OpTool User Guide

Carsten Dominik, Michiel Min, and Ryo Tazaki

May 2021, version 1.9.4

1. Introduction

This tool produces complex dust particle opacities right from the command line. It is derived from Michiel Min's DHS OpacityTool and also implements Ryo Tazaki's MMF theory for highly porous aggregates.

1.1. Capabilities

- stand-alone tool, fully command line driven, no input files need to be edited
- full scattering matrix output in several formats, including for RADMC-3D
- combining materials through mixing into a complex grain with porosity
- *built-in*: a curated collection of materials for applications in astronomy
- external refractive index data can be used just as easily
- computational methods: (i) **DHS (Distribution of Hollow Spheres)** for *irregular grains* and *low-porosity* aggregates. Standard **Mie theory** for *perfect spheres* is available as a limiting case. (ii) **MMF (Modified Mean Field)** theory for *high-porosity/fractal aggregates.* (iii) **CDE** approximation in the Rayleigh limit.
- `Python` interface module for plotting and post-processing

1.2. Terms of use

`optool` is distributed under the MIT license and can be used, changed and redistributed freely. But we do ask you to cite the code, and also the papers relevant to your particular application, when using `optool`. The references are listed below and the corresponding BibTeX entries are available in the file `optool.bib`. `optool` is hosted on github at `https://github.com/cdominik/optool`

- **optool:** Dominik, C., Min, M. & Tazaki, R. 2021, Optool, 1.9, Astrophysics Source Code Library, ascl:2104.010
- **DHS model for irregular grains:** Min, M. et al. 2005, A&A, 432, 909
- **MMF model for aggregates:** Tazaki, R. & Tanaka,H. 2018, ApJ 860, 79
- **DIANA standard Opacities:** Woitke, P. et al. 2016, A&A 586, 103
- **Third party software:** Toon, O. et al. 1981, Applied Optics 20, 3657
- References to refractive index data used in your particular application.

2. Examples

A simple grain made only of the default pyroxene, for the default grain size distribution ($a^{-3.5}$ powerlaw from 0.05 to 3000μm), on the default wavelength grid (0.05μm to 1cm).

```
optool pyr
```

Include the scattering matrix in the produced output

```
optool pyr -s
```

Reproduce the DIANA standard dust model, using a specific pyroxene (70% Mg) and carbon, in a mass ratio 0.87/0.13, and with a porosity of 25%.

```
optool pyr-mg70 0.87  c 0.13  -p 0.25
```

List the built-in materials

```
optool -c
```

Add a water ice mantle (built-in data from Warren+08) that is 20% of the core mass

```
optool pyr-mg70 0.87  c 0.13  -m h2o-w 0.2  -p 0.25
```

Like the previous example, but use ice refractive index data from a separate file.

```
optool pyr-mg70 0.87  c 0.13  -p 0.25  -m data/ice_hudgins.dat 0.2
```

Pure water ice grains in a narrow size distribution from 1 to 3 microns, with 15 sample sizes following an $f(a) \propto a^{-2.5}$ powerlaw size distribution. Also, restrict the wavelength range to 10-100μm, and turn off DHS to get perfect spheres (Mie).

```
optool h2o  -a 1 3 2.5 15  -l 10 100 -mie
```

For silicon carbide, compute the opacity of a single grain size (2.5μm) at λ=8.9μm.

```
optool -a 2.5 -l 8.9 sic
```

Represent the default dust model (DIANA, you also get this when you do not give any materials at all) in 42 grain sizes, and produce input files for RADMC-3D, one for each grain size, with full scattering matrix, chopping 3 degrees from the scattering peak.

```
optool -na 42 -d -s -radmc -chop 3
```

Use MMF to compute the opacities of dust aggregates made of pyroxene monomers. Use a monomer radius of 0.3 μm to construct aggregates with compact-volume radii between 10 and 30 μm, and a filling factor of 5%

```
optool pyr  -a 10 30  -mmf 0.3 0.05
```

As before, but fix the fractal dimension $D_f = 1.9$ instead of setting the filling factor

```
optool pyr  -a 10 30  -mmf 0.3 1.9
```

3. Installation

You can download, compile, and install `optool` with these simple steps, using the freely available GNU FORTRAN compiler `gfortran`.

```
git clone https://github.com/cdominik/optool.git
cd optool
make multi=true                      # compile with multicore support
make install bindir=~/bin/           # copy the binaries to binary path
pip install -e .                     # install the python module
```

In the compilation step, use `multi=true` to add multicore support (recommended!), `ifort=true` to use the Intel fortran compiler instead of `gfortran`, and `fits=true` to add support for writing FITS files[1]. The executable is called `optool`. The `make install` step copies it and also `optool2tex` and `optool-complete` into `bindir`.
If compiling the code is a problem, you can also find binaries for Mac and Linux at my homepage. For shell command line completion support, check the file `optool-complete`.

4. Command line arguments

`-h [OPT]`
: Show command line options or specific information about option `-OPT`.

4.1. Grain composition

If no composition is specified, the default is **-c pyr 0.87 -c c 0.13 -p 0.25**.

`-c` List available built-in materials (the keys for the **-c** and **-m** options).

