GGADT Documentation

Release 1.2.1

John Hoffman, Michael Tarczon

December 09, 2015

1 Citing GGADT 3

2 What's new 5
2.1 1.2.1 . . . 5
2.2 1.2.0 . . . 5
2.3 1.1.9 . . . 5
2.4 1.1.8 . . . 5
2.5 1.1.7 . . . 5

3 Installing GGADT 7
3.1 FFT Options . . . 7
3.2 Optimization . . . 7
3.3 What you need . . . 7
3.4 Mac Users . . . 8

4 Quick Example 9
4.1 Example 1: Total cross sections as a function of energy . . . 9
4.2 Example 2: Differential scattering cross section . . . 11
4.3 Example 3: Differential scattering cross section (in 2 dimensions) . . . 14

5 Invoking GGADT 17
5.1 Setting parameter values . . . 17
5.2 Parameter file syntax . . . 17

6 Arguments and parameters 19
6.1 Flags . . . 19
6.2 General options . . . 20
6.3 Units . . . 21
6.4 Ellipsoidal grains . . . 22
6.5 Agglomerations of spherical particles (single or multiple composition) . . . 22
6.6 Differential scattering cross section . . . 23
6.7 Energy-dependent scattering and absorption cross sections (integrated mode) . . . 23

7 Differential scattering cross section 25
7.1 Orientation averaging . . . 25
7.2 Large angle scattering . . . 25
7.3 1d vs 2d results . . . 26

8 Agglomerations of spherical monomers 27
8.1 Format of the target file . . . 27

9 Using a customized geometry 31

10 Multiple Compositions 33

11 Angle conventions 35
11.1 DDSCAT 35
11.2 MSTM 35

12 Additional files 39

13 Glossary 41

14 Indices and tables 43

Bibliography 45

Index 47

Version: 1.2 *Last updated*: December 09, 2015

Latest release `ggadt-1.2.1.tar.gz`

PDF `Manual`

Repository Github

`GGADT` is short for General Geometry Anomalous Diffraction Theory. It is a Fortran 90 program that uses ADT (Anomalous Diffraction Theory) (see *[VDH1957]*) to compute the differential scattering cross section (or the total cross sections as a function of energy) for a specified grain. ADT is valid when

$$|m-1| \ll 1$$

(i.e. when the grain is optically soft), and when

$$ka \gg 1$$

here k is the wavenumber of the incident photons and a is the effective radius of the grain, defined to be

$$a \equiv \left(\frac{3V}{4\pi}\right)^{1/3}$$

For more information about GGADT and ADT, see *[VDH1957]*, *[HD2015]*, and *[DA2006]*. See *Citing GGADT* for how to cite GGADT in a publication.

If you would like to discuss GGADT in more detail, or offer ideas for additional features, head over to the GGADT Google Group.

This documentation is a work in progress – some sections may be incomplete. Please use the GGADT Google Group to voice anything you notice is lacking or unclear in the documentation.

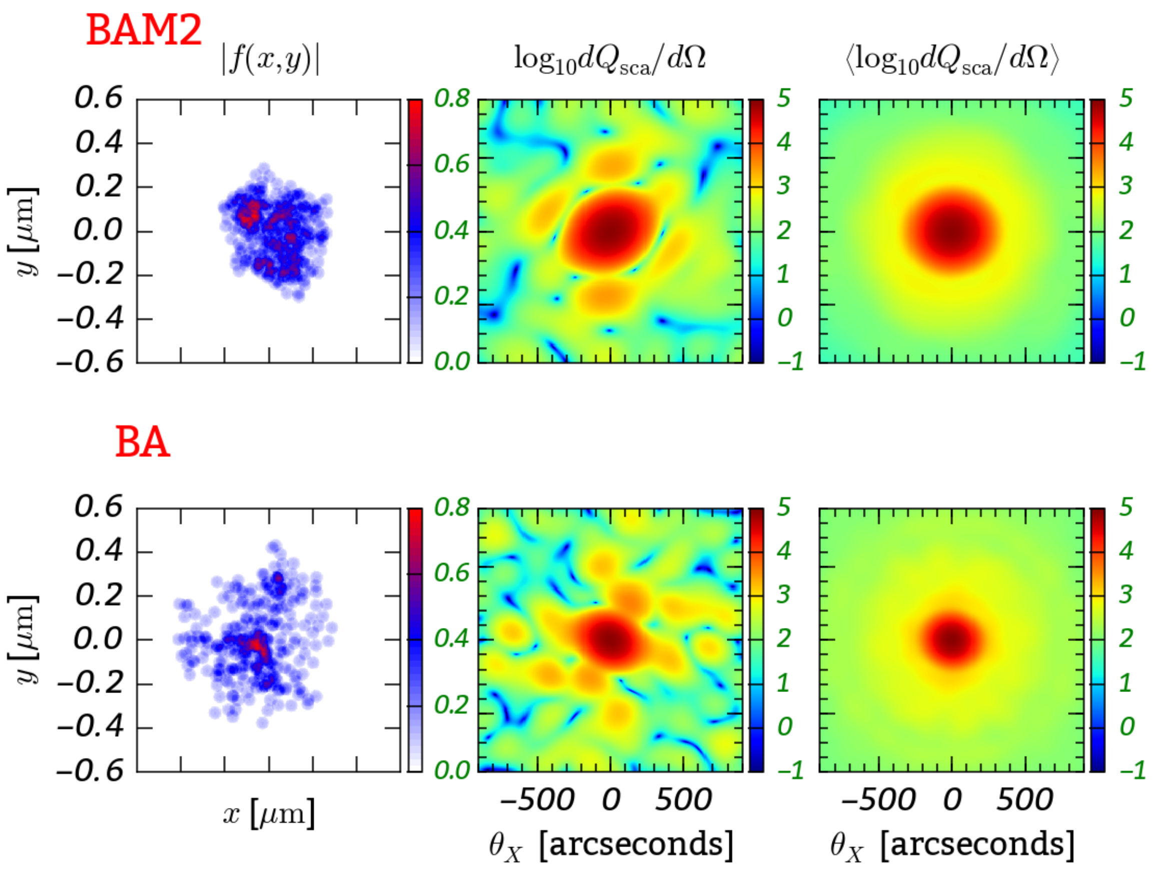

Fig. 1: GGADT results for two clusters of spheres with different porosities.

CHAPTER

ONE

CITING GGADT

If **`GGADT`** has been useful to you in your research, we would welcome your citations! Please cite the following paper (**not** the manual):

Hoffman, J., & Draine, B. T. 2015, *submitted for publication to the Astrophysical Journal*

CHAPTER

TWO

WHAT'S NEW

2.1 1.2.1

- Fixed minor bugs
- Now `gfortran` (v5.2.0) will compile GGADT without warnings when the `-Wall` flag is specified.

2.2 1.2.0

- GGADT should not break when attempting to do dCsca/dOmega calculations at large angles (i.e. close to 90 degrees).

2.3 1.1.9

- You can now tell GGADT what units you're using. See *`--units-file`*.

2.4 1.1.8

- **Removed all files/options related to FFTW support in order to avoid any confusion and to declutter the installation.**
  - FFTW support generates a disproportionate amount of overhead and complexity
  - We have run comparisons between GPFA (see *[T1992]*) and FFTW and haven't found any significant difference; if you are absolutely set on using another FFT library, let us know. If there are enough people that want support for additional FFT libraries, we will oblige.

2.5 1.1.7

- **Added `datadir` functionality for GGADT data files**
  - data files are refer to `default.params` and `allowed_ngrid_values.txt` files).
  - Users now **must** provide `configure` a `--datadir` value. Users should choose a directory that they have adequate permissions to read/write files. All GGADT files will be stored in the `datadir/ggadt` subdirectory.

CHAPTER

THREE

INSTALLING GGADT

GGADT should install easily on most Unix-based systems (e.g. Linux, Mac) by following the standard GNU install procedures, described in `INSTALL`:

```
$ ./configure --datadir=/path/to/datadir [--enable-openmp] ...
$ make
$ make install
```

This installs ggadt into a globally-accessible folder (usually `/usr/local/bin`).

Warning: Users *must* give `configure` a `--datadir`. This is where the defaults file and a couple of other necessary files will be stored (they'll actually be stored in the `${datadir}/ggadt` subdirectory). Pick a location where you have adequate permissions (e.g. your home directory).

If you did do **`make install`**, you can undo it by running **`make uninstall`**. To clean and remake GGADT, simply do **`make clean`** and then **`make install`**.

3.1 FFT Options

Right now, GGADT works with the GPFA FFT only (this is supplied with GGADT; you don't need to do anything extra to implement it). We originally planned to offer FFTW support, and this will come in future releases, but our benchmark tests didn't show any significant speed improvement when using FFTW over GPFA.

If you have another FFT library that you would like to use, let us know.

3.2 Optimization

It should give you a slight (5 - 10%) speed advantage to use the `-O3` optimization flag. We tried using `-Ofast`, but this produced a segfault for moderately large 2d grid sizes, and we didn't spend too much time trying to figure out why. So, you can use `-Ofast` for a little extra oomph, but just know that things might break unexpectedly (the normal, 1d case worked fine for us):

```
$ ./configure FCFLAGS="-O3" --datadir=/path/to/datadir [--enable-openmp] ...
```

3.3 What you need

Suitable C and Fortran 90 compilers GGADT was developed using the GNU compiler suite (version 4.5.4 and higher) throughout, but the configure script will try to work with most common compilers. GGADT has been tested to compile on versions 4.3 and later of `gfortran`. If you experience any compilation problems, please let us know!

