radmc3dPy v0.22

Attila Juhász

module: setup

get_model_desc() - Returns the brief description of the model

get_model_names() - Returns the list of available models

problem_setup_dust() - Creats a dust continuum model setup

problem_setup_gas() - Creates a gas model setup

module: analyze

read_data() - Reads variables (e.g. dust density, gas velocity, etc)

read_grid() - Reads the spatial and frequency grid

readopac() - Reads the dust opacities

readparams() - Reads the parameter file (problem_params.inp)

write_default_parfile() - Writes the default parameters for a model

radmc3dData.get_sigmadust() - Calculates the dust surface density in g/cm^2

radmc3dData.get_sigmagas() - Calculates the gas surface density in molecule/cm^2

radmc3dData.get_tau() - Calculates the continuum optical depth

radmc3dData.write_vtk() - Writes variables to a VTK format for visualisation with e.g. Paraview

module: image

makeimage() - Calculates an image with RADMC-3D (both dust continuum and channel maps)

plotimage() - Plot the image / channel map

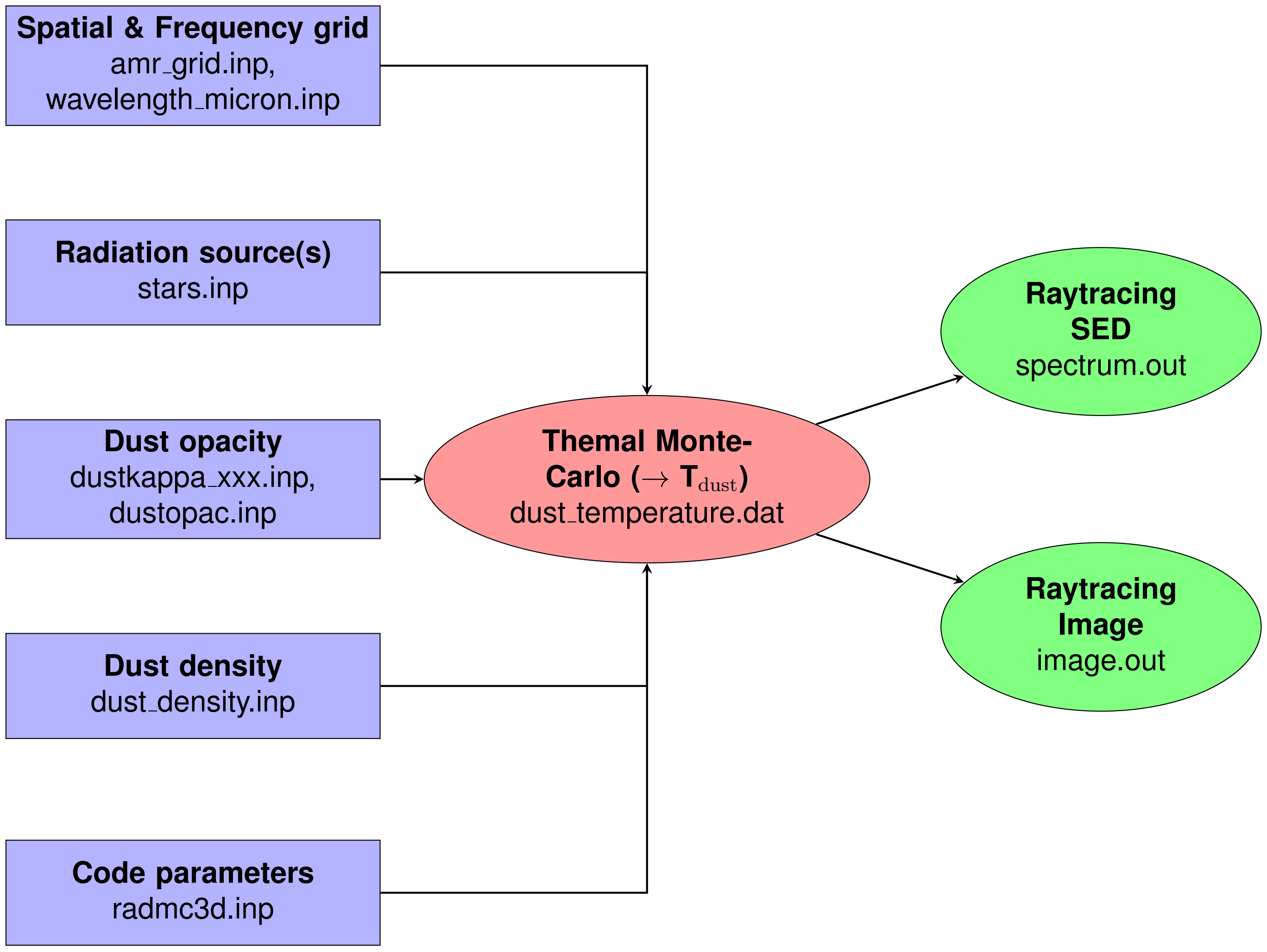
readimage() - Reads the image

radmc3dImage.imconv() - Convolve the image with a Gaussian beam

radmc3dImage.plot_momentmap() - Plots moment map for a 3d image cube

radmc3dImage.write_casafits() - Writes the image to a FITS file which is compatible with CASA