`[-c] KEY-or-FILE [MFRAC]`
: Specify a material to include in the grain. `KEYorFILE` can be the key for a builtin material, or the path to an `lnk` file. `MFRAC` is the *mass* fraction (default 1.0) of the material. You can give up to 20 materials to build up the grain. Mass fractions do not have to add up to one, they will be renormalized. All materials will be mixed together using the *Bruggeman* rule, and vacuum can be added through the porosity. A **-c** switch before each `KEY-or-FILE` is optional.

`-m KEY-or-FILE [MFRAC]`
: Like **-c**, but place this material into the grain mantle. Multiple mantle materials will be mixed using the Bruggeman rule, and than that mix will be added to the core using the *Maxwell-Garnett* rule. The **-m** is *not* optional, it must be present.

`-p POROSITY [P_MANTLE]`
: Porosity, the *volume* fraction of vacuum, a number smaller than 1. The default is 0. A single value will apply to both core and mantle, but a second value will be specific for the mantle (and may be 0).

[1]This requires the `cfitsio` library to be installed on your system.

4.2. Grain geometry and computational method

If no method is explicitly specified, the default is **-dhs 0.8**, i.e. DHS with f_{max}=0.8.

`-dhs [FMAX]`

Use the *Distribution of Hollow Spheres* (DHS, Min+ 2005) approach to model deviations from perfect spherical symmetry and low-porosity aggregates. Spheres with inner holes with volume fractions between 0 and f_{max} (default 0.8) are averaged to mimic irregularities. f_{max}=0 means to use solid spheres (Mie theory), i.e. perfectly regular grains. For backward compatibility, **-fmax** can be used instead of **-dhs**.

`-mmf [A0 [DFRAC-OR-FILL [KF]]]`

Use *Modified Mean Field* theory (MMF, Tazaki & Tanaka 2018) to compute opacities of highly porous or fractal aggregates. **-c**, **-m**, and **-p** determine the composition of monomers with radius `A0` (default 0.1μm). Particles will be aggregates with a *compact size* given by the **-a** switch, giving rise to $N = a^3/a_0^3$ monomers. `DFRAC-OR-FILL` specifies either the fractal dimension (if $>$1) or the *volume filling factor* (if $<$1). The default is 0.2. KF may be used to change the default prefactor.

`-mie`

Do a standard *Mie* calculation for perfect spheres. This is short for **-dhs 0** .

`-cde`

Compute CDE (continuous distribution of ellipsoids) Rayleigh limit opacities.

4.3. Grain size distribution

`-a AMIN [AMAX [APOW [NA]]]`

Specify (minimum) grain radius, and optionally maximum grain radius, the size distribution powerlaw and the number of size bins. You may also use options to set individual values with **-amin**, **-amax**, **-apow**, **-na**. The defaults are 0.05 μm, 3000 μm, 3.5, and *10 per size decade with a fixed minimum of 5*, respectively.

> If only a single size is specified with **-a**, then a_{max}=a_{min} and n_a=1 are implied.

> If a_{max} is negative, the range will be $a_{min} \pm a_{max}$, and a_{pow} defaults to 0.

4.4. Wavelength grid

`-l LMIN [LMAX [NLAM]]`

Specify the (minimum) wavelength, and optionally the maximum wavelength and the number of wavelengths points for the construction of the wavelength grid. The default values are 0.05 μm, 10000 μm, and 300, respectively. You may also use the options **-lmin**, **-lmax**, and **-nlam** (or **-nl**) to set individual values.

> If only one wavelength is specified with **-l**, then λ_{max}=λ_{min} and n_λ=1 are implied.

`-l FILE`

Read the wavelength grid from `FILE`. The file may start with comment lines, and the first non-comment line needs to contain the number of wavelength values in the data block below it. In the data block, the first column is expected to hold the wavelength values, in μm. In particular, an `lnk` file could be used here.

4.5. Controlling the output

The standard output is the file `dustkappa.dat`, with the opacities and the asymmetry parameter *g*. The following options control and extend the output.

`-o [DIR]`
: Put the output files in directory `DIR` instead of the current working directory. `./output` will be used if `-o` is present but `DIR` is not specified.

`-s [NANG]`
: Include the full scattering matrix in the output. `NANG` can optionally specify the number of equally-spaced angular grid points to cover the range of angles between 0 and 180 degrees. The default for `NANG` is 180 and should normally be just fine.

`-d [NSUB]`
: Divide the computation up into n_a parts to produce a file for each grain size. Each size will be an average over a range of `NSUB` (default 5) grains around the real size.

`-chop [NDEG]`
: Cap the first `NDEG` (2 if unspecified) degrees of the forward scattering peak.

`-fits`
: Write `dustkappa.fits` instead of ASCII output. With `-d`, write n_a files.

`-radmc [LABEL]`
: RADMC-3D uses a different angular grid and scattering matrix normalization. File names will contain `LABEL` if specified and have the extension `.inp`.

`-print [VAR]`
: Write to `STDOUT` instead of files. The default is to write λ, κ_{abs}, κ_{sca}, κ_{ext}, and g. When `VAR` is any of `kabs`, `ksca`, `kext`, or `g`, write only that, without header. You can use this to extract a single value, for example the 850μm extinction opacity of grains between 1 and 3mm: `optool -a 1000 3000 -l 850 -print kext`

5. Material properties

`optool` needs refractive index data to work. For your convenience, a useful list of materials is compiled into `optool`. You can also find and use other data. No matter where the data is from, you should *always* cite the original laboratory papers.

