MontAGN

Quick guide

MontAGN is fully written in `Python 2.7`
`ipython` is recommanded, if not consider using `matplotlib.pyplot.ion`

MontAGN - V 2.6

Manual V 2.0

MontAGN was written by : Lucas Grosset
Lâm Nguyen
Jan Orkisz

with the support of : Damien Gratadour
Didier Pelat
Daniel Rouan
Pierre Vermot

Manual written by : Lucas Grosset & Lâm Nguyen

Table des matières

1 Using MontAGN

1.1 Installing MontAGN

MontAGN is distributed as a python library. To use MontAGN :
— download the *.py files and put them into your `PYTHONPATH`
— create a directory for the simulation, it should contain
  — an "Input" directory where to put the dust, spectra and parameter files
  — an "Output" directory for output files
— in a ipython (or python) terminal, use `import montagn`

1.2 Basic commands

MontAGN can be run by calling the main function of the code :

In [1] : `montAGN()`

(note that there is no caps on the first letter to avoid confusion with the code's name).

It is required however to load the functions into the terminal, using `run`, `import` or `exec` on the file `'montagn.py'` :

In [1] : `import montagn`

In [2] : `montagn.montAGN()`

The function has no mandatory input parameter but can be use with many keywords as well as a parameter file. Its gives as an output a `model` class object, as defined in section 5.

1.3 Keywords

The keywords available on the main MontAGN function are listed here, following this organisation :

— **keyword** values available (`default value`)
  Explanations.

Keywords available for other useful function included in MontAGN will be described in the corresponding sections.

1.3.1 Main keywords

— **add** 0 or 1 (0)
Add (1) or not (0) the new data at the end of the output file (if it already exists). If `add=0` and the file exist, it will be replaced.

— **ask** 0 or 1 (1)
Enabled or disabled the interactive mode : ask the user (1) or not (0) the parameters for simulation. If `ask=0`, parameter file (by default) or predefined models will be used.

— **cluster** 0 or 1 (0)
`1` is mandatory when launched on clusters for management of files, directories and displays.

— **filename** 'string' (`'test'`)
prefix of the output files (between quotes ") for packets, dust densities and temperature (if `usethermal=1`).

— **model** model class object ( )
Keyword usable to give to MontAGN an already existing model class object (see 5). Allows for example to use a previously computed map temperature, or to avoid to compute again a density grid already existing. If not specified, MontAGN will create a new model class object, based on `ask` keyword.

— **nphot** positive integer ( )
number of photon packets to be launched. If not given, it will be asked to the user at the beginning of the simulation. Some typical execution time is between 0.1 and 1 s per packet (it depends on the optical depth).

— **paramfile** 'string' ( )
Name of the parameter file to be loaded, `ask` need to be set to 0.

— **path** 'string' ('')
Relative path where to locate `Input/paramfile` and to record output files.

— **usemodel** integer from list below (0)
This keyword is unused if `ask=1` and if any `paramfile` is specified.
0 uses parameters tunable in the file montagn_geometry.py
For optical depth tests models :
-1.XXxx creates a silicate benchmark of optical depth XX.xx
-2.XXxx creates an ortho graphites benchmark of optical depth XX.xx
-3.XXxx creates a para graphites benchmark of optical depth XX.xx
-4.XXxx creates an electron benchmark of optical depth XX.xx
-5.XXxx creates a pah benchmark of optical depth XX.xx
-6.XXxx creates a dust mixture benchmark of optical depth XX.xx

— **usethermal** 0 or 1 (1)
Enabled (1) or disabled (0) the thermal part of the MontAGN code. If enabled, the simulation will consider temperature adjustment of each cell as well as re-emission by dust, with a slower execution as a downside. If disabled, packet's energy are pondered by the albedo at each scattering to stand for the absorbed part of the packet's photons.

1.3.2 Model tuning keywords

— **force_wvl** float (in m) ( )
Impose if specified a unique wavelength to all the photon packets launched in the code. Please consider only using wavelength allowed by spectra and grains properties (between 20 nm and 2 mm typically).

— **nang** positive odd integer (999)
Indicate the angular resolution (number of steps) in the phase functions for scatterings. Have to be an odd number !

— **nscattmax** positive integer (10)
Maximum number of interaction that a packets will be allowed to undergo.

If it reaches this limit, it will stop propagating and will not be considered in the output.

— **nsize** positive integer (100)
Indicate the size resolution (number of steps) for grains sizes. As Q tables do not have a very high resolution, it is unnecessary to give a value higher than typically 100.

— **wvl_max** float (in m) (1e5)
Consider that photons above the indicated wavelength (in m) propagate without interaction (positive float).

1.3.3 Display keywords

— **cmap** 'string' ('jet')
Determine the colormap to be used for displays.

