radmc3dPy v0.25

Attila Juhász

module: setup

getModelDesc() - Returns the brief description of the model

getModelNames() - Returns the list of available models

problemSetupDust() - Creats a dust continuum model setup

problemSetupGas() - Creates a gas model setup

module: analyze

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

readGrid() - Reads the spatial and frequency grid

readOpac() - Reads the dust opacities

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

writeDefaultParfile() - Writes the default parameters for a model

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

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

radmc3dData.getTau() - Calculates the continuum optical depth

radmc3dData.writeVTK() - 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.plotMomentmap() - Plots moment map for a 3d image cube

radmc3dImage.writeFits() - Writes the image to a FITS file with CASA compatible header

Dust continuum model

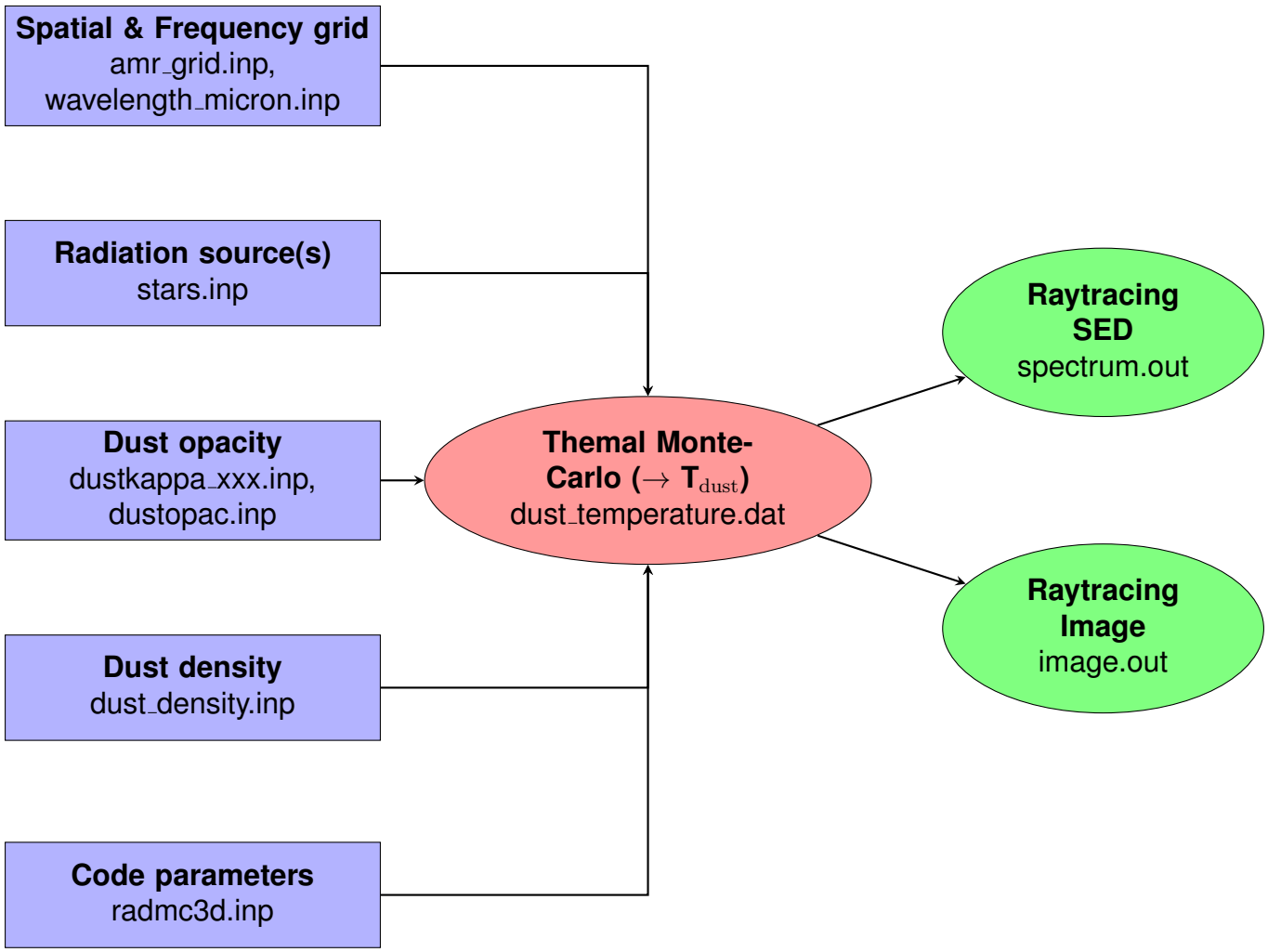

Figure 1: Structure of a dust continuum model. Inputs are marked with blue, intermediate calculation, data products are in red while green marks the output of the simulation.

radmc3dPy commands

First let us create a directory for our model. Then go to this directory and start python.

1 Import `radmc3dPy` .

```
>>> import radmc3dPy
```

2 Check which models are available:

```
>>> radmc3dPy.setup.getModelNames()
['lines_nlte_lvg_1d_1', 'ppdisk', 'simple_1', 'spher1d_1', 'spher2d_1',
'test_scattering_1']
```

3 Create a parameter file with the default values

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

To change the parameters of the model the two most straightforward possibilities are the following:

a) Open the created 'problem_params.inp' file with a text editor and change the parameters if needed.

b) When in Step 4. the 'problemSetupDust()' method is called one can add keyword arguments with the parameter names, e.g.:

```
>>>radmc3dPy.setup.problemSetupDust('ppdisk', mdisk='0.01*ms])
```

The 'problemSetupDust()' method does the following in this case: Reads the problem_params.inp file. Then overwrites the value of the mdisk parameter and uses that the new value afterwards. It also re-writes the problem_params.inp file with the new values.

