will have the unit AU
      log       : If True image scale will be logarithmic, otherwise linear
      dpc       : Distance to the source in parsec (This keywords should be set if arcsec=True, or bunit!=None)
      maxlog    : Logarithm of the lowest pixel value to be plotted, lower pixel values will be clippde
      saturate  : Highest pixel values to be plotted in terms of the peak value, higher pixel values will be cl
      bunit     : Unit of the image, (None - Inu/max(Inu), 'inu' - Inu, fnu - Jy/pixel)
      ifreq     : If the image file/array consists of multiple frequencies/wavelengths ifreq denotes the index
                  of the frequency/wavelength in the image array to be plotted
      cmask_rad : Simulates coronographyic mask : sets the image values to zero within this radius of the imag
                   The unit is the same as the image axis (au, arcsec, cm)
                   NOTE: this works only on the plot, the image array is not changed (for that used the cmask()

      cmap      : matplotlib color map
      interpolation: interpolation keyword for imshow (e.g. 'nearest', 'bilinear', 'bicubic')
```

3.4.2.5 def radmc3dPy.image.readimage (*fname* = None)

```
Function to read an image calculated by RADMC3D

INPUT:
------
    fname   : file name of the radmc3d output image (if omitted 'image.out' is used)
```

3.5 radmc3dPy.model_lines_nlte_lvg_1d_1 Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_gas_temperature
- def get_dust_temperature
- def get_gas_abundance
- def get_gas_density
- def get_dust_density
- def get_vturb
- def get_velocity

3.5.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz, Kees Dullemond 2011,2012,2013

Original IDL model by Kees Dullemond, Python translation by Attila Juhasz
```

3.5.2 Function Documentation

3.5.2.1 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_default_params ( )

```
Function to provide default parameter values

OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.5.2.2 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_desc ( )

```
Function to provide a brief description of the model
```

3.5.2.3 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_dust_density (*grid* = None, *ppar* = None)

```
Function to create the dust density distribution

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.5.2.4 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_dust_temperature (*grid* = None, *ppar* = None)

```
Function to calcualte/set the dust temperature

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the dust temperature in K
```

3.5.2.5 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_gas_abundance (*grid* = None, *ppar* = None, *ispec* = '')

```
Function to create the conversion factor from volume density to number density of molecule ispec.
The number density of a molecule is rhogas * abun

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model
    ispec - The name of the gas species whose abundance should be calculated

OUTPUT:
-------
    returns the abundance as a Numpy array
```

3.5.2.6 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_gas_density (*grid* = None, *ppar* = None)

```
Function to create the total gas density distribution

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.5.2.7 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_gas_temperature (*grid* = None, *ppar* = None)

```
Function to calcualte/set the gas temperature

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the gas temperature in K
```

3.5.2.8 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_velocity (*grid* = None, *ppar* = None)

```
Function to create the turbulent velocity field

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model
```

```
OUTPUT:
-------
    returns the turbulent velocity in cm/s
```

3.5.2.9 def radmc3dPy.model_lines_nlte_lvg_1d_1.get_vturb (*grid* = None, *ppar* = None)

```
Function to create the turbulent velocity field

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the turbulent velocity in cm/s
```

3.6 radmc3dPy.model_ppdisk Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_dust_density
- def get_gas_density
- def get_gas_abundance
- def get_velocity

3.6.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz 2011,2012,2013

Generic protoplanetary disk model

The density is given by

            Sigma(r,phi)          /    z^2     \
    rho =  ---------------- * exp| - -------- |
            Hp * sqrt(2*pi)       \    2*Hp^2 /

    Sigma - surface density
    Hp    - Pressure scale height
    Hp/r  = hrdisk * (r/rdisk)^plh

FUNCTIONS:
----------

get_dust_density() - Calculates the dust volume density in g/cm^3
get_gas_density()  - Calculates the gas volume density in g/cm^3
get_velocity()     - Calculates the velocity field
get_desc()         - Returns the short description of this model
```

3.6.2 Function Documentation

3.6.2.1 def radmc3dPy.model_ppdisk.get_default_params ( )

```
Function to provide default parameter values
```

```
OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.6.2.2 def radmc3dPy.model_ppdisk.get_desc ( )

```
Function to provide a brief description of the model
```

3.6.2.3 def radmc3dPy.model_ppdisk.get_dust_density (*rcyl* = None, *phi* = None, *z* = None, *z0* = None, *hp* = None, *sigma* = None, *grid* = None, *ppar* = None)

```
Function to create the density distribution in a protoplanetary disk

OUTPUT:
-------
    returns the volume density in g/cm^3, whether the density is that of the gas
    or dust or both depends on what is specified in the surface density/mass
```

3.6.2.4 def radmc3dPy.model_ppdisk.get_gas_abundance (*grid* = None, *ppar* = None, *ispec* = '')

```
Function to create the conversion factor from volume density to number density of molecule ispec.
The number density of a molecule is rhogas * abun

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model
    ispec - The name of the gas species whose abundance should be calculated

OUTPUT:
-------
    returns the abundance as a Numpy array
```

3.6.2.5 def radmc3dPy.model_ppdisk.get_gas_density (*rcyl* = None, *phi* = None, *z* = None, *z0* = None, *hp* = None, *sigma* = None, *grid* = None, *ppar* = None)

```
Function to create the density distribution in a protoplanetary disk

OUTPUT:
-------
    returns the volume density in g/cm^3, whether the density is that of the gas
    or dust or both depends on what is specified in the surface density/mass
```

3.6.2.6 def radmc3dPy.model_ppdisk.get_velocity (*rcyl* = None, *phi* = None, *z* = None, *z0* = None, *grid* = None, *ppar* = None)

```
Function to create the velocity field in a protoplanetary disk
```

3.7 radmc3dPy.model_simple_1 Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_dust_density

3.7.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz, Kees Dullemond 2011,2012,2013

Original IDL model by Kees Dullemond, Python translation by Attila Juhasz
```

3.7.2 Function Documentation

3.7.2.1 def radmc3dPy.model_simple_1.get_default_params ( )

```
Function to provide default parameter values

OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.7.2.2 def radmc3dPy.model_simple_1.get_desc ( )

```
Function to provide a brief description of the model
```

3.7.2.3 def radmc3dPy.model_simple_1.get_dust_density (*grid* = None, *ppar* = None)

```
Function to create the dust density distribution

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.8 radmc3dPy.model_spher1d_1 Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_dust_density

3.8.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz, Kees Dullemond 2011,2012,2013

Original IDL model by Kees Dullemond, Python translation by Attila Juhasz
```

3.8.2 Function Documentation

3.8.2.1 def radmc3dPy.model_spher1d_1.get_default_params ( )