— **display** 0 or 1 (1)
Allows (1) or not (0) the display of the density maps, final images etc.. Disabling it is recommanded in parallel usages.

— **unit** 'AU', 'm' or 'pc' ('pc')
Distance unit that will be used in displays.

1.3.4 Advanced keywords

— **debug** [list containing any of the following strings] ( )
Print information about the critical steps during simulations. All information about the specified steps, to be chosen among 'T', 'Rsub', 'tau', 'scat', 'polar' and 'dust', will be displayed.

— **nsimu** positive integer (1)
Indicate the identity of the simulation in term of thread. Mostly used for parallel usages, should be set to 1 otherwise (else, some displays will be disabled).

— **plotinfo** [integer list] ( )
Plot graphs of phase function, albedo, x.... To be updated.

— **vectormode** 0 or 1 (0)
If `display=1`, create some additional final maps with polarisation vectors over-plotted.

— **verbose** 0 or 1 (0)
If `verbose=1`, gives extra informations on the computations.

1.4 Example of execution

1.4.1 Simple launch

Example of launch of MontAGN with a model of dust cocoon around a central star from a `ipython` session :

In [1] : `import montagn`

In [2] : `model=montagn.montAGN(ask=0,paramfile='test.txt',add=0, nphot=20000,filename='test')`

Launch a simulation with 20,000 photons packets, loading parameters from parameter file `'Input/test.txt'`. Outputs will be saved in `'Output/test_***.dat'` (see output section 3). Will replace the files if already existing.

1.4.2 Launch directly from terminal

It is also possible to launch the code directly from a terminal (`bash` or else) by executing file `'montagn_paral.py'` :

> `python montagn_paral.py --paramfile='test.txt' --nphot=20000 --filename='test'`

— Most of the main options described in section 1.3 are also available, see parallelised section 1.5 for more information.
— Launching the code this way automatically disabled the display.
— Filenames will be composed from the root filename, the thread number and the inclination angle : `'filename_001_***.dat'` for the first thread, etc..

1.4.3 Typical execution time

The typical execution time depend on the mode used :
— with temperature computation and dust re-emission : 300 packets <–> 1 min
— without temperature computation and dust re-emission : 100 packets <–> 1 min

1.5 Parallel

MontAGN can be launched in parallel. If so, the code will create as many threads as asked, creating as many times the usual number of output files. The threads are fully independent in term of seed, insuring to have independent results from one thread to any other.

To launch MontAGN in parallel, the following command should be entered on a terminal in the MontAGN directory, executing file `'montagn_paral.py'` :

```
> python montagn_paral.py --options
```

— Most of options are the same as the classical `montAGN()` ones. Some of them can not be used in this mode, displays is automatically disabled and a new options can be used (`--nlaunch`).
— As an usual function called from terminal, options should be called using '--', with the corresponding parameter if required.
— The new option `--nlaunch` define the number of thread that will be created and launched in parallel. If it is not given, only one thread will be created, leading to a classical MontAGN simulation.

1.5.1 Example

> `python montagn_paral.py --paramfile='model.txt' --nphot=20000 --filename='test' --nlaunch=2`

will launch two thread, of 20,000 photons packets each, loading parameters from parameter file `'Input/model.txt'`, and will save the results in files `'Output/test_001_***.dat'` and `'Output/test_002_***.dat'`(see Output section 3)

Available options :

- `--nlaunch=##`
- `--path=''`
- `--paramfile=''`
- `--filename=''`
- `--nphot=##`
- `--usethermal`
- `--nscattmax=##`
- `--cluster`
- `--force_wvl=##`
- `--usemodel= 1, -1, -2, ...`
- `--add`
- Other parameters are not available in this mode.

2 Parameter files

Parameter files are the main way of importing a model into MontAGN. Alternatively you can also load an already existing model object (using keyword `model=` when launching montAGN), create a new model from already existing parameter hard-coded in MontAGN (`usemodel=##` the number of your model) or define your model step by step by answering a sample of questions (by setting `ask=1`).

Parameter files should contain the following informations :

- at least one line of dust definition starting with the `dust` keyword :

```
dust dust_pop_name dust_type rgmin rgmax alpha rsub
```

- `dust_pop_name` is the reference name that will be used to design this population
- `dust_type` corresponds to the characteristics of dust. It has to be one of :
  - `silicates`
  - `graphites_ortho`
  - `graphites_para`
  - `electrons`
  - `pah_neutral`
  - `pah_ionised`
- `rgmin` is the minimal radius of grains (in m)
- `rgmax` is the maximal radius of grains (in m)
- `alpha` is the power law indice of the dust's radius distribution (RMN)

- at least one line of dust structure with the keyword corresponding to the chosen geometry (see the list below) :

```
geometry dut_pop_name param1 param2 param3 ...
```

- geometry correspond to the chosen geometry (see section geometries)
- `dust_pop_name` has to be among the defined dust populations and refers to the one to be consider for this geometry (except `'fractal'` that does not require a `dust_pop_name`)
- `param1,2,3...` correspond to the different parameters required by the geometry, different for each geometry

- at least one line of source definition following the `source` keyword :

```
source source_name spectre_filename luminosity emission_properties
```

- `source_name` will be the name that the simulation will refers to as the origin of the packets
- `spectre_filename` is the name of the file to be loaded in the `'Input'` directory, containing the spectra of the source

— `luminosity` is the source luminosity (in W)
— `emission_properties` corresponds to the emission and polarisation initial direction of the source. Is there is no particular emission properties, set it to `'default'`. Else, give this emission a name and you will need to define the properties as follow.