Dust continuum model



radmc3dPy commands

1 Import `radmc3dPy` .

```
>>> import radmc3dPy
```

2 Check which models are available:

```
>>> radmc3dPy.setup.get_model_names()
['ppdisk']
```

3 Create a parameter file with the default values

```
>>> radmc3dPy.analyze.write_default_parfile('ppdisk')
```

4 Exit python and open the created 'problem_params.inp' file with a text editor and change the parameters if needed. Create all necessary imput files.

```
>>>radmc3dPy.setup.problem_setup_dust('ppdisk')
```

5 Then run RADMC-3D from the shell with the Monte-Carlo simulation to calculate the dust temperature.

```
$>radmc3d mctherm
```

6 After the thermal Monte-Carlo run has finished we can make an image.

```
$>radmc3d image npix 400 sizeau 200 lambda 880.0 incl 45.  phi 0.  posang 43.
```

7 After RADMC-3D finished we can go back to python and read the image and plot it.

```
>>>imag=radmc3dPy.image.readimage()
>>>radmc3dPy.image.plotimage()
```

8 Now we can also calculate the SED. For that we can either use the wavelength grid we used of the thermal Monte-Carlo simulation, but we can also specify our own wavelength grid. For this latter we need to create a file called `camera_wavelength_micron.inp`. The file should be a simple formatted ascii text file. The first line should contain the number of wavelengths in the file, while the the following lines should contain the number of wavelengths in a single column format. E.g.:

```
5
0.5
3.6
9.7
70.
1300.
```

SEDs can be calculated with the following command

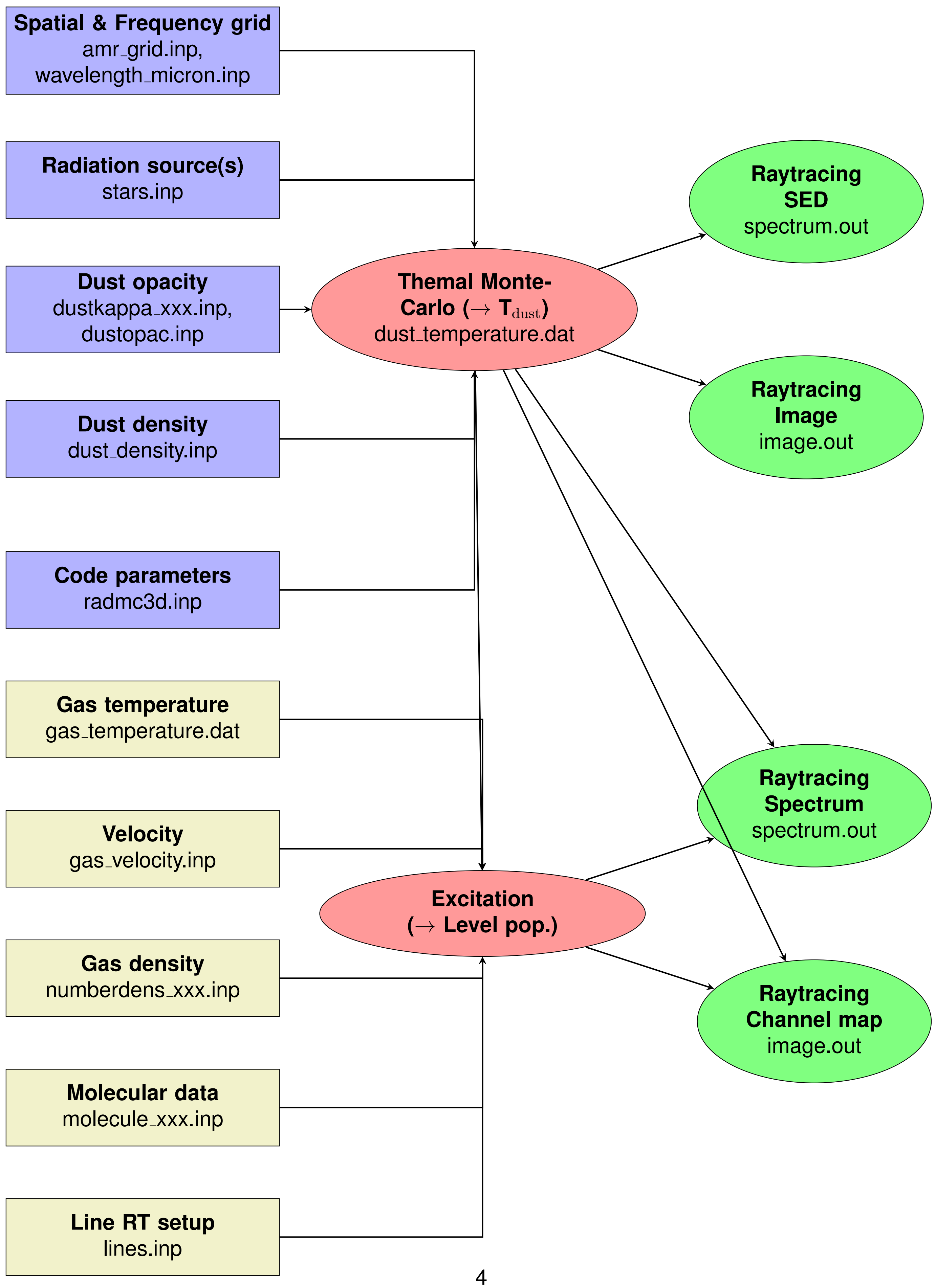
```
$>radmc3d sed incl 45.0 phi 0.0 loadlambda
```