```
Function to provide default parameter values

OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.8.2.2 def radmc3dPy.model_spher1d_1.get_desc ( )

```
Function to provide a brief description of the model
```

3.8.2.3 def radmc3dPy.model_spher1d_1.get_dust_density (*grid* = None, *ppar* = None)

```
Function to create the dust density distribution

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.9 radmc3dPy.model_spher2d_1 Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_dust_density

3.9.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz, Kees Dullemond 2011,2012,2013

Original IDL model by Kees Dullemond, Python translation by Attila Juhasz
```

3.9.2 Function Documentation

3.9.2.1 def radmc3dPy.model_spher2d_1.get_default_params ( )

```
Function to provide default parameter values
```

```
OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.9.2.2 def radmc3dPy.model_spher2d_1.get_desc ( )

```
Function to provide a brief description of the model
```

3.9.2.3 def radmc3dPy.model_spher2d_1.get_dust_density (*grid* = None, *ppar* = None)

```
Function to create the dust density distribution

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.10 radmc3dPy.model_template Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_gas_temperature
- def get_dust_temperature
- def get_gas_abundance
- def get_gas_density
- def get_dust_density
- def get_vturb
- def get_velocity

3.10.1 Detailed Description

```
This is a radmc3dPy model template
A radmc3dPy model file can contain any / all of the functions below

    get_default_params()
    get_desc()
    get_dust_density()
    get_dust_temperature()
    get_gas_abundance()
    get_gas_density()
    get_gas_temperature()
    get_velocity()
    get_vturb()

The description of the individual functions can be found in the docstrings below the function name.
If a model does not provide a variable or the variable should be calculated by RADMC-3D
(e.g. dust temperature) the corresponding function (e.g. get_dust_temperature) should be removed from
or commented out in the model file.
```

```
NOTE: When using this template it is strongly advised to renme the template model (to e.g. model_mydisk.py)
as the get_model_names() function in the setup module removes the name 'template' from the list of available
models.
```

3.10.2 Function Documentation

3.10.2.1 def radmc3dPy.model_template.get_default_params ( )

```
Function to provide default parameter values

OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.10.2.2 def radmc3dPy.model_template.get_desc ( )

```
Function to provide a brief description of the model
```

3.10.2.3 def radmc3dPy.model_template.get_dust_density (*grid* = None, *ppar* = None)

```
Function to create the dust density distribution

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.10.2.4 def radmc3dPy.model_template.get_dust_temperature (*grid* = None, *ppar* = None)

```
Function to calcualte/set the dust temperature

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the dust temperature in K
```

3.10.2.5 def radmc3dPy.model_template.get_gas_abundance (*grid* = None, *ppar* = None, *ispec* = '')

```
Function to create the conversion factor from volume density to number density of molecule ispec.
The number density of a molecule is rhogas * abun

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
```

```
    ppar - Dictionary containing all parameters of the model
    ispec - The name of the gas species whose abundance should be calculated

OUTPUT:
-------
    returns the abundance as a Numpy array
```

3.10.2.6 def radmc3dPy.model_template.get_gas_density (*grid* = None, *ppar* = None)

```
Function to create the total gas density distribution

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.10.2.7 def radmc3dPy.model_template.get_gas_temperature (*grid* = None, *ppar* = None)

```
Function to calcualte/set the gas temperature

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the gas temperature in K
```

3.10.2.8 def radmc3dPy.model_template.get_velocity (*grid* = None, *ppar* = None)

```
Function to create the turbulent velocity field

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the turbulent velocity in cm/s
```

3.10.2.9 def radmc3dPy.model_template.get_vturb (*grid* = None, *ppar* = None)

```
Function to create the turbulent velocity field

INPUT:
------
    grid - An instance of the radmc3dGrid class containing the spatial and wavelength grid
    ppar - Dictionary containing all parameters of the model

OUTPUT:
-------
    returns the turbulent velocity in cm/s
```

3.11 radmc3dPy.model_test_scattering_1 Namespace Reference

Functions

- def get_desc
- def get_default_params
- def get_dust_density

3.11.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz, Kees Dullemond 2011,2012,2013

Original IDL model by Kees Dullemond, Python translation by Attila Juhasz
```

3.11.2 Function Documentation

3.11.2.1 def radmc3dPy.model_test_scattering_1.get_default_params ( )

```
Function to provide default parameter values

OUTPUT:
-------

Returns a list whose elements are also lists with three elements:
1) parameter name, 2) parameter value, 3) parameter description
All three elements should be strings. The string of the parameter
value will be directly written out to the parameter file if requested,
and the value of the string expression will be evaluated and be put
to radmc3dData.ppar. The third element contains the description of the
parameter which will be written in the comment field of the line when
a parameter file is written.
```

3.11.2.2 def radmc3dPy.model_test_scattering_1.get_desc ( )

```
Function to provide a brief description of the model
```

3.11.2.3 def radmc3dPy.model_test_scattering_1.get_dust_density (*grid* = `None`, *ppar* = `None`)

```
Function to create the dust density distribution

OUTPUT:
-------
    returns the volume density in g/cm^3
```

3.12 radmc3dPy.natconst Namespace Reference

Variables

- float **gg** 6.672e-8
- float **mp** 1.6726e-24
- float **me** 9.1095e-28
- float **kk** 1.3807e-16
- float **hh** 6.6262e-27
- float **ee** 4.8032e-10

- float **cc** 2.99792458e10
- float **st** 6.6524e-25
- float **ss** 5.6703e-5
- float **aa** 7.5657e-15
- float **muh2** 2.3000e0
- float **ev** 1.6022e-12
- float **kev** 1.6022e-9
- int **micr** 1
- int **km** 1
- int **angs** 1
- float **ls** 3.8525e33
- float **rs** 6.96e10
- float **ms** 1.99e33
- float **ts** 5.780e3
- float **au** 1.496e13
- float **pc** 3.08572e18
- float **mea** 5.9736e27
- float **rea** 6.375e8
- float **mmo** 7.347e25
- float **rmo** 1.738e8
- float **dmo** 3.844e10
- float **mju** 1.899e30
- float **rju** 7.1492e9
- float **dju** 7.78412e13
- float **year** 3.1536e7
- float **hour** 3.6000e3
- float **day** 8.6400e4

3.12.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz 2011,2012,2013

This sub-module contains natural constants in CGS units
(Translated from RADMC's IDL function problem_natconst.pro)
```

3.13 radmc3dPy.setup Namespace Reference

Functions

- def get_template_model
- def get_model_names
- def get_model_desc
- def problem_setup_dust
- def problem_setup_gas
- def write_radmc3d_inp
- def write_lines_inp

3.13.1 Detailed Description

```
PYTHON module for RADMC3D
(c) Attila Juhasz 2011,2012,2013

This sub-module functions to set up a RADMC3D model for dust and/or line simulations
For help on the syntax or functionality of each function see the help of the individual functions