— the emission properties definition if required (if the `source` parameter is different from `'default'`) :

```
emission emission_properties em_dir polar_dir polarisation
```

— `emission_properties` corresponds to the emission properties name as given in `source`
— `em_dir` is the name of the emission direction properties. Is there is no particular emission direction, set it to `'default'`. Else, give this emission direction a name and you will need to define the properties in the following using the keyword `'emdir'`.
— `polar_dir` is the name of the polarisation direction at emission. Is there is no particular direction of polarisation, set it to `'default'`. Else, give this polarisation direction at emission a name and you will need to define the properties in the following using the keyword `'polardir'`.
— `polarisation` corresponds to initial polarisation of the source's name. Is there is no particular polarisation, set it to `'default'`. Else, give this polarisation a name and you will need to define the properties in the following using the keyword `'polar'`.

```
emdir em_dir theta phi
```

— `em_dir` is the name of the emission direction as defined in `emission`.
— `theta` the altitude emission angle $\in [-90, 90]$ (in degrees).
— `phi` the azimuthal emission angle $\in [0, 360]$ (in degrees).

```
polardir polar_dir phi_polar
```

— `polar_dir` is the name of the polarisation orientation as defined in `emission`.

— `phi_polar` the azimuthal polarisation orientation angle $\in [0, 180]$ (in degrees).

`polar polarisation Q U V`

— `polarisation` is the name of the polarisation at emission as defined in `emission`.
— `Q` the Stokes Q parameter $\in [0, 1]$.
— `U` the Stokes U parameter $\in [0, 1]$.
— `V` the Stokes V parameter $\in [0, 1]$.

— the four following lines associated with the following keywords, defining the model :

— the resolution of one cell of the map :
`res_map ##` (in m)
— the half-size of the simulation box :
`rmax_map ##` (in m)
— the aperture of the funnel (0 if there is no funnel) :
`af ##` (in degrees)
— the energy of the photon packets :
`enpaq ##` (in J) - This keyword is only relevant if one wants a simulation of a given equivalent time since it is only used to compute the emission equivalent duration.

2.1 Examples

Short example of parameter file :

```
dust silicates silicates 0.005e-6 0.25e-6 -3.5 0.05AU
spherepower [silicates] [5800.0] 0 1.0AU
res_map 1AU
rmax_map 25AU
source star spectre_NIRpeak.dat 3.846e26 'default'
af 0.
enpaq 3.846e26
```

This second example is a slightly more complex model, aimed at reproducing the interaction of a particular polarisation with a typical type of grains (related to dichroism) and therefore require specific emission properties :

```
dust graphites_ortho graphites_ortho 0.005e-6 0.25e-6 -3.5 0.00pc
cylinder [graphites_ortho] [0.1] 5.pc 5.pc
res_map 0.5pc
rmax_map 10pc
source AGN spectre_flat_NIR.dat 1e36 emission1
emission emission1 emdir1 polardir1 polar1
emdir emdir1 90. 0.
polardir polardir1 0.
polar polar1 0.999 0 0
af 0.
enpaq 1e36
```

2.2 Geometries

All geometries are organised in a similar way :

— they take as a first parameters the name(s) of the dust population(s) to be used to fill the given geometry (except `'fractal'`)

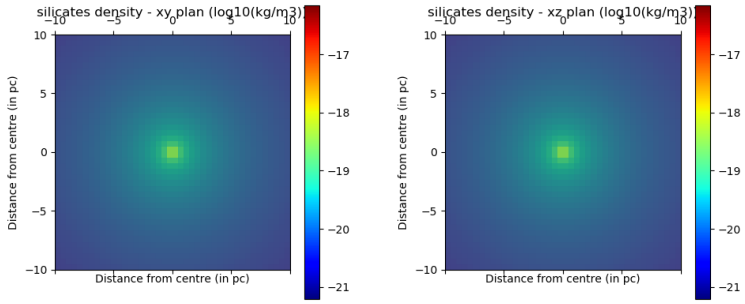
— the second one is the corresponding dust density(ies), in particles/m3 (except once again `'fractal'`) – Both these parameters have to be given between brackets [ ] even if only one dust population is used –

