ggadt Documentation

Release 1.1.2

John Hoffman, Michael Tarczon

August 31, 2015

Version: 1.1.2 *Last updated*: August 31, 2015

GGADT is short for General Geometry Anomalous Diffraction Theory. It is a Fortran 90 program that uses Anomalous Diffraction Theory (ADT; 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 thin), 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

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

CHAPTER

ONE

CONTENTS

1.1 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 [--enable-openmp] ...
$ make
```

The ggadt binary is now located in the src directory. An optional next step is to do `make install`, which installs the ggadt into a globally-accessible folder (usually `/usr/local/bin`). If you would like to do this, fine, but we prefer a more self-contained setup. We will assume for the remainder of the documentation that you did not do a `make install`.

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`.

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" [--enable-openmp] ...
```

1.1.1 Package Dependencies

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.

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`

1.2 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.

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

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

1.2.1 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:

```
$ ../src/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

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'
```

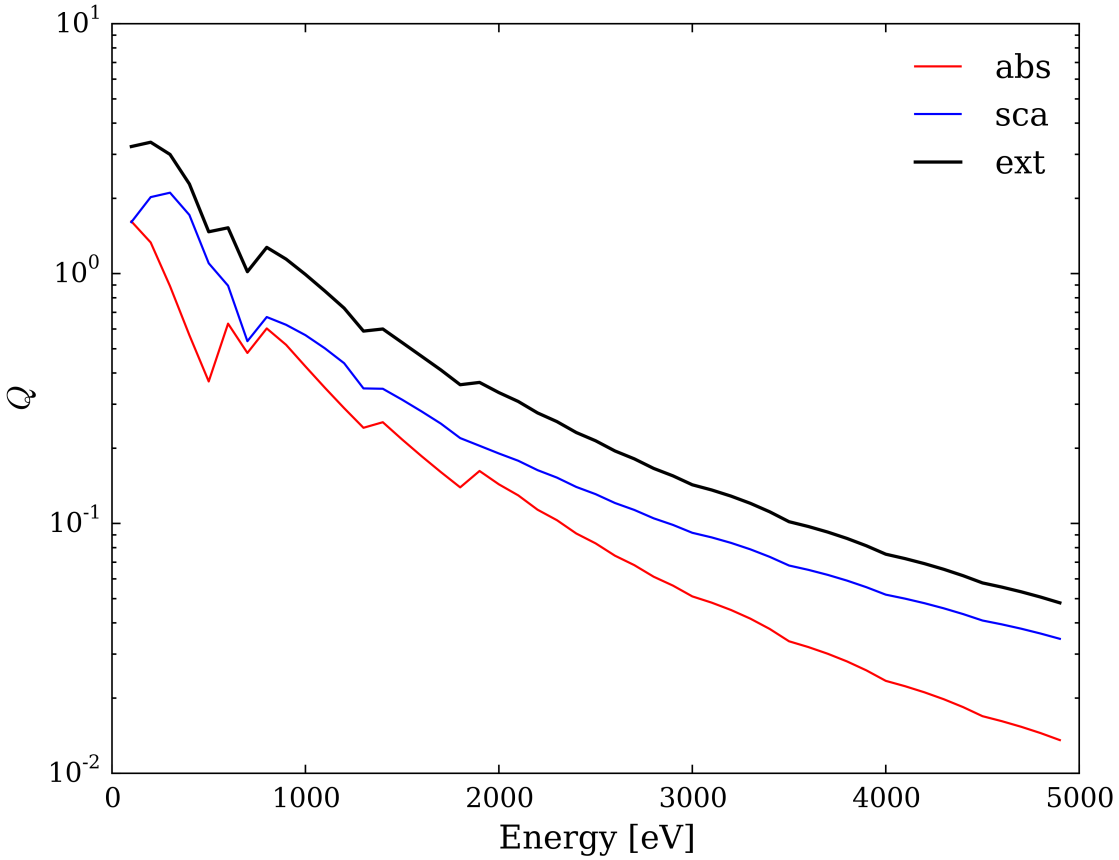


Fig. 1.1: Total cross section (**example 1**)

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:

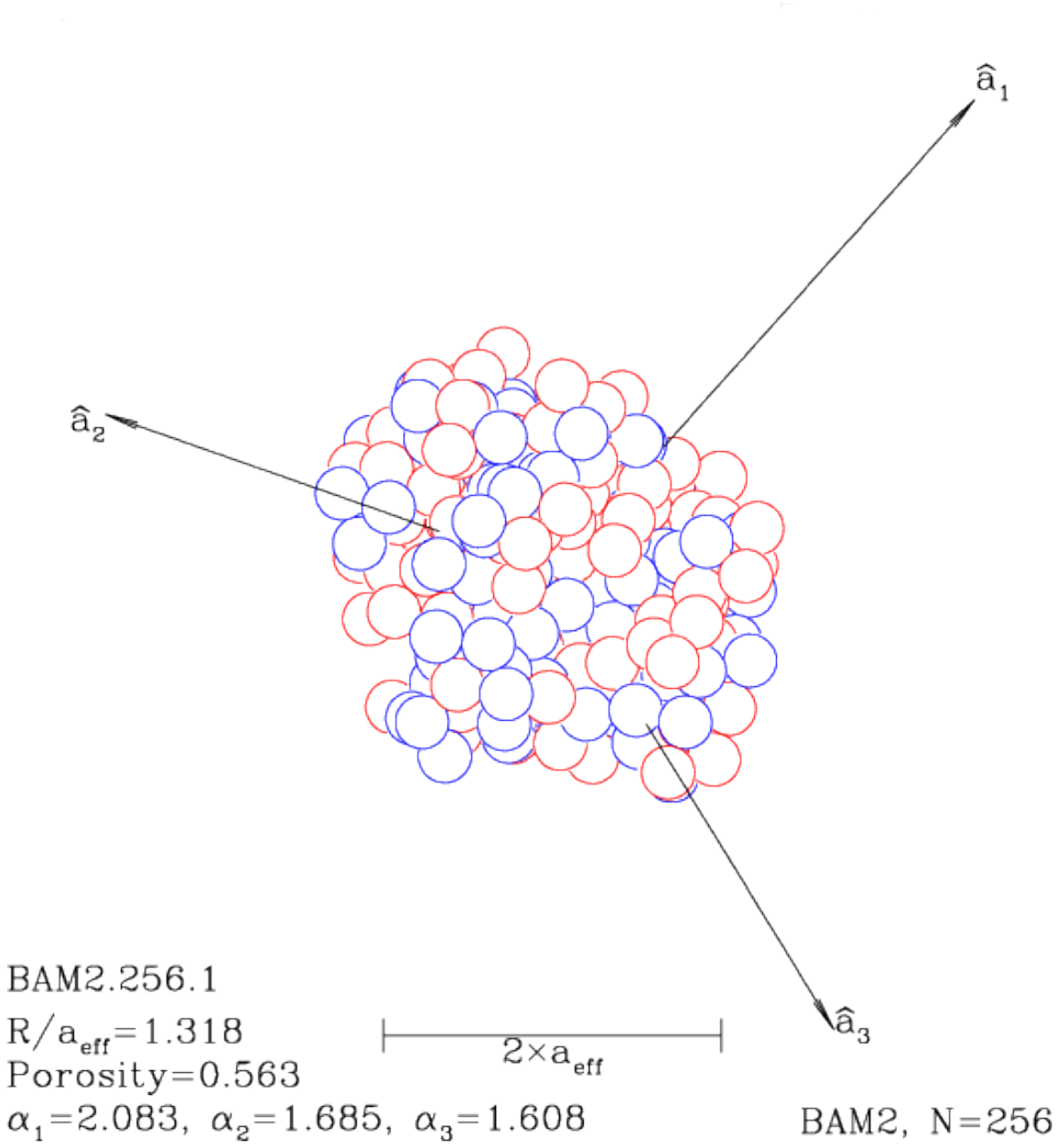


Fig. 1.2: The grain parameterized by `BAM2.256.1.targ`

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
                                              # | calculations.
norientations                   = 64
                                                                                        # If angle-mo
                                              # | over which to average calcuations
```

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

1.2.2 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:

```
$ ../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

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
```

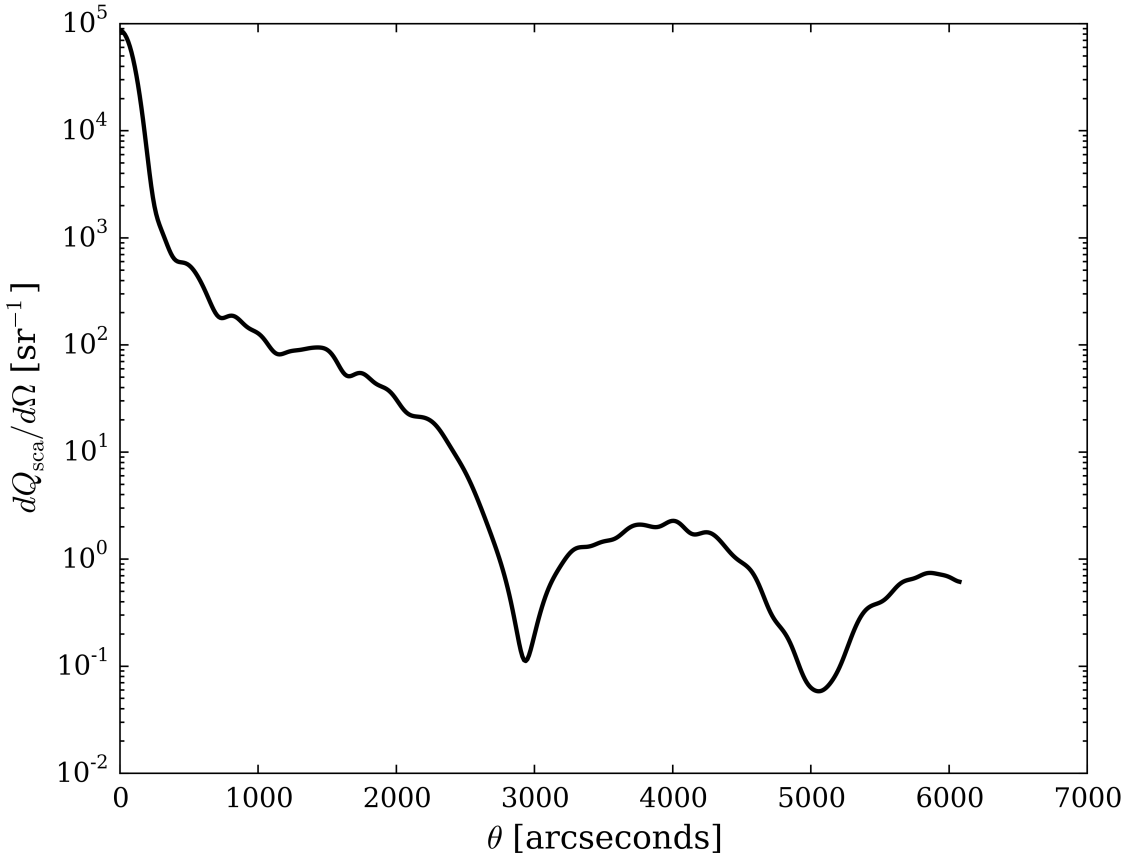


Fig. 1.3: Differential scattering cross section (**example 2**)

```
# | 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 $dC_{\rm sca}/d\Omega$ from 0 to 6000 arcseconds, with values spaced every $\delta\theta = 25$ arcseconds.

1.3 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.

1.3.1 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.

1.3.2 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.

1.4 Arguments and parameters

GGADT supports the following options:

1.4.1 General options

`--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 I/O 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 $d\sigma/d\Omega(\theta)$

--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-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). 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. The effective radius is defined to be the radius of a sphere with the same volume as the dust grain.

--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 a value of 64 is good enough.

--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. These conventions are explained later in the documentation.

--fftw-optimization=value
: Must be one of ESTIMATE, MEASURE, PATIENT or EXHAUSTIVE. For more information about what these tags mean and about FFTW wisdom, see http://fftw.org/fftw3_doc/Words-of-Wisdom_002dSaving-Plans.html#Words-of-Wisdom_002dSaving-Plans.

--angle-mode=value
Must be one of random, sequential, or file. Specifying random will average over norientations grain orientations, with each angle chosen from a uniform distribution over the domain of orientation angles. Specifying sequential will make GGADT use a set of angles that are evenly spaced over each of the orientation angle domains. 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=/path/to/angle-file which contains 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=file.