FUNCTIONS:
----------
    get_model_desc()     - Returns the brief description of a model (if the model file contains a get_desc() f
    get_model_names()    - Returns the list of available models
    get_template_model() - Copy the template model file from the library directory (radmc3dPy) to the current
    problem_setup_dust() - Function to set up a dust model
    problem_setup_gas()  - Function to set up a line simulation
    write_lines_inp()    - Writes the lines.inp master command file for line simulations
    write_radmc3d_inp()  - Writes the radmc3d.inp master command file required for all RADMC3D runs
```

3.13.2 Function Documentation

3.13.2.1 def radmc3dPy.setup.get_model_desc (*model* = ' ')

```
Returns the brief description of the selected model
```

3.13.2.2 def radmc3dPy.setup.get_model_names ( )

```
Returns the name of the available models
```

3.13.2.3 def radmc3dPy.setup.get_template_model ( )

```
Create a copy of the template model file in the current working directory.
The PYTHONPATH environment variable is checked for the installation path of radmc3dPy and
the template file is copied from the first hit in the path list.
```

3.13.2.4 def radmc3dPy.setup.problem_setup_dust (*model* = ' ', *binary* = `True`, *write_dusttemp* = `False`, *kwargs*)

```
Function to set up a dust model for RADMC3D

INPUT:
------
    model : Name of the model that should be used to create the density structure.
            The file should be in a directory from where it can directly be imported
            (i.e. the directory should be in the PYTHON_PATH environment variable or
            it should be in the current working directory)
            and the file name should be 'model_xxx.py', where xxx stands for the string
            that should be specified in this variable

    binary : If True input files will be written in binary format, if False input files are
            written as formatted ascii text.

OPTIONS:
--------
    Any varible name in problem_params.inp can be used as a keyword argument.
    At first all variables are read from problem_params.in to a dictionary called ppar. Then
    if there is any keyword argument set in the call of problem_setup_dust the ppar dictionary
    is searched for this key. If found the value belonging to that key in the ppar dictionary
    is changed to the value of the keyword argument. If no such key is found then the dictionary
    is simply extended by the keyword argument. Finally the problem_params.inp file is updated
    with the new parameter values.


FILES WRITTEN DURING THE SETUP:
```

```
------------------------------
    dustopac.inp          - dust opacity master file
    wavelength_micron.inp - wavelength grid
    amr_grid.inp          - spatial grid
    stars.inp             - input radiation field
    dust_density.inp      - dust density distribution
    radmc3d.inp           - parameters for RADMC3D (e.g. Nr of photons to be used, scattering type, etc)

STEPS OF THE SETUP:
-------------------
    1) Create the spatial and frequency grid
    2) Create the master opacity file and calculate opacities with the Mie-code if necessary
    3) Set up the input radiation field (generate stars.inp)
    4) Calculate the dust density
    5) If specified;  calculatest the dust temperature (e.g. for gas simulations, or if it is taken from an
     external input (e.g. from another model))
    6) Write all output files
```

3.13.2.5 def radmc3dPy.setup.problem_setup_gas (*model* = ' ' , *fullsetup* = False, *binary* = True, *write_gastemp* = False, *kwargs*)

```
Function to set up a gas model for RADMC3D

INPUT:
------
    model : Name of the model that should be used to create the density structure
            the file should be in a directory from where it can directly be imported
            (i.e. the directory should be in the PYTHON_PATH environment variable, or
            it should be the current working directory)
            and the file name should be 'model_xxx.py', where xxx stands for the string
            that should be specified in this variable

    fullsetup : if False only the files related to the gas simulation is written out
                (i.e. no grid, stellar parameter file and radmc3d master command file is written)
                if True the spatial and wavelength grid as well as the input radiation field
                and the radmc3d master command file will be (over)written.

    binary : If True input files will be written in binary format, if False input files are
            written as formatted ascii text.

    write_gastemp: If True a separate gas_temperature.inp/gas_tempearture.binp file will be
            written under the condition that the model contains a function get_gas_temperature()

OPTIONS:
--------
    Any varible name in problem_params.inp can be used as a keyword argument.
    At first all variables are read from problem_params.in to a dictionary called ppar. Then
    if there is any keyword argument set in the call of problem_setup_gas the ppar dictionary
    is searched for such key. If found the value belonging to that key in the ppar dictionary
    is changed to the value of the keyword argument. If no such key is found then the dictionary
    is simply extended by the keyword argument. Finally the problem_params.inp file is updated
    with the new parameter values.


FILES WRITTEN DURING THE SETUP:
-------------------------------
    fullsetup = True
        amr_grid.inp          - spatial grid
        wavelength_micron.inp - wavelength grid
        stars.inp             - input radiation field
        radmc3d.inp           - parameters for RADMC3D (e.g. Nr of photons to be used, scattering type, etc)
        lines.inp             -  line mode master command file
        numberdens_xxx.inp    - number density of molecule/atomic species 'xxx'
        gas_velocity.inp      - Gas velocity
        microturbulence.inp   - The standard deviation of the Gaussian line profile caused by turbulent
                                broadening (doublecheck if it is really the standard deviation or a factor
                                of sqrt(2) less than that!)
        gas_temperature.inp   - Gas temperature (which may be different from the dust temperature). If
                                tgas_eq_tdust is set to zero in radmc3d.inp the gas temperature in this
                                file will be used instead of the dust temperature.
    fullsetup = False
```

```
lines.inp             -  line mode master command file
numberdens_xxx.inp    - number density of molecule/atomic species 'xxx'
gas_velocity.inp      - Gas velocity
microturbulence.inp   - The standard deviation of the Gaussian line profile caused by turbulent
                        broadening (doublecheck if it is really the standard deviation or a factor
                        of sqrt(2) less than that!)
gas_temperature.inp   - Gas temperature (which may be different from the dust temperature). If
                        tgas_eq_tdust is set to zero in radmc3d.inp the gas temperature in this
                        file will be used instead of the dust temperature.
```

3.13.2.6 def radmc3dPy.setup.write_lines_inp (*ppar =* None)

```
Function to write the lines.inp master command file for line simulation in RADMC3D

INPUT:
------
    ppar : dictionary containing all parameters of a RADMC3D run
```

3.13.2.7 def radmc3dPy.setup.write_radmc3d_inp (*modpar =* None)