— the following arguments are dependent on the geometries and are given in the following with their description for each geometry :

2.2.1 BASIC GEOMETRIES

— `densturb` xxxx not usable yet

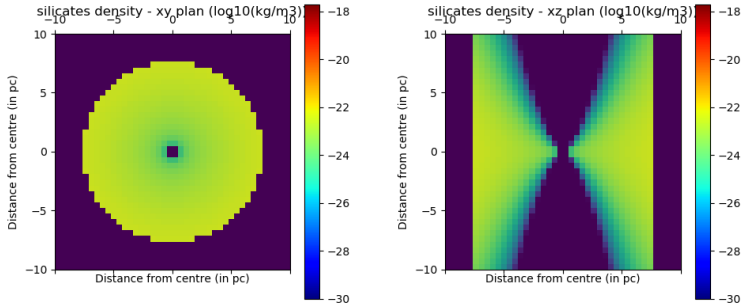
— `spherepower` 2 parameters



$$n_i(R) = n_{0,i} \left(\frac{R}{R_0}\right)^{\alpha}$$

- α : Radial power index
- R_0 : Radial typical profile size (in m)

— `gaussianprofile` 6 parameters

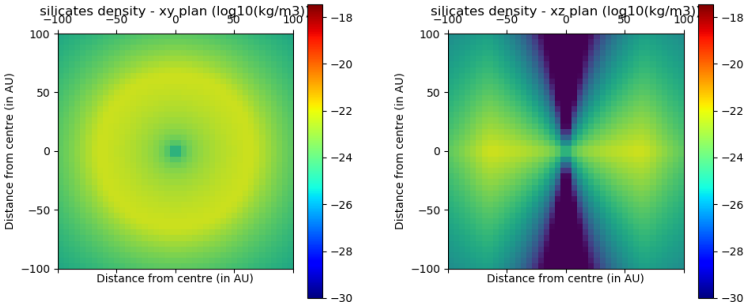


$$n_i(r,z) = \begin{cases} n_{0,i} \ \frac{3}{2} \left(\frac{r}{r_0}\right)^{\alpha} \ e^{\left(\frac{-z^{\gamma}}{2\left(h_0\left(\frac{r}{r_0}\right)^{\beta}\right)^{\gamma}}\right)}, & \text{if } r \leq r_{out} \\ 0, & \text{otherwise} \end{cases}$$

- γ : Power index
- β : Vertical power index
- α : Radial power index
- r_0 : Radial caracteristic size (in m)
- h_0 : Vertical caracteristic size (in m)

- r_{out} : Outer radius (in m)

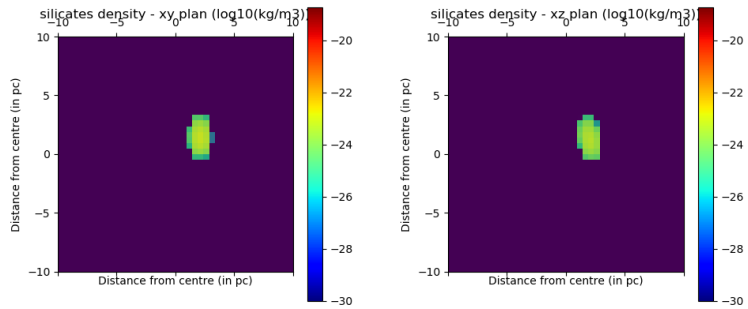
— `circumstellar_disk` 7 parameters



$$n_i(r,z) = n_{0,i}\ e^{-\left(\frac{|z|}{h_0\left(\frac{r}{r_0}\right)^{\beta}}\right)^{\gamma}} \left(\left(\frac{r}{r_c}\right)^{-2\ \alpha_{in}} + \left(\frac{r}{r_c}\right)^{-2\ \alpha_{out}}\right)^{-\frac{1}{2}}$$

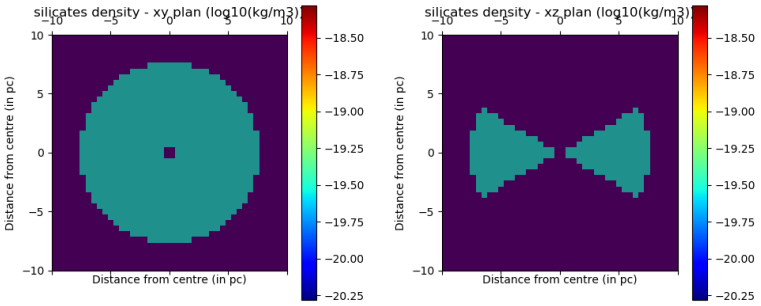
- r_0 : Typical radius where the grain density is normalised (in m)
- r_c : Critical radius for the power law slope change (in m)
- h_0 : Typical height at the typical radius (in m)
- α_{in} : Inner radial power index (>0)
- α_{out} : Outer radial power index (<0)
- β : Radial dependency of the vertical profile index (>0)
- γ : Vertical power index