OpenMP For the OpenMP version of GGADT, you must have C and Fortran compilers that are compatible with OpenMP version 4.0 or later.

3.4 Mac Users

- Install the latest edition of `XCode command-line tools`. As of Mac OSX v10.9.1, the command-line tools are not included in `XCode` automatically, so you have to do this on your own.
- **Macports** is highly recommended. To install,
  1. Download and install macports from http://www.macports.org/
  2. Install the `coreutils` package: `sudo port install coreutils`
  3. Install `gcc` (4.5.4 or higher is recommended): `sudo port install gcc45`
  4. Set the `gcc` compiler: `sudo port select --set gcc mp-gcc45`

CHAPTER

FOUR

QUICK EXAMPLE

A directory containing two example cases, `example/`, is found in the parent directory of **GGADT**. In this directory, you will find:

README A small file detailing the contents of the directory

Parameter files

> **`parameters_total_xs.ini`** Parameter file for computing total (abs, sca, ext) cross sections as a function of energy
>
> **`parameters_diffscat.ini`** Parameter file for computing differential scattering cross section as a function of scattering angle.
>
> **`parameters_diffscat_2d.ini`** Parameter file for computing (2d) differential scattering cross section as a function of scattering angle.

Python scripts

> **`plot_total_xs.py`** Plots output of GGADT calculations of total cross section
>
> **`plot_diffscat.py`** Plots output of GGADT calculations of differential scattering cross section
>
> **`plot_diffscat_2d.py`** Plots output of GGADT calculations of (2d) differential scattering cross section

Target files for clusters of spheres describes the geometry of a cluster of spheres, see *[S2003]* for more information

> **`BA.256.1.targ`** More porous aggregate
>
> **`BAM2.256.1.targ`** Less porous aggregate

4.1 Example 1: Total cross sections as a function of energy

The first example uses **GGADT** to compute the total cross sections (i.e. absorption, scattering, extinction) for a particular set of parameters, listed in `parameters_total_xs.ini`. A more detailed look at the parameter will come a bit later.

To run **GGADT** with the parameters specified in the parameter file `parameters_total_xs.ini`, simply do:

```
$ ggadt --parameter-file=parameters_total_xs.ini > total_xs.out
```

This will store the results into the file `total_xs.out`, which can then be read in by the python script **`plot_total_xs.py`**. To plot the results:

```
$ python plot_total_xs.py total_xs.out
```

Generating a plot that should look something like this

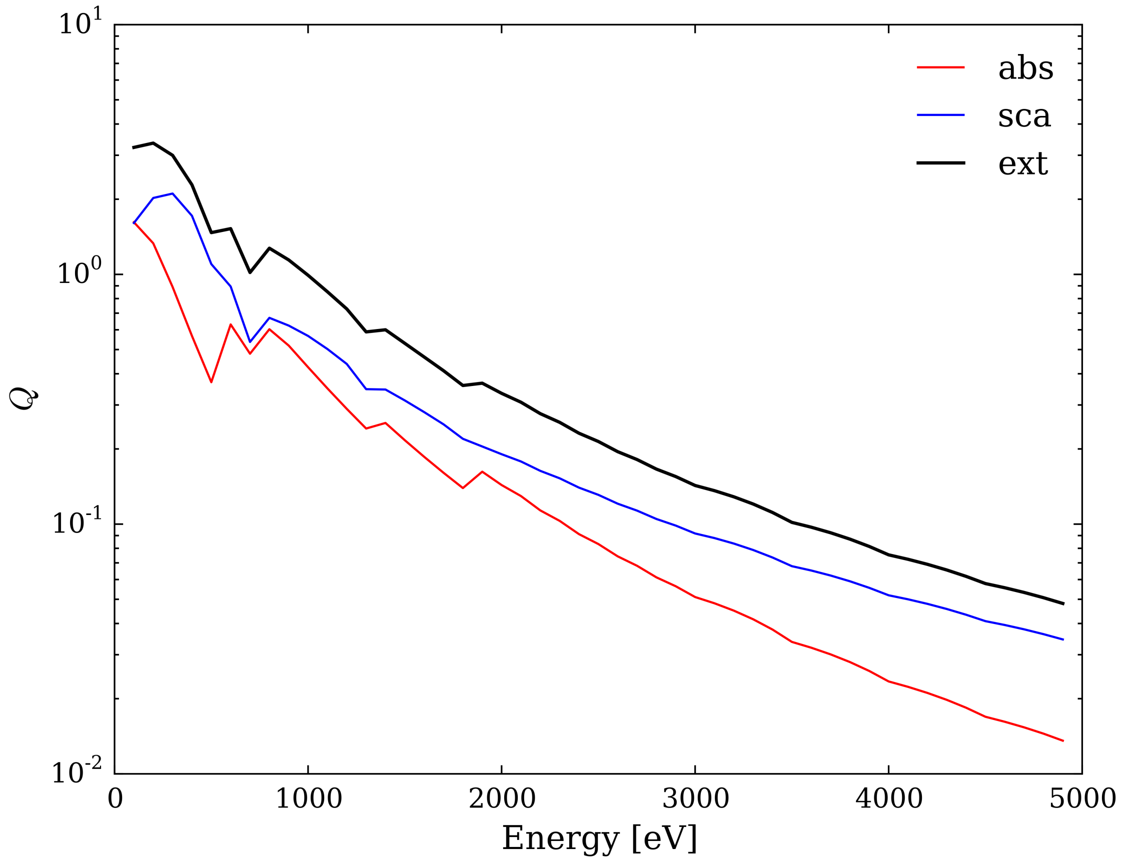

Fig. 4.1: Total cross section (**Example 1**)

4.1.1 A more thorough look at the parameter file `parameters_total_xs.ini`

Let's take a closer look at the parameter file now.

```
# Geometry of grain
# | one of: 'sphere','spheres' (or 'agglomerate'), or 'ellipsoid'
grain-geometry                  = 'spheres'
agglom-file                     = 'BAM2.256.1.targ'
```

These lines tell **GGADT** about the kind of object it will be dealing with. By specifying `--grain-geometry=spheres`, we're saying that the object is a cluster of spheres, and that the cluster is parameterized in the file `BAM2.256.1.targ`. Without going into too much detail, suffice it to say that this file describes the following object:

Let's move on.

```
use-efficiencies                = T
integrated                      = T
```

Here, we're telling **GGADT** that we want it to output results in terms of efficiencies, $Q \equiv C/\pi a^2$, and that we're computing the integrated cross sections (and not the differential scattering cross section).

```
# effective radius (radius of sphere with same volume)
aeff                            = 0.2
# Number of grid elements (along one dimension) to resolve grain
ngrain                          = 128
```

This tells **GGADT** that the effective radius of the grain $a_{\rm eff} \equiv (3V/4\pi)^{1/3}$ is 0.2 microns, and that it should be represented on a 128 x 128 two-dimensional grid.

```
material-file                   = "index_silD03"
```

The *`--material-file`* argument gives the location of the *index file*: a file specifying the energy-dependent refractive index of the grain material.

```
dephot                          = 0.1
ephot-min                       = 0.1
ephot-max                       = 5.0
```

These parameters set the energy range (and the energy resolution) over which the cross sections are computed.

```
   angle-mode                      = 'random'
    # Either 'sequential', 'random' or 'file'
    # | tells GGADT how to choose orientations over which to average
# | calculations.

   norientations                   = 64
    # If angle-mode is NOT 'file'; number of orientations
# | over which to average calcuations
```

Here we're saying that GGADT should average over 64 random orientations of the object.

4.2 Example 2: Differential scattering cross section

The second example uses GGADT to calculate the *differential scattering cross section* for a particular object and set of parameters. The `parameters_diffscat.ini` parameter file is used here.

To run **GGADT** with the parameters specified in the parameter file `parameters_total_xs.ini`, simply do:

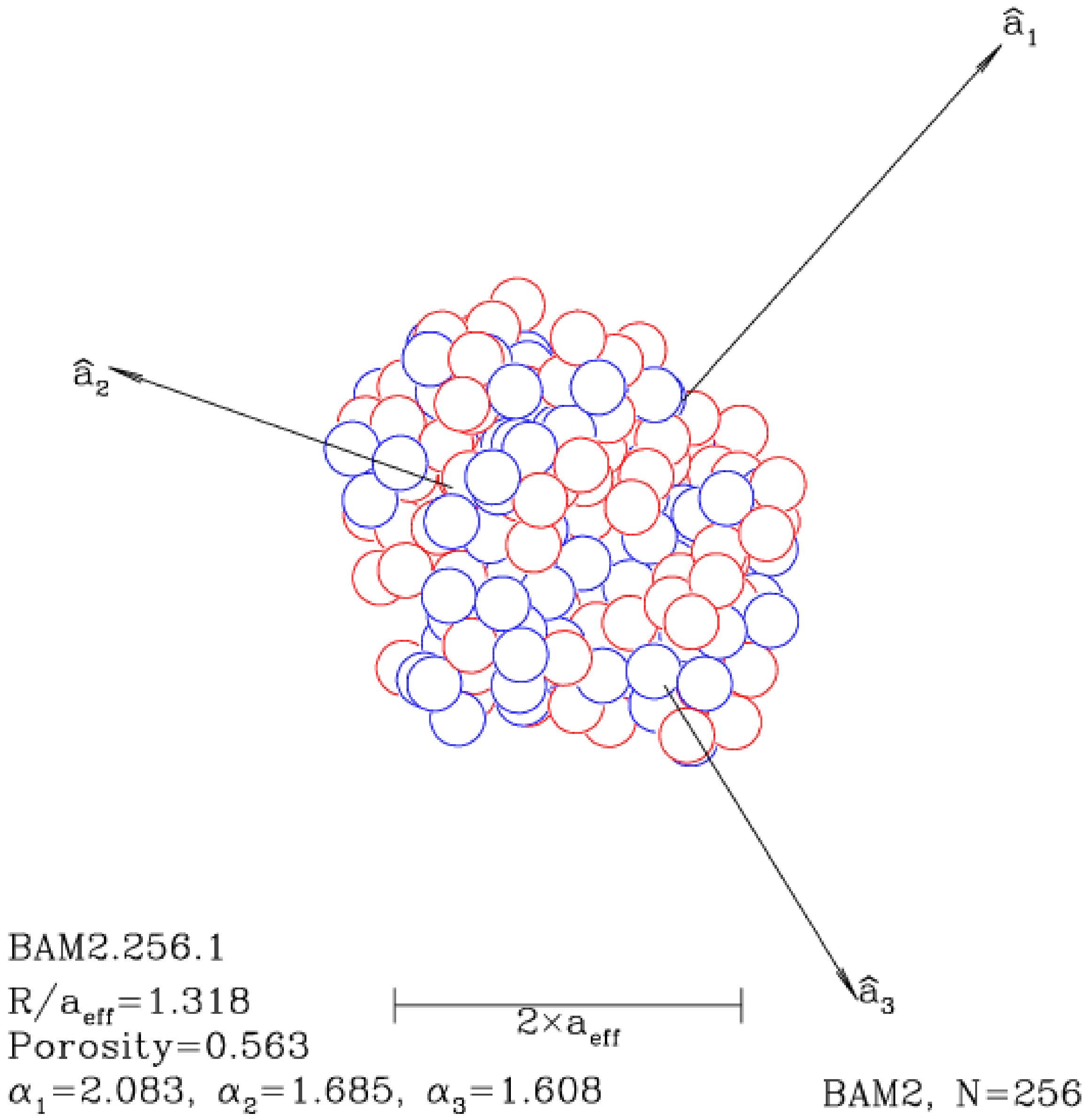

Fig. 4.2: The grain parameterized by `BAM2.256.1.targ`

```
$ ../src/ggadt --parameter-file=parameters_diffscat.ini > diffscat.out
```

This will store the results into the file `total_xs.out`, which can then be read in by the python script **`plot_total_xs.py`**. To plot the results:

```
$ python plot_diffscat.py diffscat.out
```

Generating a plot that should look something like this

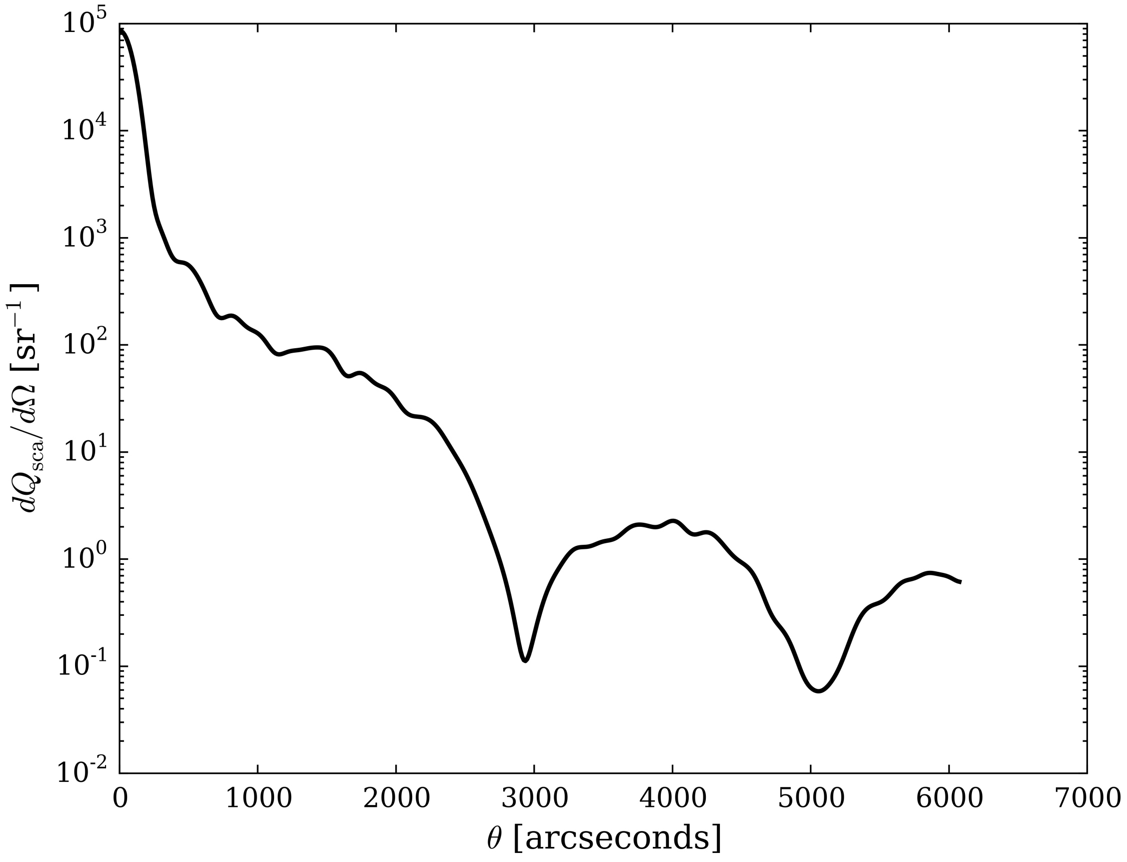

Fig. 4.3: Differential scattering cross section (**example 2**)

4.2.1 A more thorough look at the parameter file `parameters_diffscat.ini`

We'll pick through the differences between `parameters_diffscat.ini` and `parameters_total_xs.ini`.

```
# parameters for differential scattering cross section calculation
# units: keV
ephot                          = 2.0
#  Re(m-1)
# ior-re                         =-1.920E-4
#  Re(m)
# ior-im                         = 2.807E-5
# |
# | DO NOT define ior-re or ior-im if you also have defined
# | a material-file and 'ephot'. It will pull m(E) automatically.
# |
```

Here, we do the calculation at *one* photon energy (i.e. for a single value of the index of refraction, m). We can either set `--ior-re` and `--ior-im` ourselves, *or* we can specify a `--material-file` and a photon energy, `ephot` (in keV). We've chosen the latter here.

```
# number of angles to calculate diff. scat.
# cross section (ntheta) or spacing between them (dtheta)
# (arcseconds) Choose only one of these parameters
dtheta                          = 25.0
#nscatter                       = 100

# Maximum angle for which the diff. scat. cross
# section is calculated (arcseconds)
max-angle                       = 6000.
```

This tells GGADT that you want to compute $dC_{\mathrm{sca}}/d\Omega$ from 0 to 6000 *arcseconds*, with values spaced every $\delta\theta = 25$ arcseconds.

4.3 Example 3: Differential scattering cross section (in 2 dimensions)

The third example uses GGADT to calculate the differential scattering cross section, with the following differences:

1. `--do-full-2d-fft` is set to `T`.
2. `--norientations` is set to `1` (only beause this makes the plot look more interesting...)
3. `--ephot` is changed to 0.5 keV (for no particular reason).