```
Function to write the radmc3d.inp master command file for RADMC3D

INPUT:
------
    ppar : dictionary containing all parameters of a RADMC3D run
```

Chapter 4

Class Documentation

4.1 radmc3dPy.analyze.radmc3dData Class Reference

Public Member Functions

- def **__init__**
- def get_tau_1dust
- def get_tau
- def read_dustdens
- def read_dusttemp
- def read_gasvel
- def read_vturb
- def read_gasdens
- def read_gastemp
- def write_dustdens
- def write_dusttemp
- def write_gasdens
- def write_gastemp
- def write_gasvel
- def write_vturb
- def write_vtk
- def get_sigmadust
- def get_sigmagas

Public Attributes

- **grid**
- **rhodust**
- **dusttemp**
- **rhogas**
- **ndens_mol**
- **ndens_cp**
- **gasvel**
- **gastemp**
- **vturb**
- **taux**
- **tauy**
- **tauz**
- **sigmadust**
- **sigmagas**

4.1.1 Detailed Description

```
RADMC3D data class
    reading and writing dust density/temperature, gas density/temperature/velocity,
    generating a legacy vtk file for visualization

ATTRIBUTES:
-----------
    rhodust   -  Dust density in g/cm^3
    dusttemp  -  Dust temperature in K
    rhogas    -  Gas density in g/cm^3
    ndens_mol -  Number density of the molecule [molecule/cm^3]
    ndens_cp  -  Number density of the collisional partner [molecule/cm^3]
    gasvel    -  Gas velocity in cm/s
    gastemp   -  Gas temperature in K
    vturb     -  Mictroturbulence in cm/s
    taux      -  Optical depth along the x (cartesian) / r (cylindrical) / r (spherical) dimension
    tauy      -  Optical depth along the y (cartesian) / theta (cylindrical) / theta (spherical) dimension
    tauz      -  Optical depth along the z (cartesian) / z (cylindrical) / phi (spherical) dimension
    sigmadust -  Dust surface density in g/cm^2
    sigmagas  -  Gas surface density in molecule/cm^2 (or g/cm^2 depending on the dimension of rhogas)
```

4.1.2 Member Function Documentation

4.1.2.1 def radmc3dPy.analyze.radmc3dData.get_sigmadust (*self*, *idust* = 0)

```
Function to calculate dust surface density

OPTIONS:
------
    idust - index of the dust species for which the surface density should be calculated
    if omitted the calculated surface density will be the sum over all dust species
```

4.1.2.2 def radmc3dPy.analyze.radmc3dData.get_sigmagas (*self*)

```
Function to calculate gas surface density
This function uses radmc3dData.rhogas to calculate the surface density, thus the
unit of surface density depends on the unit of radmc3dData.rhogas (g/cm^2 or molecule/cm^2)
```

4.1.2.3 def radmc3dPy.analyze.radmc3dData.get_tau (*self*, *idust* = [], *axis* = 'xy', *wav* = 0., *kappa* = None, *usedkappa* = True)

```
Function to calculate the optical depth along any given combination of the axes

INPUT:
------
    idust : List of dust component indices whose optical depth should be calculated
    If multiple indices are set the total optical depth is calculated summing
    over all dust species in idust
    axis  - Name of the axis/axes along which the optical depth should be calculated
    (e.g. 'x' for the first dimension or 'xyz' for all three dimensions)
    wav   : Wavelength at which the optical depth should be calculated
    kappa : If set it should be a list of mass extinction coefficients at the desired wavelength
    The number of elements in the list should be equal to that in the idust keyword
```

4.1.2.4 def radmc3dPy.analyze.radmc3dData.get_tau_1dust (*self*, *idust* = 0, *axis* = '', *kappa* = 0.)

```
Function to calculate the optical depth of a single dust species along any given combination of the axes

INPUT:
------
    idust - Index of the dust species whose optical depth should be calculated
```

```
    axis  - Name of the axis/axes along which the optical depth should be calculated
    (e.g. 'x' for the first dimension or 'xyz' for all three dimensions)
    kappa - Mass extinction coefficients of the dust species at the desired wavelength

OUTPUT:
-------
    Returns a dictionary with the following keys;

    taux  - optical depth along the first dimension
    tauy  - optical depth along the second dimension

    (tauz is not yet implemented)
```

4.1.2.5 def radmc3dPy.analyze.radmc3dData.read_dustdens (*self*, *fname* = ' ', *binary* = True)

```
Function to read the dust density

OPTIONS:
--------
    fname - Name of the file that contains the dust density. If omitted 'dust_density.inp' is used
    (or if binary=True the 'dust_density.binp' is used).
    binary - If true the data will be read in binary format, otherwise the file format is ascii
```

4.1.2.6 def radmc3dPy.analyze.radmc3dData.read_dusttemp (*self*, *fname* = ' ', *binary* = True)

```
Function to read the dust temperature

OPTIONS:
--------
    fname - Name of the file that contains the dust temperature.
    If omitted 'dust_temperature.dat' (if binary=True 'dust_temperature.bdat')is used.
    binary - If true the data will be read in binary format, otherwise the file format is ascii
```

4.1.2.7 def radmc3dPy.analyze.radmc3dData.read_gasdens (*self*, *ispec* = ' ', *binary* = True)

```
Function to read the gas density

INPUT:
------
    ispec - File name extension of the 'numberdens_ispec.inp' (or if binary=True 'numberdens_ispec.binp') file


OPTIONS:
--------
    binary - If true the data will be read in binary format, otherwise the file format is ascii
```

4.1.2.8 def radmc3dPy.analyze.radmc3dData.read_gastemp (*self*, *fname* = ' ', *binary* = True)

```
Function to read the gas temperature

OPTIONS:
--------
    fname - Name of the file that contains the gas temperature. If omitted 'gas_temperature.inp'
    (or if binary=True 'gas_tempearture.binp') is used.
    binary - If true the data will be read in binary format, otherwise the file format is ascii
```

4.1.2.9 def radmc3dPy.analyze.radmc3dData.read_gasvel (*self*, *fname* = ' ', *binary* = True)

```
Function to read the gas velocity.
```

```
OPTIONS:
--------
    fname - Name of the file that contains the gas velocity
    If omitted 'gas_velocity.inp' (if binary=True 'gas_velocity.binp')is used.
    binary - If true the data will be read in binary format, otherwise the file format is ascii
```

4.1.2.10 def radmc3dPy.analyze.radmc3dData.read_vturb (*self*, *fname* = ' ' , *binary* = True)

```
Function to read the turbulent velocity field.