— `cloud` 8 parameters



- x_0 : x position of the cloud
- y_0 : y position of the cloud
- z_0 : z position of the cloud
- a : Largest semi-axis
- b : Second semi-axis
- c : Third semi-axis
- θ_0 : First axis polar angle
- ϕ_0 : First axis azimutal angle

— `torus` 2 parameters



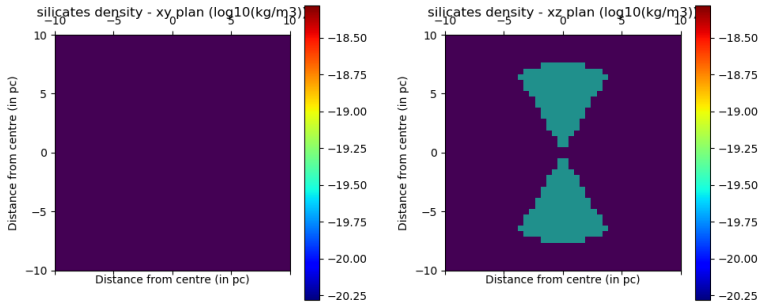
$$n_i(R,z) = \begin{cases} n_{0,i}, & \text{if } R < R_{disk} \ \& \ \mu(R,z) \leq \mu_\theta \\ \\ 0, & \text{otherwise} \end{cases}$$

with $\mu(R,z) = |\frac{z}{R}|$ and $\mu_\theta = \cos\left(\frac{\pi}{2} - \theta_0\right)$

- R_{disk} : Disc outer radius (in m)
- θ_0 : Disc half-angle (in degree)

WARNING this angle is defined as the half-angle of the torus and NOT as its half-opening angle

— `cone` 2 parameters

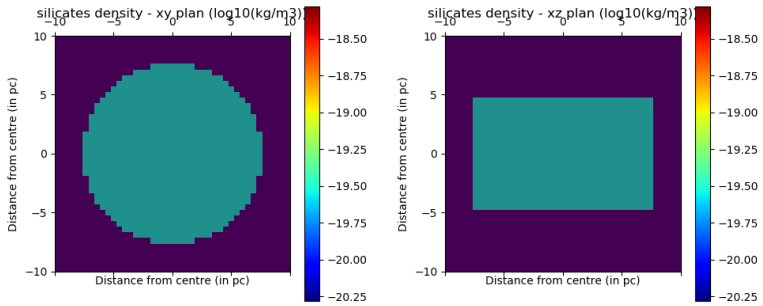


$$n_i(R,z) = \begin{cases} n_{0,i}, & \text{if } R < R_{out} \ \& \ \mu(R,z) > \mu_0 \\ \\ 0, & \text{otherwise} \end{cases}$$

with $\mu(R,z) = |\frac{z}{R}|$ and $\mu_0 = \cos(\theta_0)$

- R_{out} : Cone outer radius (in m)
- μ_0 : Cone half-aperture angle (in degree)

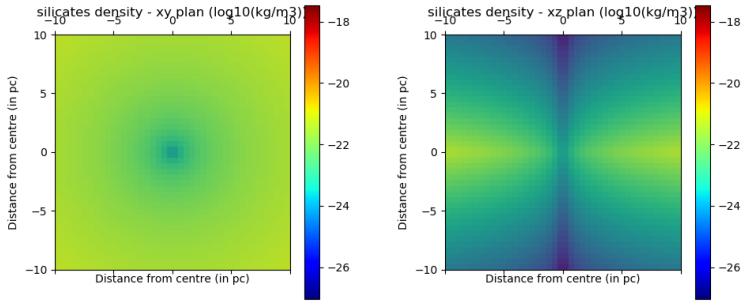
— `cylinder` 2 parameters



$$n_i(r,z) = \begin{cases} n_{0,i}, & \text{if } r < r_0 \ \& \ |z| < h_0 \\ \\ 0, & \text{otherwise} \end{cases}$$

- r_0 : Cylinder radius (in m)
- h_0 : Cylinder height (in m)

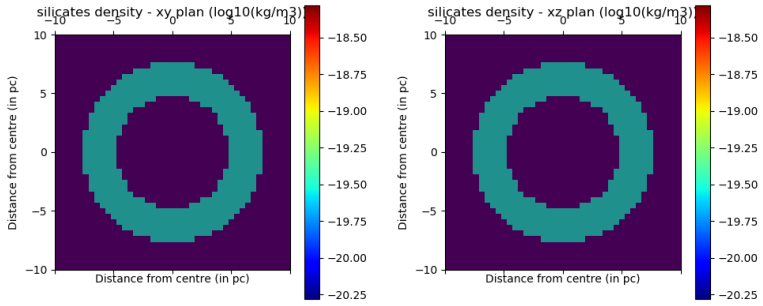
— `denspower` 3 parameters



$$n_i(r,z) = n_{0,i} \left(\frac{r}{r_0}\right)^{\alpha} e^{\frac{-|z|}{z_0}}$$

- α : Radial power index
- r_0 : Radial typical profile size (in m)
- z_0 : Vertical decay size (in m)