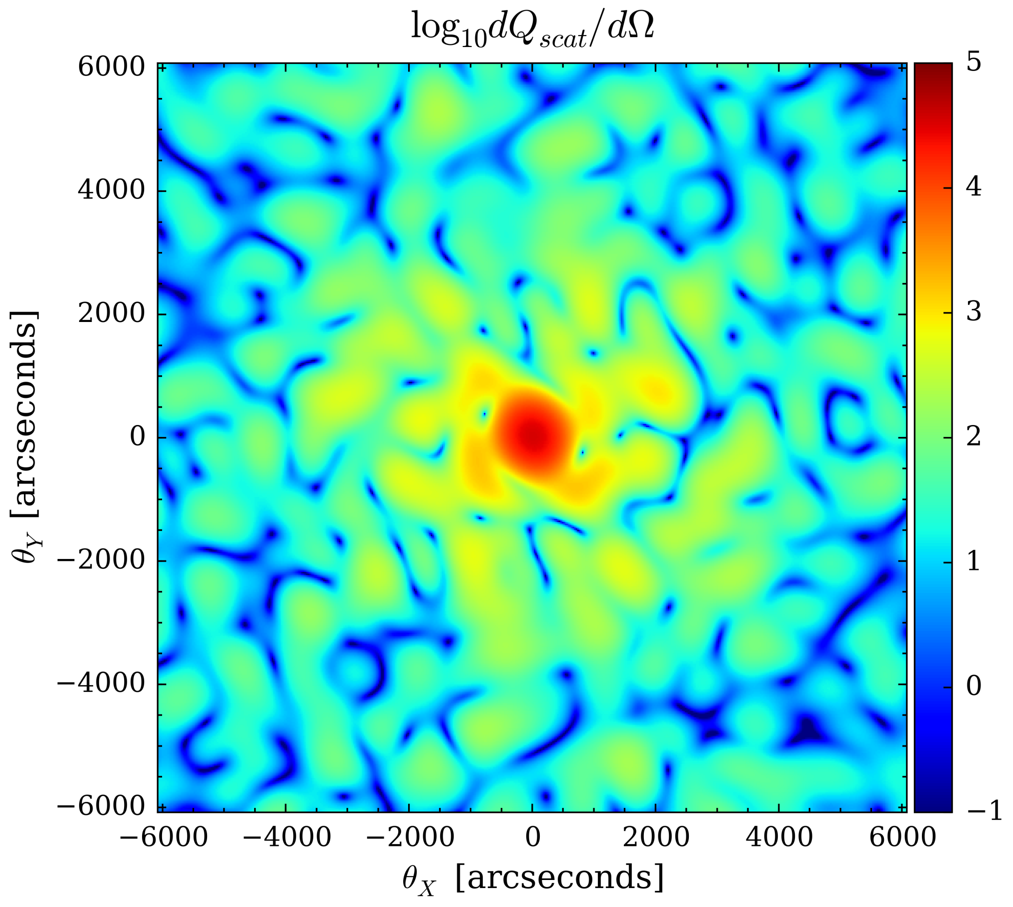

Fig. 4.4: Two-dimensional differential scattering cross section (**example 3**)

CHAPTER

FIVE

INVOKING GGADT

To see usage and all available options for ggadt, simply use the *-h* option:

```
$ ggadt -h
```

A detailed description of all of the available options are available in the *Arguments and parameters* section.

5.1 Setting parameter values

The format for running GGADT is:

```
$ ggadt [option]
```

With no options, GGADT runs calculations on a set of default grain parameters (specified in the `default.params` file). By default GGADT will calculate the differential scattering cross section as a function of scattering angle. To calculate the integrated scattering and absorption cross sections for a range of energies, add the *--integrated* flag as a command-line argument.

Parameters may be specified as command-line arguments,:

```
$ ggadt --parameter=value ...
```

or in a parameter file that is specified as a command-line argument,:

```
$ ggadt --parameter-file=/path/to/parameterfile
```

or a combination of the two::

```
$ ggadt --parameter-file=/path/to/parametefile --parameter=value ...
```

If a parameter is set by a command-line argument and by a parameter file, the command-line argument is used.

5.2 Parameter file syntax

The syntax of parameter files is::

```
#comment
parameter            = value
another-parameter    = another-value
...
```

Warning: One annoying feature of parameter file syntax is this: you cannot have spaces after a numerical parameter value. This is somewhat of a low-priority bug for the time being.

Warning: The `agglom-file` and `angle-file` options must be specified with quotes surrounding their values. In order to make the parameter file portable, you should specify an absolute file path, not a relative one. E.g.:

```
agglom-file='/absolute/path/to/parameter-file.ini'
```

NOT:

```
agglom-file=/absolute/path/to/parameter-file.ini
```

or:

```
agglom-file='../../parameter-file.ini'
```

Specifying a relative file-path (as in the last example) will work, as long as you always invoke GGADT from the location where the relative file-path is valid. For example, if I am in the parent directory, and my parameter file is in the data/parameter-files directory,:

```
agglom-file='data/parameter-files/parameter-file.ini'
```

will work, as long as I **always** run GGADT from the parent directory.

CHAPTER

SIX

ARGUMENTS AND PARAMETERS

GGADT supports the following options:

6.1 Flags

`--help, -h`
The "help" screen–prints usage information and available options.

`--version, -v`
Print the version number of GGADT on standard output and then exit successfully.

`--verbose, -d`
Be extra loud (useful for debugging)

`--quiet, -q`
Do not print progress. GGADT by default will print out a `percent complete:  XX.X%` statement during runtime; this option suppresses that.

`--timing, -t`
Do not output data. No output is useful if you care about measuring the runtime of GGADT.

`--nthreads``=value`
You can set the number of threads (for OpenMP); GGADT executes `call omp_set_num_threads(nthreads)`

`--use-padded-fft, -s`
This will force GGADT to do only one FFT. In order to provide you with the desired angular resolution of the differential cross section (set by *`--nscatter`*), GGADT will pad the 2-d grid on which the shadow function is defined. Forcing GGADT to use a padded FFT, instead of the more efficient method without padding, will run much slower.

`--force-numerical`
This will force GGADT to use a numerical calculation of the differential scattering cross section for spherical grains (an analytical Anomalous Diffraction Theory solution is implemented by default).

`--do-full-2d-fft`
Do not utilize the fact that you can make a 2d FFT a 1d FFT if you only care about the orientationally averaged value of $dC_{\mathrm{sca}}/d\Omega(\theta)$

`--save-shadow-function`
You can save a copy of the 2d shadow function of your grain (useful as a sanity check or to make pretty pictures)

`--save-orientations`
You can output all orientations that GGADT used in its calculation.

New in version 1.1.5: (added)

6.2 General options

--output-sigfigs=value

You can set how many significant digits GGADT will give you (this is useful for cutting down filesizes and read-in time).

New in version 1.1.5: (added)

--save-file-root=value

The "root" filename that the shadow function is saved to; for each orientation, GGADT saves the shadow function to:

```
[file_root]_shadow_function_o[00001, 00002, ..., N].dat
```

--parameter-file=value

Path to a correctly formatted parameter file.

--grain-geometry=value

Geometry of dust grain. Must be one of `SPHERE`, `ELLIPSOID`, `CUSTOM`, `SPHERES` (or equivalently, `AGGLOMERATION`). These are not case sensitive.

Note: A `SPHERES` grain geometry means that your grain is a conglomerate of smaller spheres. If you wish to use this grain geometry, you should supply a file (hereafter called a "agglomeration file", see the next option) that defines the relative positions of the spheres that make up the grain. The sizes of these spheres will be normalized so that the effective radius of the grain is consistent with *--aeff*.

--aeff=value

Effective radius of dust grain.

$$a_{\text{eff}} \equiv \left(\frac{3V}{4\pi}\right)^{1/3}$$

--ngrain=value

The (square root) of the number of grid elements with which you would like to resolve your grain. To get accurate results out to about 4000 arcseconds, we've found that using `ngrain=64` and higher provides sufficient accuracy for our needs.

--norientations=value

The number of grain orientations relative to the incoming X-ray photon over which to average the differential cross-section.

--grid-width=value

You can manually set the length of the grid (in microns) with this argument.

--axes-convention=value

The axes convention to use. Must be one of `DDSCAT` or `MSTM` (not case-sensitive). These conventions are explained in *Angle conventions*.

--angle-mode=value

Must be one of `RANDOM`, `SEQUENTIAL`, or `FILE`.

RANDOM Specifying random will average over norientations grain orientations, with each angle chosen from a uniform distribution over the domain of orientation angles.

SEQUENTIAL Specifying sequential will make GGADT use a set of angles that are evenly spaced over each of the orientation angle domains.

FILE Specifying file allows you to give GGADT specific orientation angles to use in orientational averaging. The file mode requires that you also specify *--angle-file* as the path to a file containing a list of orientation angles.

--angle-file=value

Path to a file containing a list of orientation angles which GGADT will use to compute orientational averages. Only used when *--angle-mode* is set to FILE.

For *--axes-convention* = DDSCAT, GGADT expects the file to contain three values per row: the first column is β values, the second column is θ values and the third column contains ϕ values.

Warning: At the moment, the MSTM convention is only partially supported, and GGADT might behave unpredictably if you use MSTM for now.

--material-file=value

Path to file defining the energy-dependent index of refraction and dielectric constant for the grain material you would like to use. The material files must be formatted in the way that GGADT expects, namely with a two-line header,:

```
17 =ICOMP: amorphous olivine-like astrosilicate
E(eV)       Re(n)-1    Im(n)     Re(eps)-1  Im(eps)
```

followed by data for the material:

```
1.000000E-05  2.435E+00 1.119E-03  1.080E+01 7.688E-03
1.240000E-05  2.435E+00 1.388E-03  1.080E+01 9.534E-03
1.771000E-05  2.435E+00 1.982E-03  1.080E+01 1.362E-02
2.000000E-05  2.435E+00 2.239E-03  1.080E+01 1.538E-02
...
```

6.3 Units

--input-unit-length=value

--input-unit-angle=value

--input-unit-energy=value

Tells GGADT what units you're using for your input.

New in version 1.1.9: (added)

--output-unit-length=value

--output-unit-angle=value

--output-unit-energy=value

Tells GGADT what units to use for outputting results. See *--units-file* for information about what units are available

New in version 1.1.9: (added)

--units-file=value

Path to file describing a set of units that GGADT will read and interpret. The format of the file should be:

```
TYPE_OF_UNIT    NAME    ABBREVIATION            CONVERSION_FACTOR
```

Where **TYPE_OF_UNIT** is one of ENERGY, LENGTH, or ANGLE.

The **CONVERSION_FACTOR** should be a numerical value that converts a quantity in this unit to the equivalent quantity in the corresponding SI unit.