5.1. Built-in materials

To access one of the built-in materials, specify the corresponding key string like `pyr-mg70`. In each material class we have selected a useful default, accessible with an even simpler generic key (for example, `pyr` is an alias for `pyr-mg70`). Most of the built-in refractive index datasets have a reasonably wide wavelength coverage - the few exceptions are highlighted by bold-face numbers. If a material is being used outside of the measured region, `optool` will still function, using extrapolated optical properties.
Even the limited number of materials we have selected to include with `optool` can be daunting. To get started with some kind of standard opacity, we recommend to work with pyroxene $\boxed{\text{pyr}}$, carbon $\boxed{\text{c}}$, and (at low temperaturs) water ice $\boxed{\text{h2o}}$.

-c Key generic	-c Key full key	Material	State	ρ g/cm^3	λ_{min} μm	λ_{max} μm	Reference
	pyr-mg100	$MgSiO_3$	amorph	2.71	0.2	500	Dorschner+95
	pyr-mg95	$Mg_{0.95}Fe_{0.05}SiO_3$	amorph	2.74	0.2	500	Dorschner+95
	pyr-mg80	$Mg_{0.8}Fe_{0.2}SiO_3$	amorph	2.9	0.2	500	Dorschner+95
$\boxed{\text{pyr}}$	pyr-mg70	$Mg_{0.7}Fe_{0.3}SiO_3$	amorph	3.01	0.2	500	Dorschner+95
	pyr-mg60	$Mg_{0.6}Fe_{0.4}SiO_3$	amorph	3.1	0.2	500	Dorschner+95
	pyr-mg50	$Mg_{0.5}Fe_{0.5}SiO_3$	amorph	3.2	0.2	500	Dorschner+95
	pyr-mg40	$Mg_{0.4}Fe_{0.6}SiO_3$	amorph	3.3	0.2	500	Dorschner+95
ens	pyr-c-mg96	$Mg_{0.96}Fe_{0.04}SiO3$	cryst	2.8	**2.0**	**99**	Jäger+98
ol	ol-mg50	$MgFeSiO_4$	amorph	3.71	0.2	500	Dorschner+95
	ol-mg40	$Mg_{0.8}Fe_{1.2}SiO_4$	amorph	3.71	0.2	500	Dorschner+95
for	ol-c-mg100	Mg_2SiO_4	cryst	3.27	**5.0**	200	Suto+06
	ol-c-mg95	$Mg_{1.9}Fe_{0.1}SiO_4$	cryst	3.33	**2.0**	8190	Fabian+01
fay	ol-c-mg00	Fe_2SiO_4	cryst	4.39	**3.0**	250	Fabian+01
	astrosil	$MgFeSiO_4$	mixed	3.3	6e-5	1e5	Draine+03
$\boxed{\text{c}}$	c-z	C	amorph?	1.8	0.05	1e4	Zubko+96
	c-p	C	amorph	1.8	0.11	800	Preibisch+93
gra	c-gra	C graphite	cryst	2.16?	0.001	1000	Draine+03
org	c-org	CHON organics	amorph	1.4	0.1	1e5	Henning+96
	c-nano	C nano-diamond	cryst	2.3	0.02	**110**	Mutschke+04
iron	fe-c	Fe	metal	7.87	0.1	1e5	Henning+96
	fes	FeS	metal	4.83	0.1	1e5	Henning+96
	sic	SiC	cryst	3.22	0.001	1000	Laor93
qua	sio2	SiO_2	amorph	2.65	0.0006	500	Kitamura+07
cor	cor-c	Al_2O_3	cryst	4.0	0.5	**40**	Koike+95
$\boxed{\text{h2o}}$	h2o-w	Water ice	cryst	0.92	0.04	2e6	Warren+08
	h2o-a	Water ice	amorph	0.92	0.04	2e6	Hudgins+93
co2	co2-w	CO_2 ice	cryst	1.6	0.05	2e5	Warren+86
nh3	nh3-m	NH_3 ice	cryst	0.75	0.14	200	Martonchik+83
co	co-a	CO ice	amorph	0.81	**3.8**	**5.8**	Palumbo+06
	co2-a / c	CO_2 ice	am / cr	1.2	**2.5**	**20**	Gerakines+20
	ch4-a / c	CH_4 ice	am / cr	0.47	**2.0**	**20**	Gerakines+20
	ch3oh-a / c	CH_3OH ice	am / cr	0.78/1.02	**2.0**	**24**	Gerakines+20

5.2. External refractory index files (`lnk` files)

`optool` can use external refractive index data in files with the following format:

- The file may start with several comment lines (lines starting with `!`, `#`, or `*`).
- The next line contains two numbers, the number of wavelengths n_λ and the specific density ρ of the material in g/cm^3.
- The remaining lines should form three columns of data: $\lambda[\mu\text{m}]$ (sorted either up or down), and the real and imaginary parts of the refractive index, n and k.

More data ready for use with `optool` is in this repository. Other resources are the Jena database, ARIA and original papers in the literature. Don't forget to add the line with n_λ and ρ! If that is not possible, `optool` will count the lines and you can specify the density after the mass fraction, like this: `optool -c path/to/file.lnk 0.7 3.42`.