In this way only the dust is taken into account the gas is neglected. If we wish to include gas emission/absorption we should use the 'spectrum' option instead of 'sed'. The loadlambda keyword at the end of the command tells RADMC-3D that it should calculate the SED for the wavelength in the camera_wavelength_micron.inp file.

```
$>radmc3d spectrum incl 45.0 phi 0.0 loadlambda
```

The result is a file called `spectrum.out` that contains the wavelength and flux in a two column ascii format.

Gas model



Spatial & Frequency grid
amr_grid.inp,
wavelength_micron.inp
Radiation source(s)
stars.inp
Dust opacity
dustkappa_xxx.inp,
dustopac.inp
Dust density
dust_density.inp
Code parameters
radmc3d.inp
Gas temperature
gas_temperature.dat
Velocity
gas_velocity.inp
Gas density
numberdens_xxx.inp
Molecular data
molecule_xxx.inp
Line RT setup
lines.inp
Themal Monte-Carlo (→ T$_{dust}$)
dust_temperature.dat
Excitation
(→ Level pop.)
Raytracing
SED
spectrum.out
Raytracing
Image
image.out
Raytracing
Spectrum
spectrum.out
Raytracing
Channel map
image.out

radmc3dPy commands

1 Import `radmc3dPy` .

```
>>> import radmc3dPy
```

2 Check which models are available:

```
>>> radmc3dPy.setup.get_model_names()
['ppdisk']
```

3 Create a parameter file with the default values

```
>>> radmc3dPy.analyze.write_default_parfile('ppdisk')
```

4 Exit python and open the created 'problem_params.inp' file with a text editor and change the parameters if needed. Create all necessary imput files.

```
>>>radmc3dPy.setup.problem_setup_gas('ppdisk')
```

5 This time we can skip the thermal Monte-Carlo simulation and go directly to raytracing, calculating images and spectra.

```
$>radmc3d image npix 400 sizeau 200 incl 45.  phi 0.  posang 43.  iline 3 vkms 1.0
```

6 This command calculates a single channel map at

```
>>>imag=radmc3dPy.image.readimage()
>>>radmc3dPy.image.plotimage()
```

7 Now we can also calculate the SED. For that we can either use the wavelength grid we used of the thermal Monte-Carlo simulation, but we can also specify our own wavelength grid. For this latter we need to create a file called `camera_wavelength_micron.inp`. The file should be a simple formatted ascii text file. The first line should contain the number of wavelengths in the file, while the the following lines should contain the number of wavelengths in a single column format. E.g.:

```
5
0.5
3.6
9.7
70.
1300.
```

SEDs can be calculated with the following command

```
$>radmc3d sed incl 45.0 phi 0.0 loadlambda
```

In this way only the dust is taken into account the gas is neglected. If we wish to include gas emission/absorption we should use the 'spectrum' option instead of 'sed'. The loadlambda keyword at the end of the command tells RADMC-3D that it should calculate the SED for the wavelength in the camera_wavelength_micron.inp file.

```
$>radmc3d spectrum incl 45.0 phi 0.0 loadlambda
```

The result is a file called `spectrum.out` that contains the wavelength and flux in a two column ascii format.