The default units file is called `units.txt` and is located in the `datadir` directory. The contents of the file are:

```
LENGTH                  meter                       m                       1.0
LENGTH                  millimeter                  mm                      1.0d-3
LENGTH                  centimeter                  cm                      1.0d-2
LENGTH                  micron                      um                      1.0d-6
LENGTH                  micrometer                  um                      1.0d-6
LENGTH                  nanometer                   nm                      1.0d-9
LENGTH                  angstrom                    A                       1.0d-10
ANGLE                   radian                      rad             1.0
ANGLE                   degree                      deg             0.0174532925
ANGLE                   arcminute                   am                      0.000290888209
ANGLE                   arcsecond                   as                      4.84813681d-6
ENERGY                  joule                       J                       1.0
ENERGY                  electron-volt               eV                      1.60217662D-19
ENERGY                  kiloelectron-volt           keV                     1.60217662D-16
```

New in version 1.1.9: (added)

Warning: Make sure that **ALL** input values of a particular type (i.e. energy, length, angle), are in the same unit. **This includes default parameters that you may not have explicitly set**. The best way to ensure this is to use a parameter file and define all relevant variables there.
The one exception to this is the format of material (or index) files; GGADT assumes that the energy column of these files is in electron volts (eV).

6.4 Ellipsoidal grains

`--grain-axis-x``=value`, **`--grain-axis-y`**`=value`, **`--grain-axis-z`**`=value`
For ELLIPSOID grains, these define the *ratio* of the lengths of the x, y and z axes.

6.5 Agglomerations of spherical particles (single or multiple composition)

`--agglom-file``=value`
Path to a correctly formatted agglomeration file. Several agglomeration files are included with the distribution (see *Additional files*), and if you wish to create your own, you must mimic exactly the format of these files.

`--material-file1``=value`, **`--material-file2`**`=value`, **`--material-filen`**`=value`
Path(s) to the index file for material 1, 2, 3, etc. See *Multiple Compositions*.

`--material-tag1``=value`, **`--material-tag2`**`=value`, **`--material-tagn`**`=value`
The name(s) by which material 1, 2, 3, etc. are referred to in the agglomerate composition file. See *Multiple Compositions*

`--agglom-composition-file``=value`
Provides the path to a text file that lists the spherical agglomerates and a material 'tag' which is associated with an index file. See *Multiple Compositions* for more information.

The agglomerate composition file is assumed to be formatted with one header line, followed by as many lines as there are agglomerates in the grain, with each line containing the index of an agglomerate and a material 'tag':

```
    j      material tag
    1    silicate
    2    silicate
    3    iron
    4    silicate
    ...
```

6.6 Differential scattering cross section

--nscatter=value
The (square root) of the number of grid elements (in theta) with which you would like to resolve the differential scattering cross section.

--dtheta=value
Alternate to *--nscatter*: this defines the spacing of the scattering angles (in arcseconds) that you would like to calculate the differential cross section.

--ephot=value
Energy of the incident X-ray photon in keV.

--ior-re=value, **--ior-im**=value
The real and imaginary components of the complex index of refraction, minus one. If *--material-file* is specified, these parameters are overwritten by the linearly interpolated value of the index of refraction at E = *--ephot*.

--max-angle=value
The maximum scattering angle for which you would like to calculate the *differential scattering cross section*.

--do-phi-averaging
If *--do-full-2d-fft* is set, you have the option of also setting *--do-phi-averaging*, which averages the resulting $dC_{\mathrm{sca}}/d\Omega(\theta, \phi)$ over ϕ at each orientation, giving you a 1d result. In certain cases (e.g. for `CUSTOM` geometries), this is faster than doing many 1d FFT's. In almost all cases that we've run, however, doing a full 2d FFT makes things slower.

--nphi=value
When *--do-phi-averaging* is set, you can set the number of ϕ values over which you average the 2d differential scattering cross section.

6.7 Energy-dependent scattering and absorption cross sections (integrated mode)

You must also define a *--material-file* to do this calculation.

--integrated
This flag tells GGADT that you would like to calculate the energy-dependent integrated absorption and scattering cross sections.

--nephots=value
Defines the number of photon energies to calculate the cross sections between *--ephot-min* and *--ephot-max*. You can also define the energy spacing directly, via the *--dephot* parameter.

--dephot=value
Alternate to *--nephots*, defines the energy spacing between different photon energies for which the cross sections are calculated.

`--ephot-min``=value`
: Defines the minimum photon energy for which the cross sections are calculated

`--ephot-max``=value`
: Defines the maximum photon energy for which the cross sections are calculated

6.7.1 Default parameter values

Default parameter values are loaded from the `default.params` file located in the parent directory of GGADT. You can edit this file, or (as a safer option), edit a copy of this file and pass the path as the value for *`--parameter-file`*.

CHAPTER

SEVEN

DIFFERENTIAL SCATTERING CROSS SECTION

GGADT by default will produce an orientation-averaged, 1-dimensional result for $dC_{\rm sca}/d\Omega(\theta)$ for a set of $theta_m$ determined by either the *--dtheta* parameter or the *--nscatter* parameter.

7.1 Orientation averaging

GGADT has three methods for producing averages over orientations, all of which are set by the `angle-mode` parameter.

random GGADT will chose `norientations` random orientations

sequential GGADT will choose `norientations`, evenly spaced over (β, θ, ϕ).

file GGADT will read in **file**; each line contains three floating point numbers, representing (β, θ, ϕ) (if *--axes-convention* is set to `DDSCAT`), or (α, β, γ) if *--axes-convention* is set to `MSTM`. See *Angle conventions* for more information about axes conventions. The file should have just three columns, each separated by white-space of some kind (e.g. tab, space):

```
0.0         0.0 0.0
0.234 0.12 0.0
...
```

7.2 Large angle scattering

ADT is asymptotically correct as $|m-1| \to 0$ and $ka \to \infty$. In these limits, scattering at large angles is very small. If you're worried about large angle scattering (say $\theta > 10^\circ$), be careful. Obtaining accurate estimates of small intensity scattering at large angles (especially angles > 90 degrees) may require a more exact treatment of the scattering problem (e.g. Mie theory).

Though, technically, the theoretical framework of ADT allows you to calculate $dC_{\rm sca}/d\Omega$ out to arbitary scattering angle θ, GGADT will fail at angles close to 90 degrees. GGADT fails partly because of the computational shortcuts that allow GGADT to be extremely fast, and partly because the authors of GGADT didn't intend for GGADT to be used at very large scattering angles where ADT may not give the correct results.

If you're using GGADT to do *very* large angle scattering ($\theta > 50^\circ$), you will likely run into one or more of these problems:

1. For the analytic case of **spherical** particles, GGADT *should* be fine up to any angle you want, but because of some *programming quirks <https://xkcd.com/1513/>*, GGADT will break at $\theta = 90^\circ - \delta$. To make δ smaller, you should set *--grid-width* to as large a number as you need. An unfortunate side effect: computation time will increase as you increase *--grid-width*, even though you're not bothering with any numerical grids. Depending on how close to 90 degrees you'd like to get, this may present too large a strain on your patience.

We'd like to say this is all intentional in order to prevent you from abusing GGADT for large angle calculations, but really it's because we're terrible programmers.

2. For numerical calculations, you run into another limit; $\sin\theta_{\max} = \lambda/\Delta x$. This can be fixed by using a finer grid, but the above problems will still plague you.

7.3 1d vs 2d results

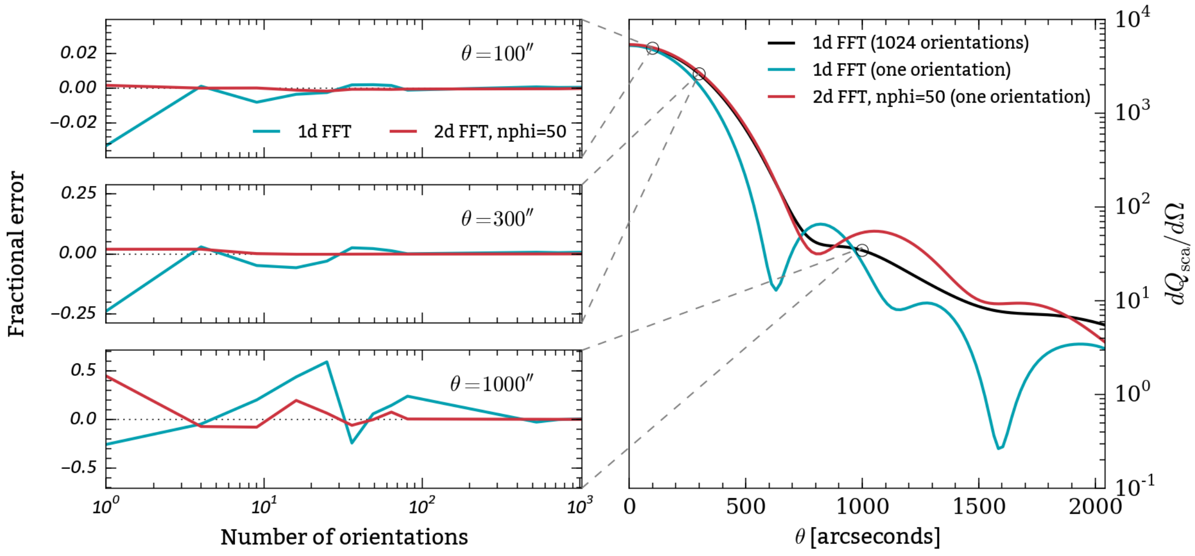

Fig. 7.1: How fast 1d and 2d computations converge. 2d computations, however, usually take significantly longer to run.