6. Output files

dustkappa.dat

This is an ASCII file containing the basic opacity results. It starts with a comment section describing the dust model and also showing the exact command line that was used to produce the file. The header is followed by the format number (3, currently), followed by the number of wavelengths in the grid, both on lines by themselves. Then follows a block with these columns:

1. wavelength λ [micron]
2. mass absorption cross section $\kappa_{\rm abs}$ [cm^2/g]
3. mass scattering cross section $\kappa_{\rm sca}$ [cm^2/g]
4. asymmetry parameter g

dustkapscatmat.dat

ASCII file with cross sections and full scattering matrix. It is an extended version of the `dustkappa.dat` file. This file has a format number (0), the number of wavelengths and then the number of angular points after the comment section. After an empty line, the same opacity block as in `dustkappa.dat` is present. Another empty line is followed by a list of the grid angles, another empty line, and then the scattering matrix elements for all wavelengths and all angles. The comment section at the start of the file shows the structure in a formal way. See the appendix for information about the normalization of the scattering matrix and about the angular grid that is used for it. Also, see the `-radmc` switch which will modify[2] the output to make sure it can be used as an input file for RADMC-3D.

dustkappa.fits

The FITS-file (ending in '.fits') is written instead of the ASCII output when using the `-fits` switch. It has two HDU blocks. The first block contains the cross sections per unit mass. This is an $n_\lambda \times 4$ matrix with these columns:

- wavelength in [micron]
- $\kappa_{\rm ext}$, $\kappa_{\rm abs}$, $\kappa_{\rm sca}$, all in [cm^2/g]

The second block contains the scattering matrix elements. It is a $n_\lambda \times 6 \times n_{\rm ang}$ matrix, containing the 6 elements of the scattering matrix for $n_{\rm ang}$ equidistant scattering angles from forward scattering (element 0) to backward scattering (element $n_{\rm ang}$-1), for each λ. The matrix elements are F_{11}, F_{12}, F_{22}, F_{33}, F_{34}, and F_{44}.

optool.tex

As a little gimmick, you can run `optool2tex` with the exact same command line arguments as used in an `optool` run. `optool.tex` then contains text and a table, describing the methods used for the opacity computation and listing the composition of the grains. All relevant references are given - the BibTeX file `optool.bib` is required for the file to be processed properly. You can rework this text to include it into your paper. For more details, read the comment section in `optool2tex`.

[2]This includes a change of the angular grid and a change in the normalization of the scattering matrix.

7. Python interface

`optool` comes with a `python` module `optool.py` that runs `optool` in the background[3] and puts all computed quantities as `numpy` arrays into a python object. This makes it straight forward to inspect and further process the output. Here is how to use it:

```
import optool
p = optool.particle('~/bin/optool pyr 0.8 -m ice 0.2 -na 24 -d')
```

The argument to `optool.particle()` must be a valid shell command[4] to run `optool`, if necessary with the full path to the `optool` binary. Depending on the presence of the `optool`'s **-d** switch, the command will produce opacities either for $n_p = 1$ particle, or for $n_p = n_a$ particles. Most of the attributes (with the exception of the global wavelength and angular grids) will therefore be arrays with the first dimension equal to n_p, even if $n_p = 1$. The resulting object will have the following attributes:

Attribute	**Type/Shape**	**Quantity**
`cmd`	`string`	The full command given in the particle() call
`radmc`	`boolean`	Output follows RADMC conventions
`scat`	`boolean`	Scattering matrix is available
`nlam`	`int`	Number of wavelength points
`lam`	`float[nlam]`	The wavelength grid
`nang`	`int`	Number of scattering angles
`scatang`	`float[nang]`	The angular grid
`materials`	`[[[...]...]... ]`	Lists with [location,$m_{\rm frac}$,ρ,material]
`np`	`int`	Number of particles, either 1 or (with -d) n_a
`fmax`	`float[np]`	Maximum volume fraction of vacuum for DHS
`pcore`, `pmantle`	`float[np]`	Porosity of the core/mantle material
`amin`, `amax`	`float[np]`	min/max grain size used for each particle
`nsub`	`int[np]`	Number of sizes averaged for each particle
`apow`	`float[np]`	Negative size distribution power law (e.g. 3.5)
`a1`, `a2`, `a3`	`float[np]`	Mean <a>, $\sqrt{<a^2>}$, and $\sqrt[3]{<a^3>}$
`rho`	`float[np]`	Specific density of grains
`kabs,ksca,kext`	`float[np,nlam]`	Absorption,scattering,extinction cross section
`gsca`	`float[np,nlam]`	Asymmetry parameter
`f11`, ..., `f44`	`float[np,nlam,nang]`	Scattering matrix element F_{11}, ... ,F_{44}
`chop`	`float[np]`	Degrees chopped off forward scattering
`plot()`	`method`	Plot the cross sections and matrix elements
`computemean()`	`method`	Compute Planck/Rosseland mean opacities
`tmin,tmax,ntemp`	`float,float,int`	Temperature grid for mean opacities
`temp`	`float[ntemp]`	Temperatures used for mean opacities
`kplanck,kross`	`float[np,ntemp]`	Mean opacities, after calling `computemean()`
`norm`	`string`	Current scattering matrix normalization
`scatnorm()`	`method`	Check/change scat. matrix normalization
`sizedist()`	`method`	Sum opacities over a size distribution

[3]The module runs the command as a subprocess, with output to a temporary subdirectory.
[4]As a string, or as a list like `['/path/to my/command','arg1','arg2',...]`.