--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
...
```

1.4.2 Ellipsoidal grains

--grain-axis-x=value
For ELLIPSOID grains, we define the lengths of x, y and z axes.

--grain-axis-y=value

--grain-axis-z=value

1.4.3 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 data/agglomerations), 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.

--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.

--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.

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
    ...
```

1.4.4 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 $dC_{\rm sca}/d\Omega(\theta,\phi)$ over ϕ at each orientation, giving you a 1d result. In certain cases (i.e. for `custom` geometries), this is faster than doing many 1d FFT's.

--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.

1.4.5 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

Default parameter values

Default parameter values are loaded from the `.defaults` 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`.

1.5 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.

1.5.1 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=ddscat`), or (α, β, γ) if `axes-convention=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
...
```

1.5.2 1d vs 2d results

Using the `nphi` parameter

1.6 Sphere

For spherical particles, GGADT uses the following integral expression (see, e.g. *[DA2006]*)

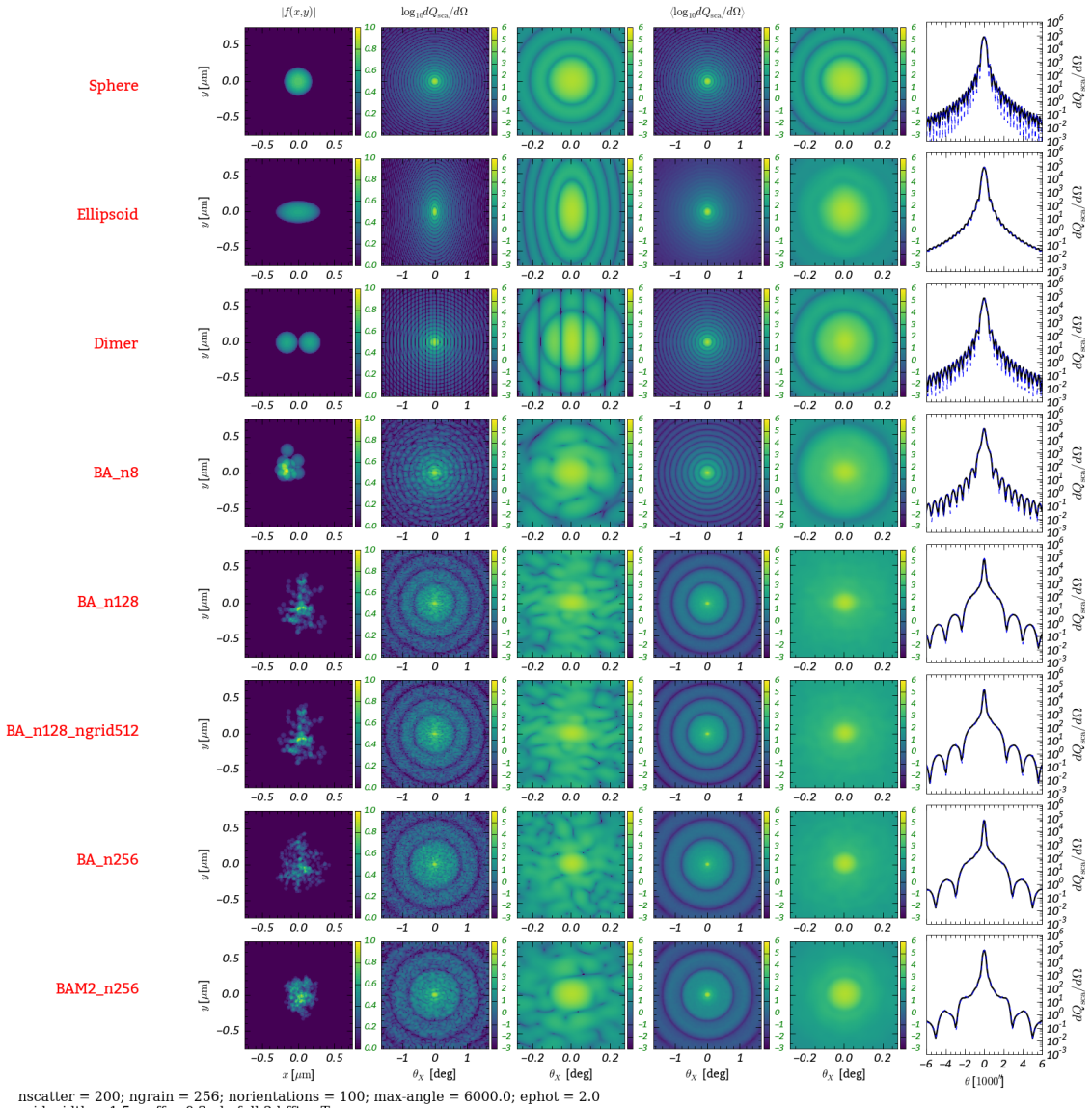
$$S(\theta) = (ka)^2 \int_0^{\pi/2} du J_0(ka \sin\theta \cos u) \left(1 - e^{i\rho \sin u}\right) \sin u \cos u,$$