7.3.1 Using the `nphi` parameter

The *--nphi* parameter (only used when *--do-full-2d-fft* is specified) lets you take advantage of a full 2D calculation of $dC_{\rm sca}/d\Omega(\theta, \phi)$, so long as you're only interested in the *orientation-averaged* result that only depend on the scattering angle θ.

By averaging over ϕ for a single orientation, we reduce the number of random (or non-random) orientations that we need to calculate the full 2D $dC_{\rm sca}/d\Omega(\theta, \phi)$ in order to reach an accurate estimate of $dC_{\rm sca}/d\Omega(\theta)$ (the orientation-averaged result).

CHAPTER

EIGHT

AGGLOMERATIONS OF SPHERICAL MONOMERS

GGADT also natively supports grains composed of spherical *monomers*. Two examples:

In order to use this geometry, you must set `--grain-geometry` to `spheres` or `agglomeration`, and tell GGADT where the agglomeration target file is by setting `--agglom-file` to `/path/to/targetfile.targ`.

8.1 Format of the target file

GGADT expects a very particular format for the `.targ` file (Fortran is not very forgiving).

Here's a relevant snippet from the `read_spheres()` function in the `spheres.f90` module

```
    read(1,'(52X,i2,7X,i4)') MIGRATE,ISEED
read(1,'(i9,f12.2,3f11.6)') NS, VTOT, ALPHA(1), ALPHA(2), ALPHA(3)
read(1,'(3f10.6,13X)') A_mat(1,1), A_mat(2,1), A_mat(3,1)
read(1,'(3f10.6,13X)') A_mat(1,2), A_mat(2,2), A_mat(3,2)
read(1,'(A)') junk
```

And here's the beginning of the `BA.256.1.targ` file:

```
multisphere target generated by agglom with MIGRATE= 0 ISEED=  -1
      256      256.00   4.542054   3.871312   3.192810 = NS, VTOT, alpha(1-3)
  0.777146 -0.629320  0.000000 = A_1 vector
  0.629320  0.777146  0.000000 = A_2 vector
       j      x(j)        y(j)        z(j)      2*a(j)
       1    0.000000    0.000000    0.000000  1.000000
       2   -0.132818   -0.362148    0.922610  1.000000
       3   -0.231104   -0.859202   -0.456468  1.000000
       4   -0.064126   -0.651197    1.877456  1.000000
       5    0.289849   -1.257871    1.060564  1.000000
        ...
```

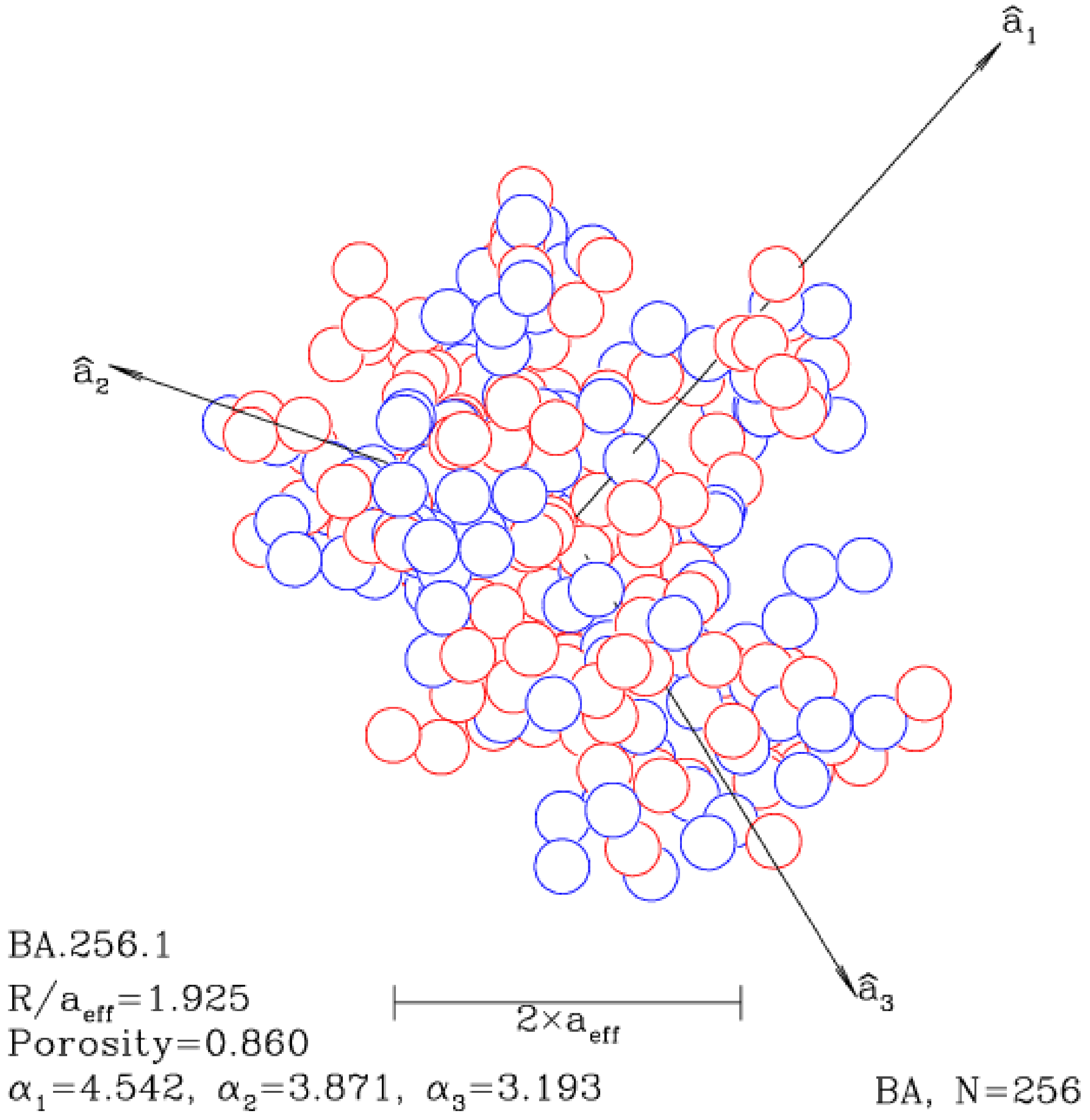

Fig. 8.1: “BA” aggregate (porous); 256 monomers; `BA.256.1.targ`

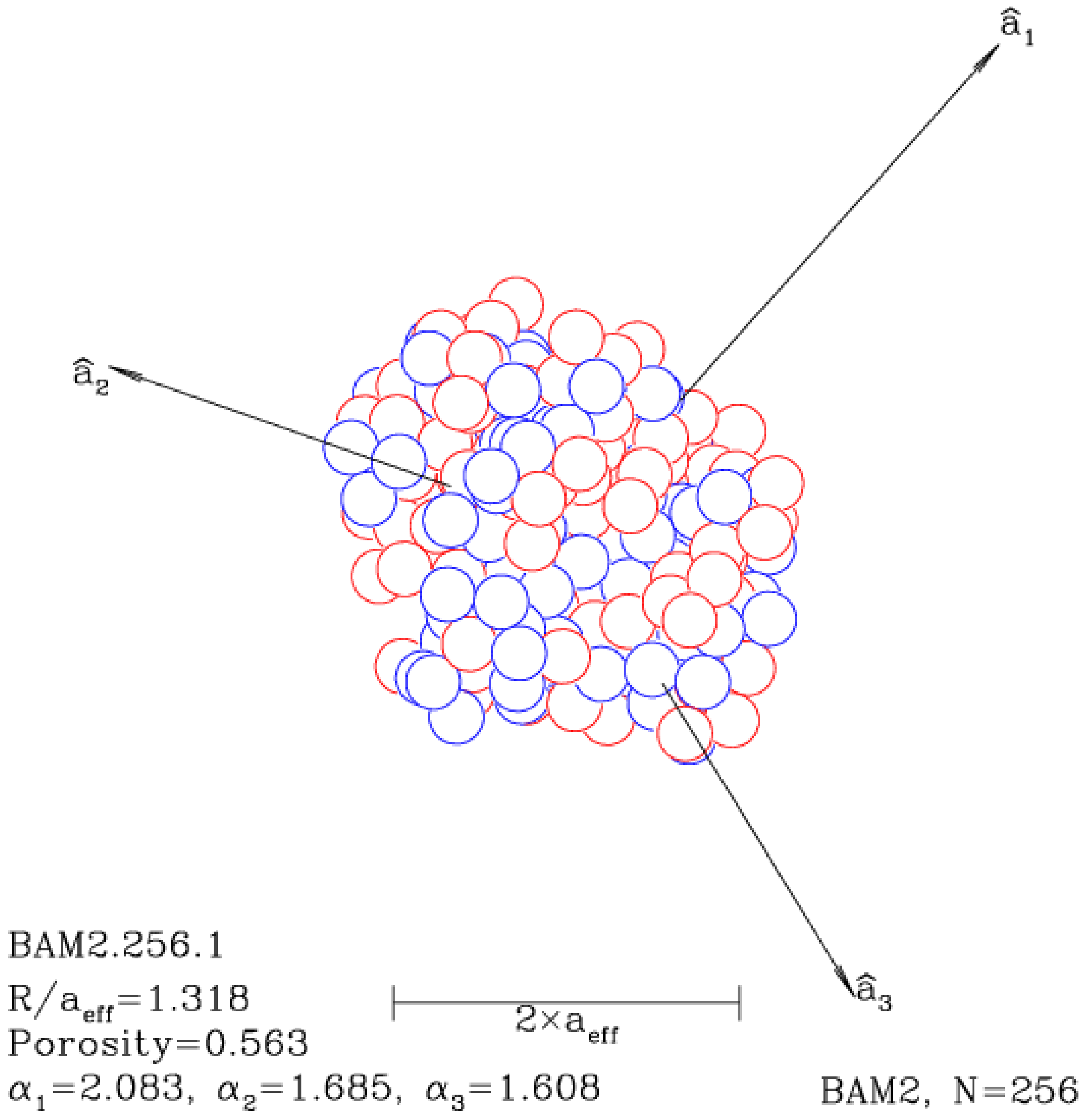

Fig. 8.2: “BAM2” aggregate (least porous); 256 monomers; `BAM2.256.1.targ`

CHAPTER

NINE

USING A CUSTOMIZED GEOMETRY

GGADT supports three common ways to specify the geometry of a grain: either as a sphere, an ellipsoid, or an agglomeration of spheres. If you would like to explore other grain geometries that are not adequately described by these three options, you may program a grain geometry yourself (and then subsequently recompile GGADT).