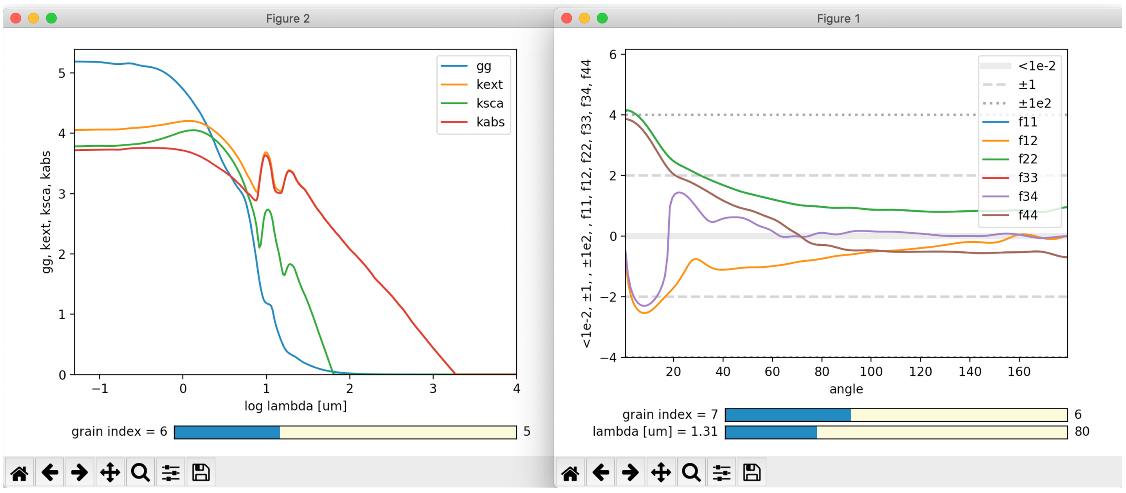


Figure 1: Screenshot of plots created by running `p.plot()` on an optool particle.

Applying the `plot()` method to a `particle` object like `p.plot()` will produce these plots:

- a plot showing the opacities κ_{abs}, κ_{sca}, and κ_{ext} as a function of wavelength, along with the asymmetry parameter g (on a linear y-scale). Note that the blue g curve does not have its own axis, imagine the full y axis going from 0 to 1 for g.
- a plot showing the scattering matrix elements as a function of scattering angle, with sliders to go through grain sizes and wavelengths. When interpreting the y axis, note that we plot the positive/negative $\log_{10}$ of positive/negative matrix elements, compressing the range from 10^{-2} to 10^{2} into a line (use the grey lines as a guide, ignore the y-axis labels).
- If the `computemean` method has been called first, the mean opacities κ_{Planck} and κ_{Ross} are shown in a separate plot. The mean opacities are per unit of grain mass, so please apply a dust-to-gas mass ratio to obtain opacities for a gas-dust mixture.

The python module has a few more tricks up its sleeve (for details check the documentation inside the Python module file `optool.py`):

- A `lnktable` class to read, plot, modify and write `lnk` files.

  ```
  x = optool.lnktable('lnk_data/sio2-Kitamura2007.lnk')
  x.plot()
  ```

- Compute Planck and Rosseland mean opacities

  ```
  p = optool.particle('optool  pyr 0.87  c 0.13 -p 0.25')
  p.computemean(tmin=10.,tmax=1500.,ntemp=300)
  ```

- *Particle arithmetic*: multiplying `optool.particle` objects with factors and adding them, or applying size distributions to a pre-computed set of opacities. The following page contains a number of examples.

Compute the opacities of 100 olivine silicate grain sizes and of 50 carbon grain sizes, and store the opacities in cache directories. This works by specifying the directory as the second argument. In a new session, if the directories still exist and were produced using the same commands, the opacities are simply read back in.

```
import optool
import numpy as np
sil  = optool.particle('optool -d -a 0.001 100 0 100 ol-mg50',cache='sil')
carb = optool.particle('optool -d -a 0.001 3.0 0 50  c',cache='carb')
```

Apply powerlaw size distributions, and limit the size of the contributing grains. Note that a power law f(a)$\propto$ a$^{-3.5}$ implies using a power a$^{-2.5}$ when computing the number of particles per size bin on a logarithmic size grid. No normalization is necessary - the `sizedist` method will take care of that.

```
nsil = sil.a1**(-2.5)               # power law, no normalization required
nsil[sil.a1<0.01] = 0               # no grains smaller than 0.01um
nsil[sil.a1>0.3]  = 0               # no grains larger  than 0.3um
sil_pl = sil.sizedist(nsil)         # pass the relative number for each size

nc = carb.a1**(-2.5)                # power law, no normalization required
nc[carb.a1>0.3]=0                   # no grains larger than 0.3um
carb_pl = carb.sizedist(nc)         # pass the relative number for each size
```

`sil_pl` and `carb_pl` are now objects with a single opacity each, obtained by adding opacities with the weights of the size distribution. The opacities are still per g of total grain mass. Let's add these two opacities with mass weights, to get something resembling an interstellar dust opacity produced by a mixture of silicate and carbon grains:

```
ptot = 0.7*sil_pl + 0.3*carb_pl     # weights should add up to 1
ptot.plot()                         # plot the resulting opacity
```