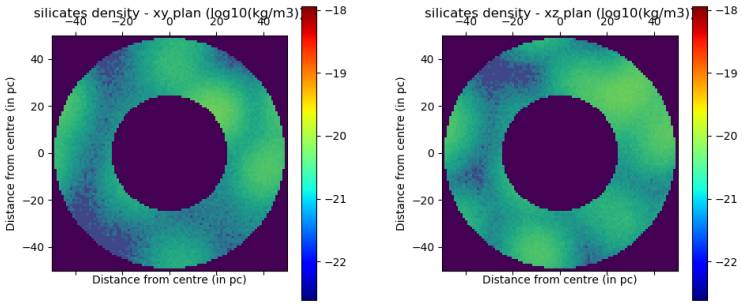
— `shell` 2 parameters



$$n_i(R) = \begin{cases} n_{0,i}, & \text{if } R \geq R_{in} \ \& \ R \leq R_{out} \\ 0, & \text{otherwise} \end{cases}$$

- R_{in} : Inner radius (in m)
- R_{out} : Outer radius (in m)

— `fractal` 7 (without any previous parameter for dust populations) parameters

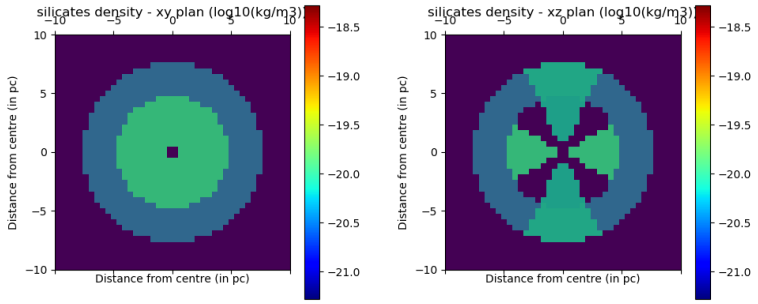


(a hierarchically clumped shell (Elmegreen 1997) of given grains based on the algorithm described in Mathis et al. (2002))

- f : Ratio of grains
- N : Number of new positions that, at each level, are substituted for each position in the previous level (N^H positions after H levels)
- D : Volume fractal dimension
- H : Number of hierarchical levels
- R_{in} : Inner radius
- R_{out} : Outer radius
- $M_{fractal}$: Total mass of the hierarchically clumped shell

2.2.2 COMPOSITE GEOMETRIES

— `AGN_simple` 8 parameters



composite of torus, shell and cone
$n_i(R, r, z) = n_{i,torus}(R, z, R_{disk}, \theta_{disk}) + n_{i,shell}(R, R_{disk}, R_{out}) + n_{i,cone}(R, z, R_{out}, \theta_0)$

- `[names]` : [name(s) of the dust population(s)]
- $[n_{i,torus}]$: [dust density(ies) in the torus] (in particles/m3)
- $[n_{i,shell}]$: [dust density(ies) in the shell] (in particles/m3)
- $[n_{i,cone}]$: [dust density(ies) in the cone] (in particles/m3)
- R_{disk} : Disc outer radius and shell inner radius (in m)
- R_{out} : Cone and shell outer radius (in m)
- θ_{disk} : Disc half-angle (in degree)
WARNING this angle is defined as the half-angle of the torus and NOT as its half-opening angle
- θ_0 : Cone half-aperture angle (in degree)

— `torus_Murakawa` 4 parameters

In progress In progress

$n_i(r, z, R) = n_{i,disk}(r, z) + n_{i,env}(R, z)$

$$\text{with } n_{i,disk}(r,z) = n_{0,i} \left(\frac{r}{R_{disk}}\right)^{-15/8} e^{-\frac{\pi}{4}\left(\frac{z}{R_{disk}\ H\ (r/R_{disk})^{9/8}}\right)^2},$$
$$\mu(R,z) = |\tfrac{z}{R}|$$
$$\text{and } n_{i,env}(R,z) = \frac{\dot{M}_{env}}{4\pi\sqrt{G\ M_s\ R^3}} \left(1+\frac{\mu}{\mu_0}\right)^{-1/2} \left(\frac{\mu}{\mu_0} + \frac{2\ \mu_0^2\ R_{disk}}{R}\right)^{-1}$$

- R_{disk} : Disc outer radius (in pc)
- H : Ratio of the disk height at the disk boundary to the disk outer radius
- $\dot{M}_{env}$: Envelope mass infall (in Msol/yr)
- M_s : Mass of the star (in Msol)

3 Output

The mains output are the files containing all the exited photons data. There are however some other files than can be created, all starting with the strings set to `path/Output/filename`. They will all be detailed here :

— `_xxx_phot.dat` files contain all the information about photons, at inclination angle `xxx` (in degrees). The list of all recorded informations is in section 3.1. These files are the one used to reconstruct images, SED...