OPTIONS:
--------
    fname - Name of the file that contains the turbulent velocity field
    If omitted 'microturbulence.inp' (if binary=True 'microturbulence.binp') is used.
    binary - If true the data will be read in binary format, otherwise the file format is ascii
```

4.1.2.11 def radmc3dPy.analyze.radmc3dData.write_dustdens (*self*, *fname* = ' ' , *binary* = True)

```
Function to write the dust density

OPTIONS:
--------
    fname - Name of the file into which the dust density should be written. If omitted 'dust_density.inp' is u
    binary - If true the data will be written in binary format, otherwise the file format is ascii
```

4.1.2.12 def radmc3dPy.analyze.radmc3dData.write_dusttemp (*self*, *fname* = ' ' , *binary* = True)

```
Function to write the dust density

OPTIONS:
--------
    fname - Name of the file into which the dust density should be written. If omitted 'dust_density.inp' is u
    binary - If true the data will be written in binary format, otherwise the file format is ascii
```

4.1.2.13 def radmc3dPy.analyze.radmc3dData.write_gasdens (*self*, *fname* = ' ' , *ispec* = ' ' , *binary* = True)

```
Function to write the gas density

INPUT:
------
fname  - Name of the file into which the data will be written. If omitted "numberdens_xxx.inp" and
 "numberdens_xxx.binp" will be used for ascii and binary format, respectively (xxx is the name of the molecule
ispec  - File name extension of the 'numberdens_ispec.inp' (if binary=True 'numberdens_ispec.binp')
 file into which the gas density should be written
binary - If true the data will be written in binary format, otherwise the file format is ascii
```

4.1.2.14 def radmc3dPy.analyze.radmc3dData.write_gastemp (*self*, *fname* = ' ' , *binary* = True)

```
Function to write the gas temperature

OPTIONS:
--------
    fname - Name of the file into which the gas temperature should be written. If omitted
    'gas_temperature.inp' (if binary=True 'gas_tempearture.binp') is used.
    binary - If true the data will be written in binary format, otherwise the file format is ascii
```

4.1.2.15 def radmc3dPy.analyze.radmc3dData.write_gasvel (*self, fname* = ' ', *binary* = True)

```
Function to write the gas velocity

OPTIONS:
--------
    fname  - Name of the file into which the gas temperature should be written.
    If omitted 'gas_velocity.inp' (if binary=True 'gas_velocity.binp') is used.
    binary - If true the data will be written in binary format, otherwise the file format is ascii
```

4.1.2.16 def radmc3dPy.analyze.radmc3dData.write_vtk (*self, vtk_fname* = ' ', *ddens* = False, *dtemp* = False, *idust* = [0], *gdens* = False, *gvel* = False, *gtemp* = False)

```
Function to dump all physical variables to a legacy vtk file

INPUT:
------
    vtk_fname : name of the file to be written, if not specified 'radmc3d_data.vtk' will be used
    ddens     : if set to True the dust density will be written to the vtk file
    dtemp     : if set to True the dust temperature will be written to the vtk file
    idust     : a list of indices that specifies which dust component should be written
        if not set then the first dust species (zero index) will be used
    gdens     : if set to True the gas density will be written to the vtk file
    gtemp     : if set to True the gas temperature will be written to the vtk file
    gvel      : if set to True the gas velocity will be written to the vtk file
```

4.1.2.17 def radmc3dPy.analyze.radmc3dData.write_vturb (*self, fname* = ' ', *binary* = True)

```
Function to write the microturbulence file

OPTIONS:
--------
    fname - Name of the file into which the turubulent velocity field should be written.
    If omitted 'microturbulence.inp' (if binary=True 'microturbuulence.binp') is used.
    binary - If true the data will be written in binary format, otherwise the file format is ascii
```

4.2 radmc3dPy.analyze.radmc3dDustOpac Class Reference

Public Member Functions

- def **__init__**
- def readopac
- def makeopac
- def mixopac
- def read_masteropac
- def write_masteropac
- def run_makedust

Public Attributes

- **freq**
- **nwav**
- **nfreq**
- **kabs**
- **ksca**
- **phase_g**
- **ext**
- **idust**
- **therm**

Static Public Attributes

- tuple **grid** radmc3dGrid()
- **wav** grid.wav
- list **ext** [ ]
- tuple **rfile** open(ppar['lnk_fname'][idust], 'r')
- list **w** [ ]
- list **n** [ ]
- list **k** [ ]
- tuple **dum** rfile.readline()
- tuple **w** array(w, dtype=float)
- tuple **n** array(n, dtype=float)
- tuple **k** array(k, dtype=float)
- tuple **wfile** open('opt_const.dat', 'w')
- string **lnk_fname** 'opt_const.dat'
- list **mixnames** ['dustkappa_igsize_'+str(igs+1)+'.inp']
- list **mixspecs** [['dustkappa_idust_'+str(idust+1)+'_igsize_'+str(igs+1)+'.inp' for idust in range(len(ppar['lnk_-fname']))]]
- list **therm** [True for i in range(len(ext))]

4.2.1 Detailed Description

```
Dust opacity class

ATTRIBUTES:
-----------
    wav     - wavelength grid
    freq    - frequency grid
    nwav    - number of wavelengths
    kabs    - absorption coefficient per unit mass
    ksca    - scattering coefficient per unit mass
    phase_g - phase function
    ext     - if set it contains the file name extension of the duskappa_ext.Kappa file
    therm   - if False the dust grains are quantum-heated (default: True)
    idust   - index of the dust species in the dust density distribution array

    NOTE: Each attribute is a list with each element containing the corresponding data for
          a given dust species

METHODS:
--------
    readopac()            - Read the dust opacity files
    read_masteropac()     - Read the master opacity file
    write_masteropac()    - Write the master opacity file
    makeopac()            - Calculates opacities with the Mie-code that comes with RADMC-3D (using the run_makec
    run_makedust()        - Runs the Mie-code to calculate dust opacities
```

4.2.2 Member Function Documentation

4.2.2.1 def radmc3dPy.analyze.radmc3dDustOpac.makeopac (*self*, *ppar* = None, *wav* = None)

```
Function to create dust opacities for RADMC3D using MIE calculation

INPUT:
------
    ppar  - dictionary containing all parameter of the simulation

OPTIONS:
--------
    wav  - numpy.ndarray containing the wavelength grid on which the mass absorption coefficients should be ca
```

4.2.2.2 def radmc3dPy.analyze.radmc3dDustOpac.mixopac (*self*, *ppar* = None, *mixnames* = [], *mixspecs* = [], *mixabun* = [], *writefile* = True)

```
Function to mix opacities

INPUT:
------
    ppar     - A dictionary containing all parameters of the actual model setup
        If any keyword is set besides ppar, the value of the separate keyword
        will be taken instead of that in ppar. If mixname, mixspecs, and mixabun are all set
        ppar is completely omitted and not necessary to set when mixopac is called.
    mixnames  - Names of the files into which the mixed dust opacities will be written (not needed if writefil
    mixspecs  - Names of the files from which the dust opacities are read (not needed if readfile=False)
    mixabun   - Abundances of different dust species
    writefile - If False the mixed opacities will not be written out to files given in mixnames.
```

4.2.2.3 def radmc3dPy.analyze.radmc3dDustOpac.read_masteropac (*self*)