Here, J_0 is the $n = 0$ Bessel function. By default, GGADT will numerically solve this integral (much faster than solving the general equations). You can force GGADT to use the general solution by passing the `--force-numerical` flag (alternatively, you can set `force-numerical=T` in your parameter file).

1.7 Ellipsoids

1.8 Agglomerations of spherical monomers

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



nscatter = 200; ngrain = 256; norientations = 100; max-angle = 6000.0; ephot = 2.0
grid-width = 1.5; aeff = 0.2; do-full-2d-fft = T

Fig. 1.4: GGADT results for many examples

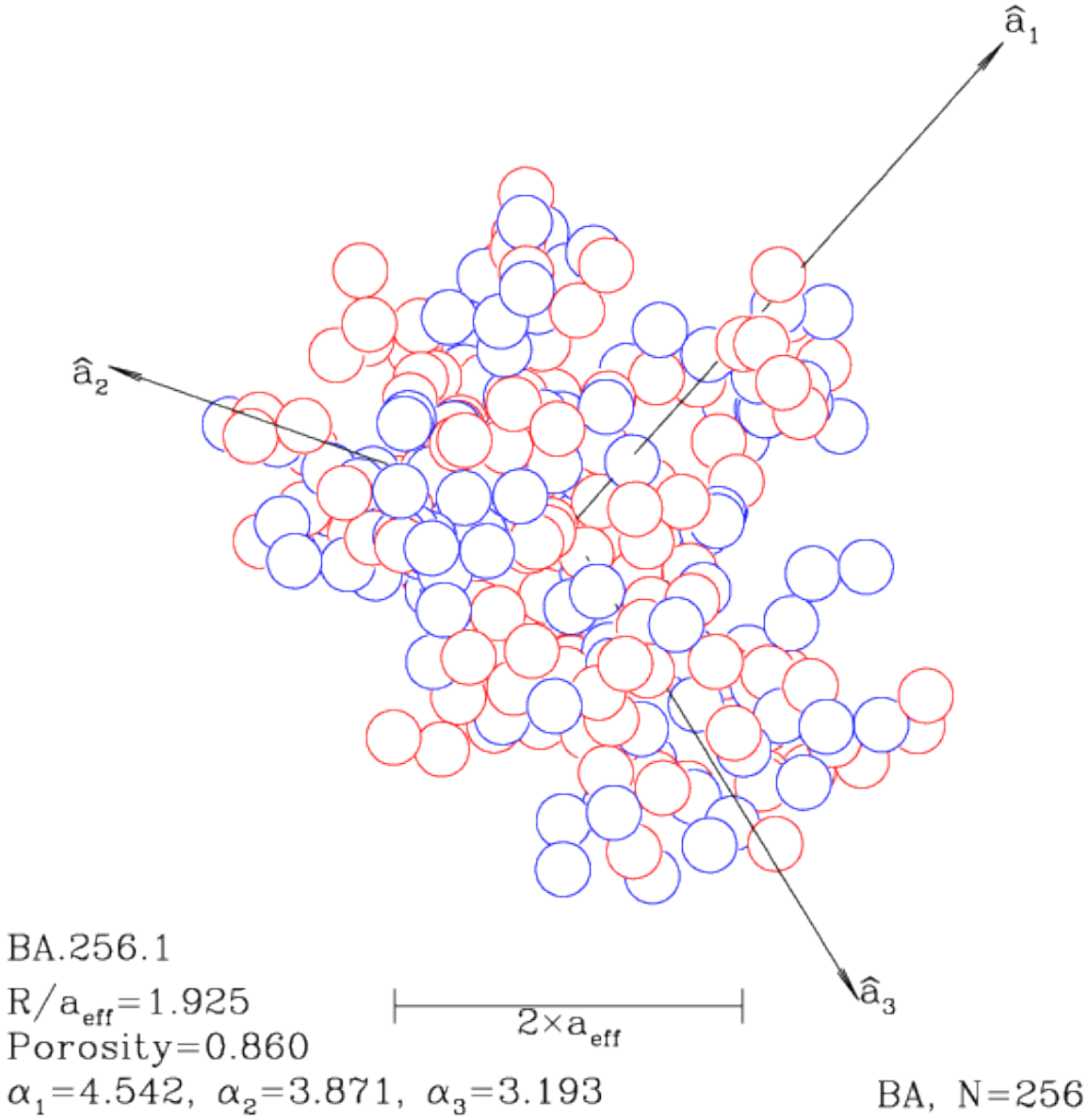


Fig. 1.5: "BA" aggregate (porous); 256 monomers; `BA.256.1.targ`

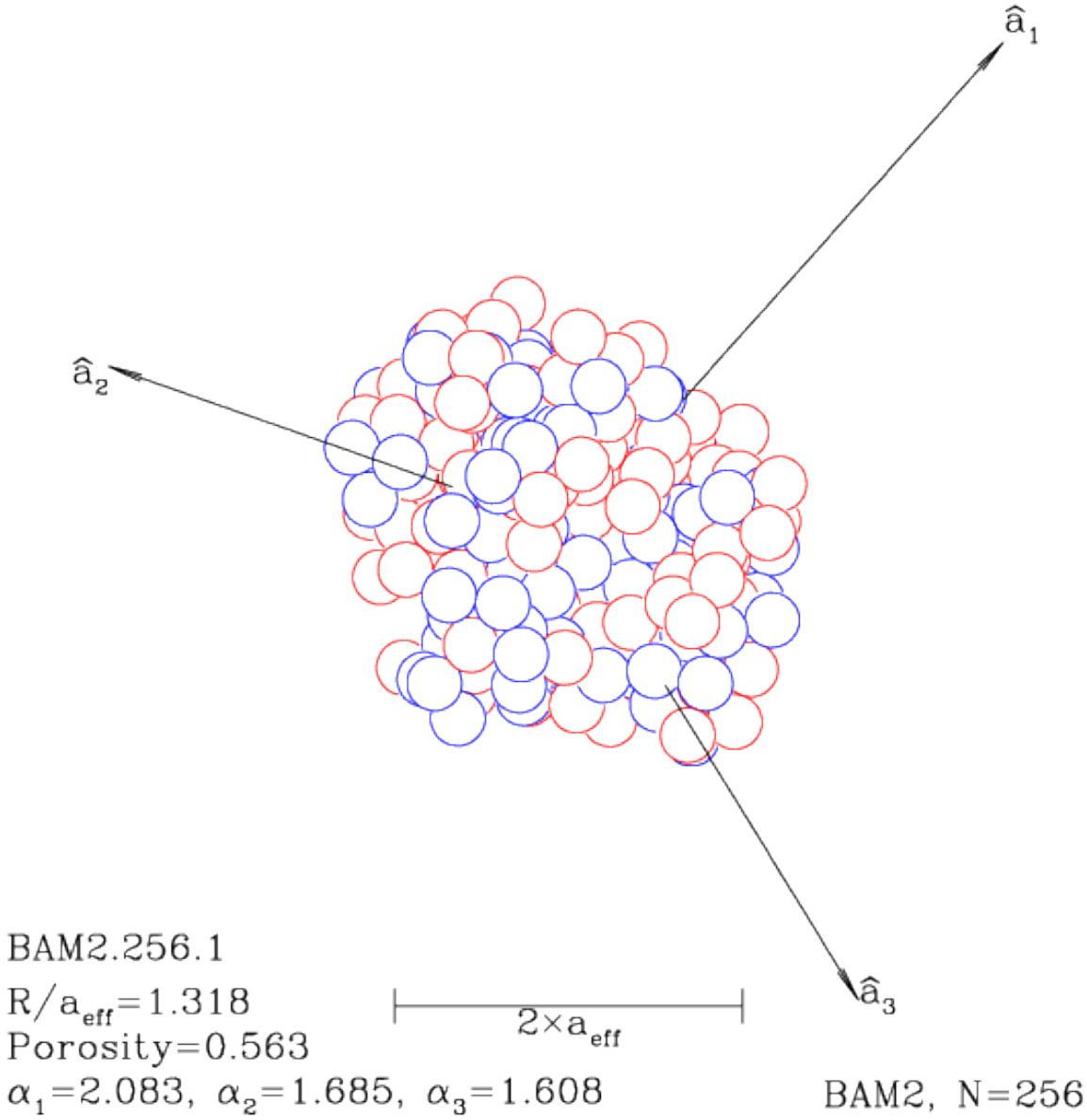


Fig. 1.6: "BAM2" aggregate (least porous); 256 monomers; `BAM2.256.1.targ`

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

1.8.1 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
        ...
```