Now let's assume we are looking at an interstellar cloud, where the dust is just one percent of the total mass. We want to have the opacity per unit of *gas mass* instead, and we need Planck and Rosseland mean opacities:

```
p_ism = ptot * 0.01                 # dilute the opacity
p_ism.computemean(tmax=1300)        # Compute mean opacities
p_ism.plot()                        # Plot the results
```

Other size distributions can be made just as easily. Here is a log-normal size distribution for the silicate grains, with a peak abundance at a size of a_m=1.3 microns, and a logarithmic width of σ=0.2:

```
sil_ln = sil.sizedist( np.exp(-0.5* (np.log(sil.a1/1.3)/0.2)**2) )
sil_ln.write('dkap_ln.dat')         # write opacity to a file
```

A. Units

Due to conventions in our field, the input and output of `optool` uses the following units

- **microns** for grain sizes and wavelengths.[5]
- **g/cm^3** for mass densities of materials
- **cm^2 g^{-1}** for opacities $\kappa_{\rm abs}$, $\kappa_{\rm sca}$, and $\kappa_{\rm ext}$
- **sr^{-1}** or **cm^2 g^{-1} sr^{-1}** for the scattering matrix elements, see below.

B. Scattering Matrix: The fine print

B.1. Phase function normalization

A number of different normalizations for the scattering matrix are being used in the literature and in computational tools. The differences are significant, and it is important to be aware of the choice. For `optool` we are using a convention (Hovenier (2004)) in which the average over all directions of the 1-1 element of the scattering matrix equals unity, i.e.

$$\oint_{(4\pi)} F_{11}(\lambda,\Theta)d\Omega = 2\pi\int_{-1}^{1} F_{11}(\lambda,\mu)\mathrm{d}\mu = 4\pi \quad , \tag{1}$$

with $\mu = \cos\Theta$. `optool` can also produce output for RADMC-3D which uses instead

$$\oint_{(4\pi)} Z_{11}(\lambda,\Theta)d\Omega = 2\pi\int_{-1}^{1} Z_{11}(\lambda,\mu)\mathrm{d}\mu = \kappa_{\rm sca}(\lambda) \quad . \tag{2}$$

The books by Bohren & Huffman and by Mishchenko use different normalizations again. You can change the normalization of the scattering matrix in the python interface with the `scatnorm()` method. By default, it checks the current normalization. Using an argument `'r'`, `'b'`, `'m'`, or `'h'` will modify the normalization.

B.2. Forward-scattering peak

Particles that are much larger than the wavelength of the considered radiation can show extreme forward scattering, where much of the *scattered* radiation is sent into just a few degrees around the forward direction. This can be difficult to handle for radiative transfer codes which have limited angular resolution or limited sampling. MCMax3D has the `nspike` keyword to deal with this issue. Other tools (e.g. RADMC-3D) require this to be taken care of by the process that creates the opacity files. The `-chop` switch specifies a number of degrees around the forward scattering direction. Inside that cone, the scattering matrix gets limited to the value at the edge of the cone. To compensate and ensure energy conservation, the scattering cross section will be reduced accordingly. As a result, the radiation that would be *scattered* into this narrow range of angles will be treated as if it did have *no interaction at all* with the grain.

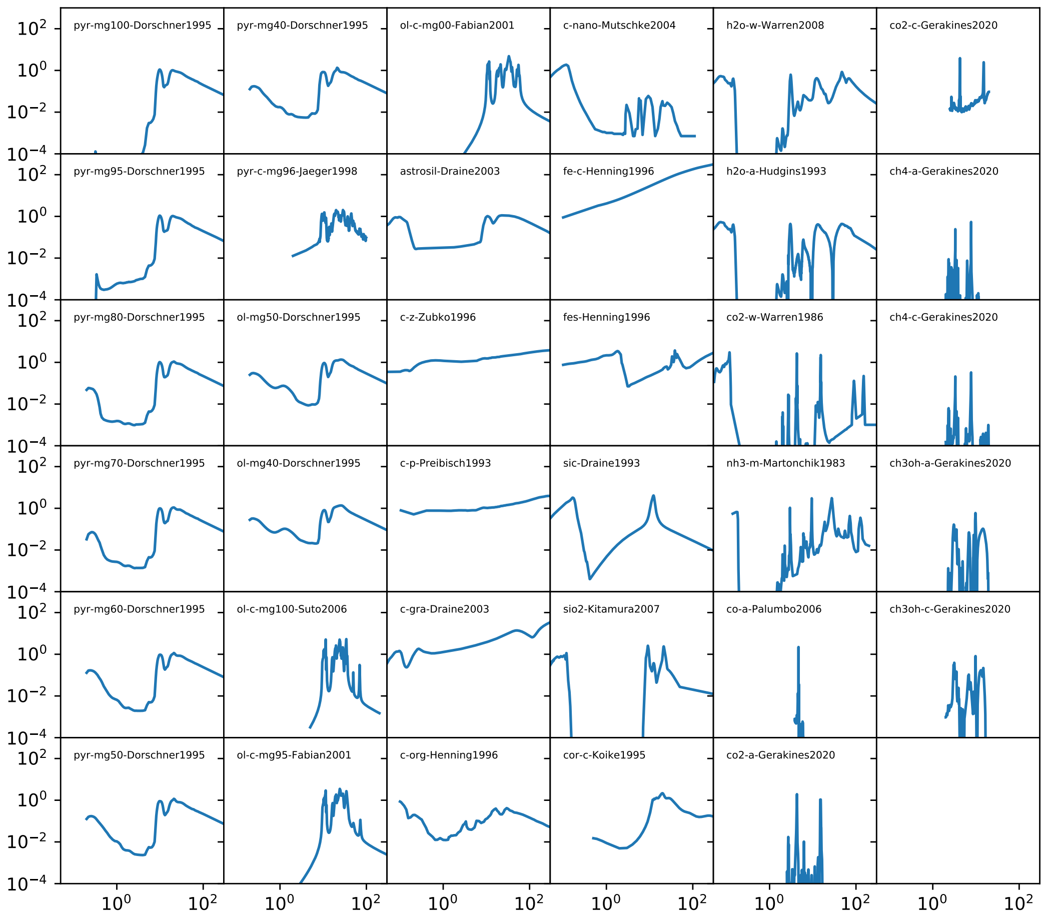
[5]When giving a grain size or a wavelength on the command line, you can write `1.3*mm`, `340*GHz`, or `4000/cm` and `optool` will do the right thing, converting to 1300μm, 881.7μm, and 2.5μm, respectively.