```
Function to read the master opacity file 'dustopac.inp'
it reads the dustkappa filename extensions (dustkappa_ext.inp) corresponding to dust species indices

OUTPUT:
-------
Returns a dictionary with the following keys:
    'ext'   - list of dustkappa file name extensions
    'therm' - a list of integers specifying whether the dust grain is thermal or quantum heated
      (0 - thermal, 1 - quantum heated)
```

4.2.2.4 def radmc3dPy.analyze.radmc3dDustOpac.readopac (*self*, *ext* = [''], *idust* = None)

```
Function to read the dust opacity files

INPUT:
------
    ext  : file name extension (file names should look like 'dustkappa_ext.inp')
    idust: index of the dust species in the master opacity file (dustopac.inp') - starts at 0
```

4.2.2.5 def radmc3dPy.analyze.radmc3dDustOpac.run_makedust (*self*, *freq* = None, *gmin* = None, *gmax* = None, *ngs* = None, *lnk_fname* = None, *gdens* = None)

```
Interface function to the F77 code makedust to calculate mass absorption
coefficients from the optical constants using Mie-theory

INPUT:
------
    freq       - numpy.ndarray containing the frequency grid on which the opacities should be calculated
    gmin       - minimum grain size
    gmax       - maximum grain size
    ngs        - number of grain sizes
    gdens      - density of the dust grain in g/cm^3
    lnk_faname - name of the file in which the optical constants are stored

OUTPUT:
-------
    result         - numpy.ndarray[nfreq,ngs] containing the resulting opacities

FILE OUTPUT:
------------
    dustopac_i.inp - Contains the dust opacities in radmc3d format
    dustopac.inp   - Master dust opacity file
```

4.2.2.6 def radmc3dPy.analyze.radmc3dDustOpac.write_masteropac (*self*, *ext* = None, *therm* = None)

```
Function to write the master opacity file 'dustopac.inp'

INPUT:
------
    ext : list of dustkappa file name extensions
    therm : list of integers specifying whether the dust grain is thermal or quantum heated
    (0-thermal, 1-quantum)
```

4.3 radmc3dPy.analyze.radmc3dGrid Class Reference

Public Member Functions

- def **__init__**
- def make_wav_grid
- def write_wav_grid
- def make_spatial_grid
- def write_spatial_grid
- def read_grid
- def get_cell_volume

Public Attributes

- **crd_sys**
- **act_dim**
- **nx**
- **ny**
- **nz**
- **nxi**
- **nyi**
- **nzi**
- **nwav**
- **nfreq**
- **x**
- **y**
- **z**
- xi

  This has to be done properly if ppar.has_key('xres_nlev'): ri_ext = array([self.xi[0], self.xi[ppar['xres_nspan']]]) for i in range(ppar['xres_nlev']): dum_ri = ri_ext[0] + (ri_ext[1]-ri_ext[0]) ∗ arange(ppar['xres_nstep']+1, dtype=float64) / float(ppar['xres_nstep']) print ri_ext[0:2]/au print dum_ri/au ri_ext_old = array(ri_ext) ri_ext = array(dum_ri) ri_ext = append(ri_ext,ri_ext_old[2:]) print ri_ext/au print '---------—'.
- **yi**
- **zi**
- **wav**
- **freq**

4.3.1 Detailed Description

```
Class for the spatial and frequency grid used by RADMC3D

ATTRIBUTES:
-----------
    crd_sys    - 'car'/'cyl'/'sph' coordinate system of the spatial grid
    act_dim    - A three element vector the i-th element is 1 if the i-th dimension is active, otherwize the i
    nx         - Number of grid points in the x (cartesian) / r (cylindrical) / r (spherical) dimension
```

```
    ny         - Number of grid points in the y (cartesian) / theta (cylindrical) / theta (spherical) dimensio
    nz         - Number of grid points in the z (cartesian) / z (cylindrical) / phi (spherical) dimension
    nxi        - Number of cell interfaces in the x (cartesian) / r (cylindrical) / r (spherical) dimension
    nyi        - Number of cell interfaces in the y (cartesian) / theta (cylindrical) / theta (spherical) dime
    nzi        - Number of cell interfaces in the z (cartesian) / z (cylindrical) / phi (spherical) dimension
    nwav       - Number of wavelengths in the wavelength grid
    freq       - Number of frequencies in the grid (equal to nwav)
    x          - Cell centered x (cartesian) / r (cylindrical) / r (spherical)  grid points
    y          - Cell centered y (cartesian) / theta (cylindrical) / theta (spherical)  grid points
    z          - Cell centered z (cartesian) / z (cylindrical) / phi (spherical)  grid points
    xi         - Cell interfaces in the x (cartesian) / r (cylindrical) / r (spherical)  dimension
    yi         - Cell interfaces in the y (cartesian) / theta (cylindrical) / theta (spherical)  dimension
    zi         - Cell interfaces in the z (cartesian) / z (cylindrical) / phi (spherical)  dimension
    wav        - Wavelengh  grid
    freq       - Frequency  grid

METHODS:
--------
```

4.3.2 Member Function Documentation

4.3.2.1 def radmc3dPy.analyze.radmc3dGrid.get_cell_volume (*self*)

```
Function to calculate the volume of grid cells
```

4.3.2.2 def radmc3dPy.analyze.radmc3dGrid.make_spatial_grid (*self*, *crd_sys* = None, *xbound* = None, *ybound* = None, *zbound* = None, *nxi* = None, *nyi* = None, *nzi* = None, *ppar* = None)

```
Function to create the spatial grid

INPUT:
------
    crd_sys     - 'car'/'sph'  Coordinate system of the spatial grid
    xbound      - List (with at least two elements) of boundaries for the grid along the first dimension
    ybound      - List (with at least two elements) of boundaries for the grid along the second dimension
    zbound      - List (with at least two elements) of boundaries for the grid along the third dimension
    nxi         - Number of grid points along the first dimension. List with len(xbound)-1 elements with
            nxi[i] being the number of grid points between xbound[i] and xbound[i+1]
    nyi         - Same as nxi but for the second dimension
    nzi         - Same as nxi but for the third dimension

OPTIONS:
--------
    ppar        - Dictionary containing all input parameters of the model (from the problem_params.inp file)
          if ppar is set all keyword arguments that are not set will be taken from this dictionary
```

4.3.2.3 def radmc3dPy.analyze.radmc3dGrid.make_wav_grid (*self*, *wbound* = None, *nw* = None, *ppar* = None)

```
Function to create a wavelength/frequency grid

INPUT:
------
    wbound : list of at least two elements containing the wavelength boundaries of the wavelength grid
    nw     : list of len(wbound)-1 elements containing the number of wavelengths between the bounds
     set by wbound
OPTIONS:
--------
    ppar   : parameter dictionary
```

4.3.2.4 def radmc3dPy.analyze.radmc3dGrid.read_grid (*self*, *fname* = ' ')