— `_dust_xy.dat` file contain a density map of `dust` grains in the xy plane.

— `_dust_xz.dat` file contain a density map of `dust` grains in the xz plane.

Furthermore, if the re-emission (`usethermal=1`) is enabled, some additional files are created :

— `_T_update.dat` file contain the list of all the temperature update that occurred, with the corresponding cells.

— `_T_#.dat` file contain a sliced temperature map at a specific time (`#`= 0 for example is for the initial map, `#`= 3 corresponds to the final map).

3.1 Photons files

Table 1 list all the informations recorded in the `_xxx_phot.dat` files with the corresponding unit if needed :

TABLE 1 – Output parameters recorded in MontAGN simulations

Output parameter	Symbol	unit
Photon number	i	
Out inclination angle	θ	$^\circ$
Out azimuthal angle	ϕ	$^\circ$
Normalised Stokes U parameter	U	%
Normalised Stokes Q parameter	Q	%
Normalised Stokes V parameter	V	%
Out polarisation orientation angle	ϕ_{QU}	$^\circ$
Last interaction position x	x	pc
Last interaction position y	y	pc
Last interaction position z	z	pc
Number of interaction	n_{inter}	
Number of re-emission	n_{reem}	
Wavelength	λ	m
Label of the packet	label	
Energy of the packet	E	J
Emission time of the packet	δt	s
Total emission time (only if first packet)	Δt	s

4 Displays

The two main displaying functions included in MontAGN are `display_SED()` and `plot_image()`. Both these functions uses the `_xxx_phot.dat` files to construct their maps or graphs.

The last display function is `plot_Tupdate()` which allows to compute maps of useful informations about the temperature changes within the cells. It therefore uses and requires file `_T_update.dat`.

4.1 `display_SED`

`display_SED()` according to its name is a function allowing to show the SED of packets received under the selected restriction. It gives as an output two maps, a linear and a log scale ones.

In [2] : `display_SED(filename)`

4.2 `plot_image`

The second one, `plot_image()` is the most elaborated display function and compute maps according to requirements. It has one input parameter, `filename`, the name of the file in which load the data to display, and many options, detailed below. Note that if `thethaobs` is set to a particular angle while executing `montAGN`, the code will automatically call `plot_image()` for the corresponding inclination angle.

In [3] : `plot_image(filename)`

will read the data in file `'filename_phot.dat'`
Keywords :

— **outname** 'string' (`'test'`)
Root name of the images that will be recorded (if `rec=1`).

— **suffixe** 'string' (`''`)
Suffix to be placed at the end of the name of recorded files.

— **thetaobs** float within the range [0,180] ( )
Inclination angle selected for the display of maps.

— **dtheta** positive float < 180 (5)
Tolerance interval around the given angle `thetaobs` for photons packets selection. If `[]` is specified, all photons packets within the file will be used.

— **obj** 'AGN' or 'star' (`'AGN'`)
Allows to select have an idea of the final scale (UA ou pc)

— **path** 'string' (`''`)
Indicate the path to find the files to be read. Will read the file `path/filename`.

— **dat** 0 or 1 (1)
If set to `1`, automatically add `'.dat'` at the end of `'filename'`.

— **resimage** integer ( )

Resolution of the final images. Depends on keyword `resunit`. If `resunit` is not given, or set to `'pixel'`, `resimage` gives the number of pixels that will be used on one axis to display the maps (for example `51` for an image of `51x51`). Else, `resimage` determine the resolution of the image in `resunit`.

— **resunit** 'string' (`'pixel'`)
Gives the unit of the value entered with `resimage`.

— **diffn** positive integer or null ( )
If given, uses only packets with this value of number of scatterings.

— **ndiffmax** positive integer ( )
If given, define the maximum value of number of scattering for packets to be used.

— **cmap** 'string' (`'jet'`)
Determine the colormap to be used for displays.

— **enpaq** float (`3.86e26`)
Initial energy in photons packets. Used to compute the probability map.

— **coupe** 0 or 1 (`0`)
Allows to extract cuts of the central lines (`1`) and to compute histograms of polarisation within these cuts.

— **vectormod** 0 or 1 (`0`)
If `display=1`, create some additional final maps with polarisation vectors plotted above.