NOTE: If the value of the keyword argument is given as a string it will be written as a string unchanged to the 'problem_params.inp' file but will be interpreted and converted to double/float/int within the setup script. I.e. if mdisk='0.01*ms' is given as a keyword argument in the call of problemSetupDust() then in the problem_params.inp file it will appear in the exact same way: mdisk='0.01*ms'. However, if the keyword argument is given as mdisk=0.01*ms the problem_params.inp will contain mdisk = 1.9900000e+31.

4 Set up the model and create all necessary input files.

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

Then we need to copy the dust opacity file called 'dustkappa_silicate.inp' from the python_examples/dataf directory within the distribution root directory to the current model directory.

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

```
$>radmc3d mctherm
```

Alternatively we can also make a system call from within Python, e.g.:

```
>>>import os
>>>os.system('radmc3d mctherm')
```

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

```
$>radmc3dPy.image.makeImage(npix=400, sizeau=200, wav=880., incl=45, posang=43.)
```

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

```
>>>imag=radmc3dPy.image.readImage()
>>>radmc3dPy.image.plotImage(imag, arcsec=True, dpc=140., log=True, maxlog=5)
```

Here 'arcsec=True' sets the image axes to arc second that also requires the knowledge of the distance, which is set in parsec by the 'dpc=140.' keyword. The 'log=True', sets logarithmic stretch of the image. The 'maxlog=5' sets a clip of the displayed image by 10^{-5} below its maximum value, i.e. the displayed image values will be between max(image) and max(image)*10^{-5}.

8 We can also convolve the image with an arbitrary elliptical gaussian Gaussian beam

```
>>>conv_imag = imag.imConv(fwhm=[0.05, 0.1], pa=40., dpc=140.)
```

The fwhm of the Gaussian beam should be in arcsec, the positing angle of the major axis of the beam ellipse should be in degrees, and the distance to the source in pc (dpc keyword) should be given.

9 We can also write the image into a fits file:

```
>>>imag.writeFits(fname='image.fits', dpc=140., coord='03h0m0s -29d0m0s')
```

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 Follow the instructions for the dust models from Step 1. to 5.

6 Create the necessary input files for the gas simulations:

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

7 Calculate a channel map at a single frequency/wavelength

```
$>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()
```