```
Function to read the spatial (amr_grid.inp) and frequency grid (wavelength_micron.inp).
```

```
OPTIONS:
--------
    fname - File name from which the spatial grid should be read. If omitted 'amr_grid.inp' will be used.
```

4.3.2.5 def radmc3dPy.analyze.radmc3dGrid.write_spatial_grid (*self*, *fname* = '')

```
Function to write the wavelength grid to a file (e.g. amr_grid.inp)

OPTIONS:
--------
    fname - File name into which the spatial grid should be written. If omitted 'amr_grid.inp' will be used.
```

4.3.2.6 def radmc3dPy.analyze.radmc3dGrid.write_wav_grid (*self*, *fname* = '')

```
Function to write the wavelength grid to a file (e.g. wavelength_micron.inp)

OPTIONS:
--------
    fname  - File name into which the wavelength grid should be written. If omitted 'wavelength_micron.inp' wi
```

4.4 radmc3dPy.image.radmc3dImage Class Reference

Public Member Functions

- def **__init__**
- def get_closure_phase
- def get_visibility
- def writefits
- def plot_momentmap
- def get_momentmap
- def readimage
- def imconv

Public Attributes

- **image**
- **imageJyppix**
- **x**
- **y**
- **nx**
- **ny**
- **sizepix_x**
- **sizepix_y**
- **nfreq**
- **freq**
- **nwav**
- **wav**

4.4.1 Detailed Description

```
RADMC3D image class

ATTRIBUTES:
-----------
    image       - The image as calculated by radmc3d (the values are intensities in erg/s/cm^2/Hz/ster)
    imageJyppix - The image with pixel units of Jy/pixel
    x           - x coordinate of the image [cm]
    y           - y coordinate of the image [cm]
    nx          - Number of pixels in the horizontal direction
    ny          - Number of pixels in the vertical direction
    sizepix_x   - Pixel size in the horizontal direction [cm]
    sizepix_y   - Pixel size in the vertical direction [cm]
    nfreq       - Number of frequencies in the image cube
    freq        - Frequency grid in the image cube
    nwav        - Number of wavelengths in the image cube (same as nfreq)
    wav         - Wavelength grid in the image cube
```

4.4.2 Member Function Documentation

4.4.2.1 def radmc3dPy.image.radmc3dImage.get_closure_phase (*self*, *bl* = None, *pa* = None, *dpc* = None)

```
Function to calculate clusure phases for a given model image for any arbitrary baseline triplet

INPUT:
------
    bl       - a list or Numpy array containing the length of projected baselines in meter(!)
    pa       - a list or Numpy array containing the position angles of projected baselines in degree(!)
    dpc      - distance of the source in parsec

    NOTE, bl and pa should either be an array with dimension [N,3] or if they are lists each element of
the list should be a list of length 3, since closure phases are calculated only for closed triangles

OUTPUT:
-------
    returns a dictionary with the following keys:

bl     - projected baseline in meter
pa     - position angle of the projected baseline in degree
nbl    - number of baselines
u      - spatial frequency along the x axis of the image
v      - spatial frequency along the v axis of the image
vis    - complex visibility at points (u,v)
amp    - correlation amplitude
phase  - Fourier phase
cp     - closure phase
wav    - wavelength
nwav   - number of wavelengths
```

4.4.2.2 def radmc3dPy.image.radmc3dImage.get_momentmap (*self*, *moment* = 0, *nu0* = 0, *wav0* = 0)

```
Function to calculate moment maps

INPUT:
------
    moment : moment of the channel maps to be calculated
    nu0    : rest frequency of the line in Hz
    wav0   : rest wavelength of the line in micron

OUTPUT:
-------
    map : Numpy array with the same dimension as the individual channel maps
```

4.4.2.3 def radmc3dPy.image.radmc3dImage.get_visibility (*self*, *bl* = None, *pa* = None, *dpc* = None)

```
Function to calculate visibilities for a given set of projected baselines and position angles
with the Discrete Fourier Transform

INPUT:
------
    bl       - a list or Numpy array containing the length of projected baselines in meter(!)
    pa       - a list or Numpy array containing the position angles of projected baselines in degree(!)
    dpc      - distance of the source in parsec

OUTPUT:
-------
    returns a dictionary with the following keys:

bl     - projected baseline in meter
pa     - position angle of the projected baseline in degree
nbl    - number of baselines
u      - spatial frequency along the x axis of the image
v      - spatial frequency along the v axis of the image
vis    - complex visibility at points (u,v)
amp    - correlation amplitude
phase  - phase
wav    - wavelength
nwav   - number of wavelengths
```

4.4.2.4 def radmc3dPy.image.radmc3dImage.imconv (*self*, *fwhm* = None, *pa* = None, *dpc* = None)

```
Function to convolve a radmc3d image with a two dimensional Gaussian psf

INPUT:
------
      fwhm    : A list of two numbers; the FWHM of the two dimensional psf along the two principal axes
            The unit is assumed to be arcsec if dpc keyword is set, otherwise the unit is pixel
      pa      : Position angle of the psf ellipse (counts from North counterclockwise)
      dpc     : Distance of the source in pc, if omitted the unit of FWHM is assumed to be pixel

OUTPUT:
-------
      result  : same
      'cimage': The convolved image with the psf (unit is erg/s/cm/cm/Hz/ster)
      'image' : The original unconvolved image (unit is erg/s/cm/cm/Hz/ster)
      'psf'   : Two dimensional psf
      'x'     : first coordinate axis of the psf/image
      'y'     : second coordinate axis of the psf/image
```

4.4.2.5 def radmc3dPy.image.radmc3dImage.plot_momentmap (*self*, *moment* = 0, *nu0* = 0, *wav0* = 0, *dpc* = 1., *au* = False, *arcsec* = False, *cmap* = None, *vclip* = None)

```
Function to plot moment maps

INPUT:
------
    moment : moment of the channel maps to be calculated
    nu0    : rest frequency of the line in Hz
    wav0   : rest wavelength of the line in micron
    dpc    : distance of the source in pc
    au     : If true displays the image with AU as the spatial axis unit
    arcsec : If true displays the image with arcsec as the spatial axis unit (dpc should also be set!)
    cmap   : matplotlib colormap
    vclip  : two element list / Numpy array containin the lower and upper limits for the values in the moment
      map to be displayed
```

4.4.2.6 def radmc3dPy.image.radmc3dImage.readimage (*self*, *fname* = None)