— **sym** 0, 1, or 2 (`0`)
Uses or not symmetries to compute the Q and U maps.
`0` cylindrical symmetry used
`1` symmetry according to equatorial plane is assumed as well as cylindrical symmetry
`2` symmetry of all four quadrants is assumed as well as cylindrical symmetry

— **rec** 0 ou 1 (0)
If set to 1, will record the displayed images, using 'saveformat' format and 'outname' name for files.

— **saveformat** 'pdf' or 'png' ('pdf')
Define the file format of recorded images.

4.3 plot_Tupdate

In [5] : plot_Tupdate('filename')

Displays the temperature maps extracted from 'filename_T_update.dat'
Keywords :
— **path** 'string' ('')
Indicate the path to find the files to be read. Will read the file path/filename.

— **dat** 0 or 1 (1)
If set to 1, automatically add '.dat' at the end of 'filename'.

— **unity** 'pc' or 'AU' ('pc')
Indicate the unit to be used on the axis.

— **rec** 0 ou 1 (0)
If set to 1, will record the displayed images, using 'saveformat' format and 'outname' name for files.

— **saveformat** 'pdf' or 'png' ('pdf')
Define the file format of recorded images.

— **size** positive integer (100)
Define the number of pixel to be used on each axis to display profiles and maps.

4.4 3D Display functions

4.4.1 `load3d_T`

In [6] : `load3d_T(data)`

Load the temperature map and return a 3D array.
Keywords :
— **data** 'string' ;or Model ('')
The file 'prefix_T_map.dat' recorded by save3d_T or the model returned by montAGN.

4.4.2 `load3d_rho`

In [7] : `load3d_rho(data,grain=None)`

Load the density map and return a 3D array.
Keywords :
— **data** 'string' or Model ('')
The file 'prefix_rho_dust_pop_name_map.dat' recorded by save3d_rho or the model returned by montAGN.

— **grain** 'string' (`required only if data is Model`)
The dust_pop_name to be loaded from the model returned by montAGN.

4.4.3 `plot3d`

In [8] : `plot3d(param,res,step=1,xmin=None,xmax=None,ymin=None,ymax=None,zmin=None,zmax=None,param_min=None,param_max=None,param_log=False,vmin=None,vmax=None,cmap=cm.jet,alpha=1,title='',unit='pc')`

Plot data in 3D.
Keywords :

— **param** array ([])
The data to be plotted in 3D, loaded using load3d_T or load3d_rho.

— **res** float ()
The resolution of the simulation grid.

— **step** integer ()
Indicate the step to sample param (for smaller data size and faster rendering).

— **xmin,xmax,ymin,ymax,zmin,zmax,param_min,param_max** float ()
Specify the ranges of x,y,z and param to plot.

— **param_log** boolean ()
If 'True', use logarithmic scale for param

— **vmin,vmax** scalar ()
Specify the range of param to normalize for the colormap.

— **cmap** Colormap ()
Specify the colormap for the plot.

— **alpha** scalar ()
The alpha blending value, between 0 (transparent) and 1 (opaque).

— **title** string ()
The title for the plot.

— **unit** string ()
The unit for the axes.

4.4.4 `tomo3d`

In [9] : `tomo3d(param,res,step=1,xmin=None,xmax=None,xstep=None,ymin=None,ymax=None,ystep=None,zmin=None,zmax=None,zstep=None,param_min`

=None,param_max=None,param_step=None,param_log=False,cmap=cm.jet,alpha=1,title='',unit='pc')

Sample data by x,y,z, data values and plot in 3D.
Keywords :
similar to plot3d, with additional parameters :

— **xstep, ystep, zstep, param_step** float ()
Indicate the steps to sample x,y,z and param.

5 MontAGN class objects

MontAGN has a class object called `model` which keeps all the important parameters about the simulation. It is therefore possible to have access to these parameters, at the end of a simulation or before its start. A `model` object contains :

— **sources** ()
list of sources

— **af** ()
aperture of the funnel

— **graininfo** ()
structure containing information about dust species

— **Rsub** ()
list of sublimation radii

— **energy** ()
energy of a packet

— **Dt** (0)
time length if the simulation

— **ndiffmax** (50)
maximum number of scattering allowed

— **usemodel** ()
indices of model to be used

— **thetaobs** ([90])
inclination of observation, for final plotting

— **thetarec** ([0,45,90,135,180])
list of recorded angles (not used any more)

— **dthetaobs** (5)
angle interval for registration into files

— **usethermal** (1)
whether use or not the re-emission

— **Temp_mode** ()
temperature method used

— **nRsub_update** (0)
whether use or not the sublimation radius evolution

— **map.res** ()
resolution of the grid

— **map.Rmax** ()
size of the grid

— **map.N** (int(Rmax/res)+1)
number of cells along an axis (the grid is $8 \times N^3$)

— **map.grid** ([])
grid containing all dust densities as well as temperatures

— **map.dust** ()
list of dust distribution used