B.3. Angular grid

`optool` uses an angular grid in one degree steps from 0 to 180 degrees. The full degrees are the cell *interfaces* of that grid. `optool` computes the scattering matrix at the cell *midpoints*, i.e. at 0.5°, 1.5° etc to 179.5°, for a total of 180 values. The scattering matrix is normalized in this way, so that a numerical integral gives the correct result.
RADMC-3D requires the values of the scattering matrix on the cell *boundaries*, so at 0°, 1° etc to 180°, for a total of 181 values. For the input files for RADMC-3D, we interpolate and extend the computed values to the cell boundaries.

C. Overview of optical properties

This grid plot shows the imaginary parts of all built-in materials, in the wavelength range from 0.05 to 300 μm. Some if the ices have only data in a small range, where the vibrational transitions lie. However, these materials can be used over a much broader wavelength range, because the extrapolation becomes problematic only in the UV where electronic transitions kick in.



D. How to ingest refractive index data for another material

Using external refractive index data means that you have to keep track of where those files are. It can be convenient to compile your favorite materials into `optool`, so that accessing them will be as simple as using the built-in materials. Here is how to do that:

1. Give your `lnk` file a name exactly like `pyr-mg70-Dorschner1995.lnk`, where the start of the name (`pyr-mg70`) is the key to access the material and `Dorschner1995` (the text after the final `-`) is the reference.
2. Put this file into the `lnk_data` directory.
3. Optionally edit `lnk_data/lnk-help.txt`, so that `optool -c` will list the new material. Note that, in order to define *generic keys*, optool looks for pairs that look like `genkey -> fullkey` in this file.
4. Run `make ingest` to update `optool_refind.f90`, now with your new material.
5. Recompile and install the code.

E. Internals

This appendix describes some key aspects of the internal workings of the code.

Refractive Index Data

Measured refractive index data is obtained from data compiled into the code, or read-in from a file. That data is then interpolated and extrapolated onto the wavelength grid requested for the computation. Extrapolation toward short wavelengths is done keeping the refractive indices constant. Extrapolation toward long wavelengths assumes that the last two measured data points define a powerlaw. Interpolation in the measured grid is done using double-logarithmic interpolation.

Mixing

Once the refractive index for all involved materials is available, the core and the mantle mixtures are created independently, using the Bruggeman rule. Mass fractions are converted into volume fractions, and porosity is implemented using vacuum as an additional material. The subroutine doing the mixing uses an iterative procedure that is very stable, also for a large number of components.
If there is a mantle, the Maxwell Garnett rule is applied with the core being treated as an inclusion inside a mantle matrix.

DHS

In order to simulate irregularities in grains (irregular shapes, or the properties of low-porosity aggregates), `optool` averages the opacities of grains with an inner empty region, over a range of volume fractions of this inner region between 0 and f_{max}. The subroutine used to compute the opacities and scattering matrix elements for these structures is `DMiLay` (Toon & Ackerman 1981). When the size parameter $x = 2\pi a/\lambda$ exceeds a value of 10^4, no DHS averaging is used. A standard Mie calculation is performed, using the routine `MeerhoffMie` (de Rooij+ 1984), for a fixed size parameter of 5000, with proper scaling to the actual size of the particle.