To do this, you must know Fortran 90/95. Open the `custom.f90` file, and edit the `ior_custom()` function to return the index of refraction at an arbitrary position in the grain. At the moment, the custom grain geometry is set up to be a spherical grain.

```
function ior_custom(x,y,z,en,ephots,ior_res,ior_ims,a)

real(kind=dp_real), intent(in) :: x,y,z,en
real(kind=dp_real), intent(in) :: ephots(:)
real(kind=dp_real), intent(in) :: ior_res(:,:), ior_ims(:,:)
complex(kind=dp_complex) :: ior_custom

! write code for your own grain geometry here.
!
! This function is currently set up to do a sphere
!

! inputs :
!        *  x,y,z
!            cartesian coordinates of the grain in the DDSCAT axis convention
!
!        *  en
!            photon energy at which the index of refraction is requested
!
!        *  ephots
!            an array of photon energies to use for interpolation purposes
!
!        *  ior_res, ior_ims
!            The real and imaginary components of the index of refraction
!
!        *  a
!                        The effective radius of the grain
!
! :note: the ephots and ior_res/ior_ims arrays are organized such that
!
!                  m - 1 = ior_res(i,j) + sqrt(-1) * ior_ims(i,j)
!
!       is the index of refraction (minus one) for material number
!               "i" at a photon energy of ephots(j). Linear interpolation
!               can be used to obtain m-1 for arbitrary photon energies.

real(kind=dp_real), intent(in) :: a
```

```
    if (mag(x,y,z) < a) then
        if (size(ior_res(1,:)) .eq. 1) then
            ior_custom = CMPLX(ior_res(1,1),ior_ims(1,1))
        else
            ior_custom = CMPLX(linterp(ephots,ior_res(1,:),en),&
                                                linterp(ephots,ior_ims(1,:),en) )
        end if
    else
        ior_custom = CMPLX(0.0,0.0)
    end if

end function ior_custom
```

After you have finished writing the function corresponding to the desired grain geometry, you'll need to recompile ggadt by moving to the parent directory and doing:

```
make clean
make
```

Then, set the `--grain-geometry` parameter to "custom" the next time you run GGADT.

CHAPTER

TEN

MULTIPLE COMPOSITIONS

For grains consisting of spherical monomers, GGADT allows you to vary the composition throughout the grain.

1. Create a file that provides the composition of each monomer (e.g. `composition_file.dat`) in the following format:

```
j    material-tag
1              iron
2              carbon
3              iron
4              silicate
...
```

   Here, `iron`, `silicate`, etc. are *material tags*, and `j` refers to the index of the monomer.

2. Tell GGADT which index files these tags refer to by setting

```
agglom-composition-file = /path/to/composition_file.dat

material-tag1                   = iron
material-file1                  = /path/to/iron_index.dat

material-tag2                   = carbon
material-file2                  = /path/to/carbon_index.dat

material-tag3                   = silicate
material-file3                  = /path/to/silicate_index.dat
```

   in your parameter file (or via the command line).

3. Tell GGADT that you're using a `SPHERES` geometry

```
grain-geometry                  = spheres
agglom-file                     = /path/to/target_file.targ
```

CHAPTER

ELEVEN

ANGLE CONVENTIONS

GGADT supports two different angle conventions. You can tell GGADT which one to use by specifying the *--axes-convention* parameter as either `MSTM` or `DDSCAT`.

11.1 DDSCAT

Same conventions that are used in the DDSCAT code:

11.2 MSTM

Same conventions used in the MSTM code:

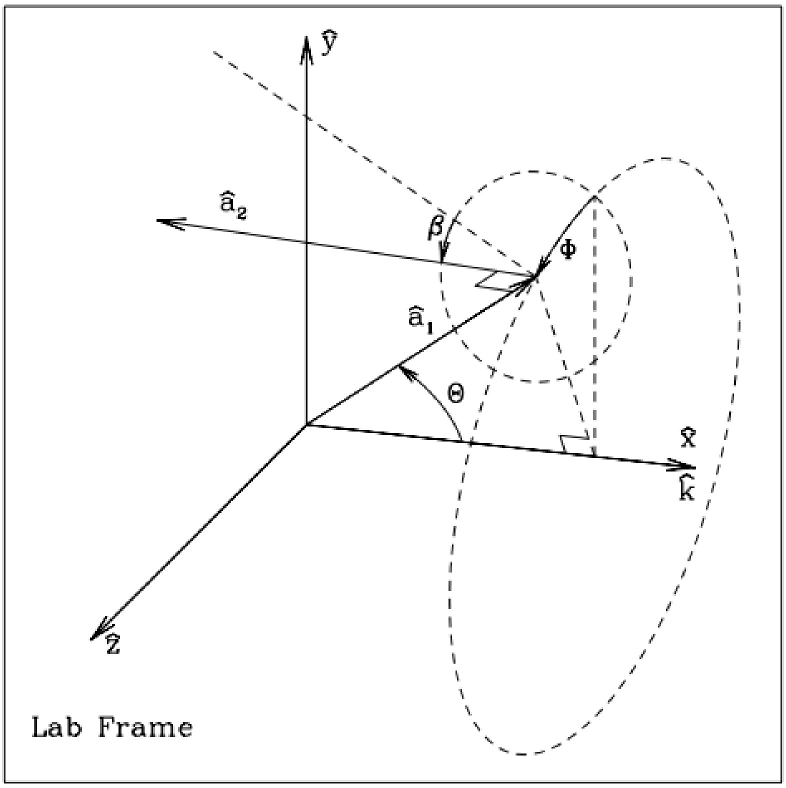

Figure 7: Target orientation in the Lab Frame. $\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\mathrm{LF}}$ is the direction of propagation of the incident radiation, and $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{\mathrm{LF}}$ is the direction of the real component (at $x_{\mathrm{LF}} = 0$, $t = 0$) of the first incident polarization mode. In this coordinate system, the orientation of target axis $\hat{\mathbf{a}}_1$ is specified by angles Θ and Φ. With target axis $\hat{\mathbf{a}}_1$ fixed, the orientation of target axis $\hat{\mathbf{a}}_2$ is then determined by angle β specifying rotation of the target around $\hat{\mathbf{a}}_1$. When $\beta = 0$, $\hat{\mathbf{a}}_2$ lies in the $\hat{\mathbf{a}}_1$,$\hat{\mathbf{x}}_{\mathrm{LF}}$ plane.