```
Function to read an image calculated by RADMC3D
```

```
INPUT:
------
 fname   : file name of the radmc3d output image (if omitted 'image.out' is used)
```

4.4.2.7 def radmc3dPy.image.radmc3dImage.writefits (*self*, *fname* = '', *dpc* = 1., *coord* = '03h10m05s -10d05m30s', *bandwidthmhz* = 2000.0, *casa* = False)

```
Function to write out a RADMC3D image data in fits format (CASA compatible)

INPUT:
------
 fname        : File name of the radmc3d output image (if omitted 'image.fits' is used)
 coord        : Image center coordinates
 bandwidthmhz : Bandwidth of the image in MHz (equivalent of the CDELT keyword in the fits header)
 casa         : If set to True a Stokes axis will be added to the image cube to make it compatible
        with the casa simulator
```

4.5 radmc3dPy.analyze.radmc3dPar Class Reference

Public Member Functions

- def **__init__**
- def readparams
- def add_par
- def load_defaults
- def write_parfile

Public Attributes

- **ppar**
- **pdesc**
- **pblock**
- **pvalstr**
- **pvarstr**

Static Public Attributes

- list **dumlist** [ ]
- string **dumline** '-'
- tuple **dumline** rfile.readline()
- **comment** False
- list **varlist** [ ]
- int **iline** 0
- list **ind** dumlist[iline]
- list **blockname** dumlist[iline]
- list **vlist** dumlist[iline]
- list **lbind** vlist[1]
- list **cind** vlist[1]
- **inBrokenLine** False
- list **expr** vlist[1]
- list **com** vlist[1]
- string **com** ' '
- int **iline2** 0

- list **dummy** dumlist[iline + iline2]
- tuple **cind2** dummy.find('#')
- tuple **lbind2** dummy.find('\\')
- **iline** iline+iline2
- tuple **glob** globals()
- tuple **loc** locals()
- tuple **val** eval(varlist[i][1], glob)

4.5.1 Detailed Description

```
Class for parameters in a RADMC-3D model

ATTRIBUTES:
-----------
    ppar   : Dictionary containing parameter values with parameter names as keys
    pdesc  : Disctionary containing parameter description (comments in the parameter file) with parameter name
    pblock : Dictionary containing the block names in the parameter file and parameter names as values
    pvalstr: Dictionary containing parameter values as strings with parameter names as keys
```

4.5.2 Member Function Documentation

4.5.2.1 def radmc3dPy.analyze.radmc3dPar.add_par (*self*, *parlist* = [])

```
Function to add parameter to the radmc3DPar parameter class
If the paramter is already defined its value will be modified

INPUT:
------
    parlist - If the parameter is already defined parlist should be a two element
      list 1) parameter name, 2) parameter expression/value as a string

      If the parameter is not yet defined parlist should be a four element
      list 1) parameter name, 2) parameter expression/value as a string
      3) Parameter description (= comment field in the parameter file)
```

4.5.2.2 def radmc3dPy.analyze.radmc3dPar.load_defaults (*self*, *model* = '' , *ppar* = {}, *reset* = True)

```
Function to fill up the classs attributes with default values

OPTIONS:
------
    model - Model name whose paraemters should also be loaded
    ppar - Dictionary containing parameter values as string and parameter names as keys
   Default values will be re-set to the values in this dictionary

    reset - If True the all class attributes will be re-initialized before
    the default values would be loaded. I.e. it will remove all entries
    from the dictionary that does not conain default values either in this
    function or in the optional ppar keyword argument
```

4.5.2.3 def radmc3dPy.analyze.radmc3dPar.readparams (*self*, *fname* = '')

```
Function to read a parameter file
The parameters in the files should follow the python syntax


INPUT:
------
    fname  : file name to be read (if omitted problem_params.inp is used)

OUTPUT:
```

```
-------
    Returns a dictionary with the parameter names as keys
```

4.5.2.4 def radmc3dPy.analyze.radmc3dPar.write_parfile (*self*, *fname* = *' '*)

```
Function to write a parameter file


INPUT:
------
    fname  : File name to be read (if omitted problem_params.inp is used)
```

4.6 radmc3dPy.analyze.radmc3dStars Class Reference

Public Member Functions

- def **__init__**
- def find_peak_starspec
- def read_starsinp
- def write_starsinp
- def get_stellar_spectrum

Public Attributes

- **mstar**
- **tstar**
- **rstar**
- **lstar**
- **nstar**
- **pstar**
- **wav**
- **freq**
- **fnu**
- **nwav**
- **nfreq**

4.6.1 Detailed Description

```
Class of the radiation sources (currently only stars)

ATTRIBUTES:
-----------
    mstar - List of stellar masses
    tstar - List of stellar effective temperatures
    rstar - List of stellar radii
    lstar - List of stellar luminosities
    nstar - Number of stars
    pstar - Locations (coordinates) of the stars
    wav   - Wavelength for the stellar spectrum
    freq  - Frequency for the stellar spectrum
    fnu   - Stellar spectrum (flux@1pc)
    nwav  - Number of wavelenghts in the stellar spectrum
    nfreq - Number of frequencies in the stellar spectrum
```

4.6.2 Member Function Documentation

4.6.2.1 def radmc3dPy.analyze.radmc3dStars.find_peak_starspec (*self*)

```
Function to  calculate the peak wavelength of the stellar spectrum

OUTPUT:
-------
    Returns the peak wavelength of the stellar spectrum in nu*Fnu for all
stars as a list
```

4.6.2.2 def radmc3dPy.analyze.radmc3dStars.get_stellar_spectrum (*self*, *tstar* = None, *rstar* = None, *lstar* = None, *nu* = None, *wav* = None)

```
Function to calculate a blackbody stellar spectrum

INPUT:
------
    tstar : Effective temperature of the star in [K]
    rstar : Radius of the star in [cm]
    lstar : Bolometric luminosity of the star [erg/s] (either rstar or lstar should be given)
    nu    : frequency grid on which the spectrum should be calculated [Hz]
    wav   : wavelength grid on which the spectrum should be calculated [micron]
```

4.6.2.3 def radmc3dPy.analyze.radmc3dStars.read_starsinp (*self*, *fname* = '')

```
Function to read the stellar data from the stars.inp file

OPTIONS:
--------
    fname - File name of the file that should be read (if omitted stars.inp will be used)
```

4.6.2.4 def radmc3dPy.analyze.radmc3dStars.write_starsinp (*self*, *wav* = None, *freq* = None, *pstar* = None, *tstar* = None)

```
Writes the stars.inp file

INPUT:
------
    wav   - Wavelength grid for the stellar spectrum
    freq  - Frequency grid for the stellar spectrum (either freq or wav should be set)
    pstar - List of the cartesian coordinates of the stars (each element of pstar should be a list of three el
    with the [x,y,z] coordinate of the individual stars)
    tstar - List containing the effective temperature of the stars
```

Index