MMF

To construct fluffy/fractal aggregates, `optool` needs the number of monomers N, the fractal dimension D_f, and a scaling factor k_f which are related to the radius of gyration R_g of the aggregate by $N = k_\mathrm{f}(R_\mathrm{g}/a_0)^{D_\mathrm{f}}$. The size a of the particles as specified by the **-a** switch is interpreted as the *compact*[6] size of all material in the aggregate, so that $N = a^3/a_0^3$, where a_0 is the monomer radius. The average volume filling factor f can be expressed by $f = N \cdot (\sqrt{3/5}\, a_0/R_\mathrm{g})^3$. To determine the structure of the aggregates, the user can specify a structure parameter. If that parameter is larger than 1, it is interpreted as the *fractal dimension* D_f. Using a fixed fractal dimension means that the volume filling factor will decrease with aggregate size. As an alternative, the structure parameter can be less than 1. In that case, it is interpreted as a fixed *volume filling factor* f that applies to all aggregate sizes - with the implication that then the fractal dimension increases as a function of size. The fractal prefactor k_f is chosen automatically so that the asymptotic density of small aggregates is the monomer material density. To force another value for the prefactor, it can be given explicitly as the third value of the `mmf` option. The following table summarizes the relevant equations.

	`-mmf A0 DF`	`-mmf A0 FILL`	`-mmf A0 DF KF`
f	$N^{(D_\mathrm{f}-3)/3}$	**given by user**	$\sqrt{27/125}\, k_\mathrm{f}^{3/D_\mathrm{f}} N^{3-1/D_\mathrm{f}}$
D_f	**given by user**	$3\ln N\, /\, \ln(N/f)$	**given by user**
k_f	$(5/3)^{D_\mathrm{f}/2}$	$(5/3)^{D_\mathrm{f}/2}$	**given by user**

With the structure defined, `optool` then applies the formalism from Tazaki & Tanaka (2018) and Tazaki (2021) to compute cross sections and the scattering matrix. `optool` also computes the phase shift $\Delta\phi$ to check the validity of the scattering matrix. If the condition $\Delta\phi < 1$ for accurate scattering matrix results is violated, a warning will be issued. However, the opacities will remain applicable.

CDE

CDE (Continuous Distribution of Ellipsoids) is an analytical formalism by Bohren & Huffman (1998) to compute the opacity of a very broad shape distribution. This method is only applicable in the Rayleigh limit $x = 2\pi a \ll \lambda$ and $|mx| \ll 1$. `optool` will issue a warning if the computation leaves the bounds of this condition. The scattering matrix will be computed from a single sphere in the Rayleigh limit.

F. Troubleshooting

- If you get oscillations in the opacities, in particular at long wavelengths, the grain size resolution is not sufficient. Use more grain sizes (**-a**, **-na** and **-d** switches).
- If you do not remember how to reproduce a specific run, just check the output file header. It contains the exact command that was used to produce the file.

[6] still including the porosity specified with the **-p** switch (which is porosity residing in the monomers themspelves), but not any "porosity" resulting from the aggregate structure

G. Acknowledgments

- The Jena Database of Optical Constants and the Aerosol Refractive Index Archive for their invaluable collections of refractive index datasets.
- Rens Waters, Thomas Henning, Xander Tielens, Elisabetta Palumbo, Laurent Pilon, Jeroen Bouwman, and Melissa McClure for discussions around optical properties of cosmic dust analogues.
- Charléne Lefévre for SIGMA, which inspired me to add grain mantles.
- Kees Dullemond for discussions about the RADMC-3D input format and the scattering matrix, for the idea to write `optool2tex` and for letting me include his incredible python plotting routine `viewarr` (available on github).
- Gabriel-Dominique Marleau for testing and feedback, in particular on `optool2tex`.

H. Bibliography

- Bohren, C.F. and Huffman, D.R. 1998, Wiley-VCH, *Absorption and Scattering of Light by Small Particles*
- Draine, B. 2003, ApJ 598, 1017
- Draine, B. 2003, ApJ 598, 1026
- Dorschner, J. et al. 1995, A&A 300, 503
- Fabian, D. et al. 2001, A&A 378, 228
- Gerakines, P. and Hudson, R. 2020, ApJ 901, 52
- Henning, Th. and Stognienko, R. 1996, A&A 311,291
- Hovenier, J., 2004, Report available on ADS.
- Jäger, C. et al. 1998, A&A 339, 904
- Kitamura, R. et al. 2007, Applied Optics 46,33, p. 8188
- Koike, C. et al. 1995, Icarus 114, 203
- Laor, A. and Draine, B. 1993, ApJ 402, 441
- Martonchik, J. 1984, Applied Optics 23, 541
- Min, M. et al. 2005, A&A, 432, 909
- Min, M. et al. 2016, A&A, 585, 13
- Mishchenko, M. et al. 2002, Cambridge University Press, *Scattering, absorption, and emission of light by small particles*
- Mutschke, H. et al. 2004, A&A 423, 983
- Okuzumi, S. et al. 2009, ApJ 707, 1247
- Tazaki, R. et al. 2016, ApJ 823, 70
- Tazaki, R. & Tanaka, H. 2018, ApJ 860,79
- Tazaki, R. 2021, MNRAS, in press
- Toon, O. & Ackerman,T. 1981, Applied Optics 20, 3657
- Woitke, P. et al. 2016, A&A 586, 103
- Palumbo, E. et al. 2006, PCCP 8, 279
- Preibisch, Th. et al. 1993, A&A 279, 577
- de Rooij W. and van der Stap, C. 1984, A&A 131, 237
- Steyer, T. 1974, PhD Thesis, The University of Arizona
- Warren, S. and Brandt, R. 2008, JGRD,113, D14220
- Warren, S. 1986, Applied Optics 25, 2650
- Zubko, V. et al. 1996, MNRAS 282, 1321