Fig. 11.1: DDSCAT conventions

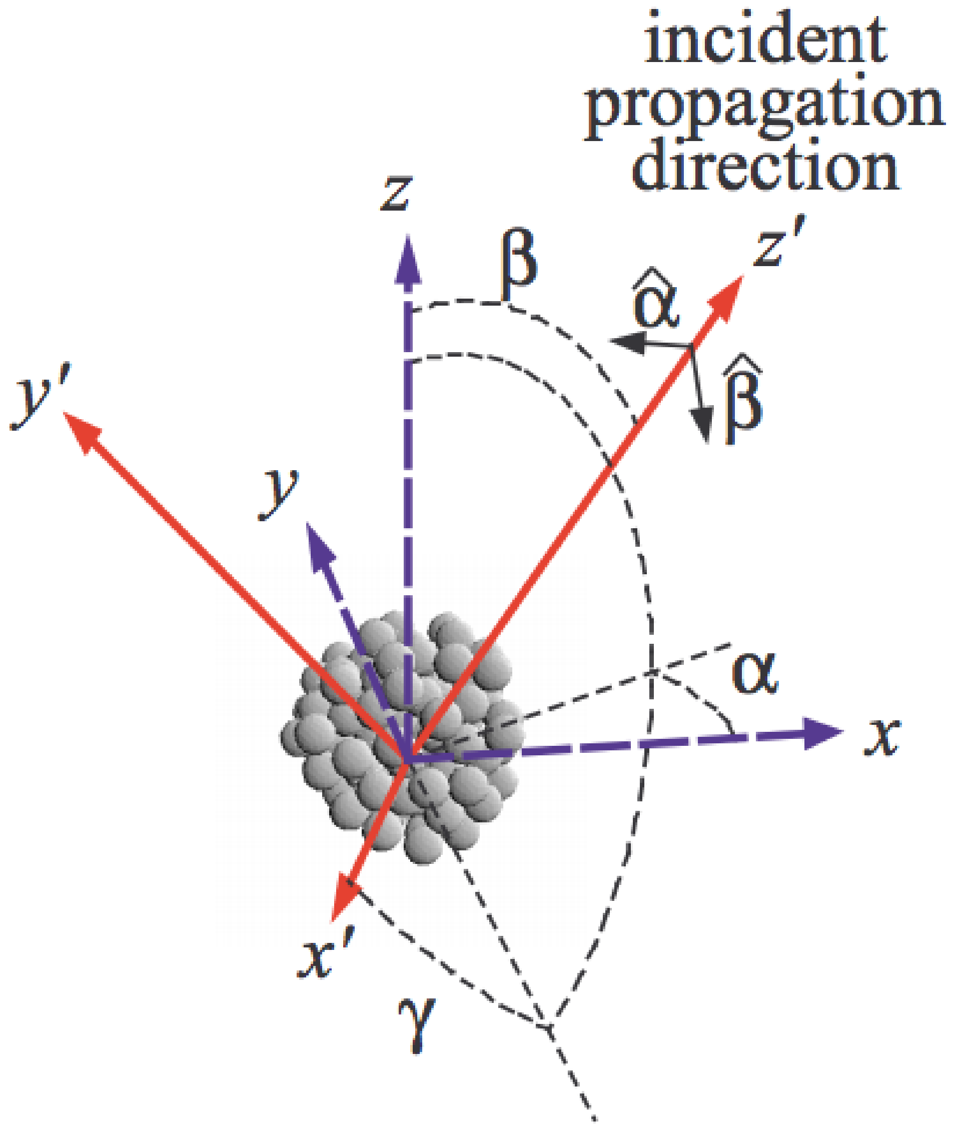

Figure 2: Target and incident field frames

Fig. 11.2: MSTM conventions

CHAPTER TWELVE

ADDITIONAL FILES

To keep GGADT lean, we opted not to include target files and index files (except for those in the `example/` directory). You can download tarballs of both index files and a complete list of target files pulled from http://www.astro.princeton.edu/~draine/agglom.html .

- `Index files` (328 Kb)
- `Target files` (6.6 Mb)

CHAPTER

THIRTEEN

GLOSSARY

material tags Shortname that refers to a particular index file.

optically soft Optically soft refers to materials with an index of refraction very close to 1. $|m - 1| \ll 1$

optically thin Optically thin refers to the case when $\mathrm{Im}[m]ka \ll 1$.

wavenumber The wavenumber is defined to be

$$k \equiv \frac{2\pi}{\lambda}$$

where λ is the wavelength of the light.

effective radius The effective radius of a grain is the radius of a (uniform-density) sphere that has the same mass as that grain:

$$a_{\mathrm{eff}} \equiv (3V/4\pi)^{1/3}$$

index file The index file refers to a text file containing the index of refraction and dielectric function for a given material and for a given number of energies. It must be formatted in a particular way (see `--material-file`)

differential scattering cross section Defined to be

$$\frac{dC_{\mathrm{sca}}}{d\Omega}$$

arcseconds An arcsecond is 1/60 th of an arcminute, which in turn is 1/60 th of a degree.

monomers Grains such as BA, BAM1, and BAM2 grains are composed of many spherical particles, called *monomers*.

CHAPTER

FOURTEEN

INDICES AND TABLES

- genindex
- search

BIBLIOGRAPHY

[DA2006] Draine, B. T., & Allaf-Akbari, K. 2006, ApJ, 652, 1318

[HD2015] Hoffman, J., & Draine, B. T. 2015, *submitted for publication to the Astrophysical Journal*

[VDH1957] van de Hulst H., Light scattering by small particles, 1957, John Wiley & Sons, Inc., NY.

[T1992] Temperton, C. 1992, A Generalized Prime Factor FFT Algorithm for Any $N = 2^p3^q5^r$, SIAM Journal of Scientific and Statistical Computing, 13, 676-686.

[S2003] Shen, Y., Draine, B. T., & Johnson, E. T. 2008, ApJ, 689, 260 ads

Symbols

–aeff=value
  command line option, 20
–agglom-composition-file=value
  command line option, 22
–agglom-file=value
  command line option, 22
–angle-file=value
  command line option, 21
–angle-mode=value
  command line option, 20
–axes-convention=value
  command line option, 20
–dephot=value
  command line option, 23
–do-full-2d-fft
  command line option, 19
–do-phi-averaging
  command line option, 23
–dtheta=value
  command line option, 23
–ephot-max=value
  command line option, 24
–ephot-min=value
  command line option, 23
–ephot=value
  command line option, 23
–force-numerical
  command line option, 19
–grain-axis-x=value, –grain-axis-y=value, –grain-axis-z=value
  command line option, 22
–grain-geometry=value
  command line option, 20
–grid-width=value
  command line option, 20
–help, -h
  command line option, 19
–input-unit-angle=value
  command line option, 21
–input-unit-energy=value
  command line option, 21
–input-unit-length=value
  command line option, 21
–integrated
  command line option, 23
–ior-re=value, –ior-im=value
  command line option, 23
–material-file1=value, –material-file2=value, –material-filen=value
  command line option, 22
–material-file=value
  command line option, 21
–material-tag1=value, –material-tag2=value, –material-tagn=value
  command line option, 22
–max-angle=value
  command line option, 23
–nephots=value
  command line option, 23
–ngrain=value
  command line option, 20
–norientations=value
  command line option, 20
–nphi=value
  command line option, 23
–nscatter=value
  command line option, 23
–nthreads=value
  command line option, 19
–output-sigfigs=value
  command line option, 20
–output-unit-angle=value
  command line option, 21
–output-unit-energy=value
  command line option, 21
–output-unit-length=value
  command line option, 21
–parameter-file=value
  command line option, 20
–quiet, -q
  command line option, 19
–save-file-root=value
  command line option, 20

–save-orientations
    command line option, 19
–save-shadow-function
    command line option, 19
–timing, -t
    command line option, 19
–units-file=value
    command line option, 21
–use-padded-fft, -s
    command line option, 19
–verbose, -d
    command line option, 19
–version, -v
    command line option, 19

A

agglom-file, 14
agglomerates
    BA, 8
    BAM2, 8
angle conventions, 33
angle-mode
    file, 24
    random, 24
    sequential, 24
arcseconds, **41**

B

BA, 35
BAM1, 35
BAM2, 35

C

command line option
    –aeff=value, 20
    –agglom-composition-file=value, 22
    –agglom-file=value, 22
    –angle-file=value, 21
    –angle-mode=value, 20
    –axes-convention=value, 20
    –dephot=value, 23
    –do-full-2d-fft, 19
    –do-phi-averaging, 23
    –dtheta=value, 23
    –ephot-max=value, 24
    –ephot-min=value, 23
    –ephot=value, 23
    –force-numerical, 19
    –grain-axis-x=value, –grain-axis-y=value, –grain-axis-z=value, 22
    –grain-geometry=value, 20
    –grid-width=value, 20
    –help, -h, 19
    –input-unit-angle=value, 21
    –input-unit-energy=value, 21
    –input-unit-length=value, 21
    –integrated, 23
    –ior-re=value, –ior-im=value, 23
    –material-file1=value, –material-file2=value, –material-filen=value, 22
    –material-file=value, 21
    –material-tag1=value, –material-tag2=value, –material-tagn=value, 22
    –max-angle=value, 23
    –nephots=value, 23
    –ngrain=value, 20
    –norientations=value, 20
    –nphi=value, 23
    –nscatter=value, 23
    –nthreads=value, 19
    –output-sigfigs=value, 20
    –output-unit-angle=value, 21
    –output-unit-energy=value, 21
    –output-unit-length=value, 21
    –parameter-file=value, 20
    –quiet, -q, 19
    –save-file-root=value, 20
    –save-orientations, 19
    –save-shadow-function, 19
    –timing, -t, 19
    –units-file=value, 21
    –use-padded-fft, -s, 19
    –verbose, -d, 19
    –version, -v, 19
compiler flags
    FCFLAGS, -O3, -Ofast, 5
compilers
    gcc, gfortran, 5
configure, 5
    datadir, 3

D

DDSCAT, 33
default grain parameters, 17
default.params, 14
differential scattering cross section, 8, 14, **41**

E

effective radius, 1, **41**
efficiencies, 11
example, 8

F

FFT
    FFTW, 3
    GPFA, 3

I

index file, 35, **41**
install, 7
integrated cross sections, 11

M

macports, 8
material file, 32
material tags, 32, **41**
monomers, **41**
MSTM, 33
multiple compositions, 32

N

nphi, 24

O

OpenMP, 8
optically soft, 1, **41**
optically thin, **41**

P

parallelization
    –enable-openmp, OpenMP, 5
parameter file, 8, 14
    parameters_diffscat.ini, 8
    parameters_diffscat_2d.ini, 8
    parameters_total_xs.ini, 8
photon energy, 14
python
    plot_diffscat.py, 8
    plot_diffscat_2d.py, 8
    plot_total_xs.py, 8

R

random orientations, 11
refractive index, 11

T

target file, 35

W

wavenumber, 1, **41**

X

Xcode, 5