1.8.2 Multiple compositions

GGADT allows you to set each monomer to have a different composition; for instance, some monomers might be iron particles, some might be carbon, and some might be silicate.

1.9 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 for
    !                              the material(s) specified.
    !
    !          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.

1.10 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`.

1.10.1 DDSCAT

Same conventions that are used in the DDSCAT code:

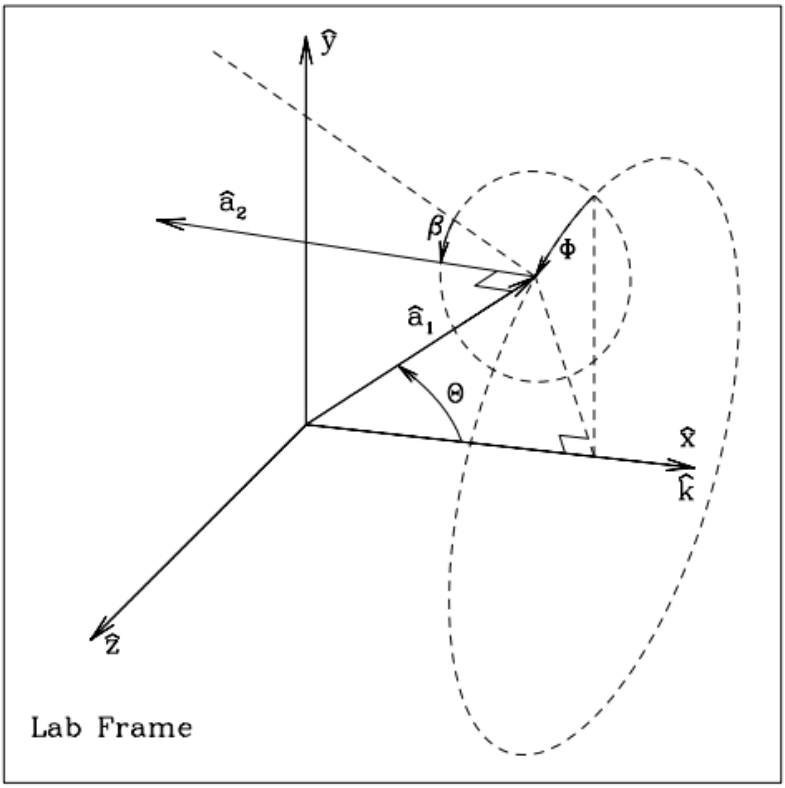


Figure 7: Target orientation in the Lab Frame. $\hat{\mathbf{x}} = \hat{\mathbf{x}}_{\rm LF}$ is the direction of propagation of the incident radiation, and $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{\rm LF}$ is the direction of the real component (at $x_{\rm LF} = 0$, $t = 0$) of the first incident polarization 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 $\hat{\mathbf{a}}_1$,$\hat{\mathbf{x}}_{\rm LF}$ plane.

Fig. 1.7: DDSCAT conventions

1.10.2 MSTM

Same conventions used in the MSTM code:

1.11 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)

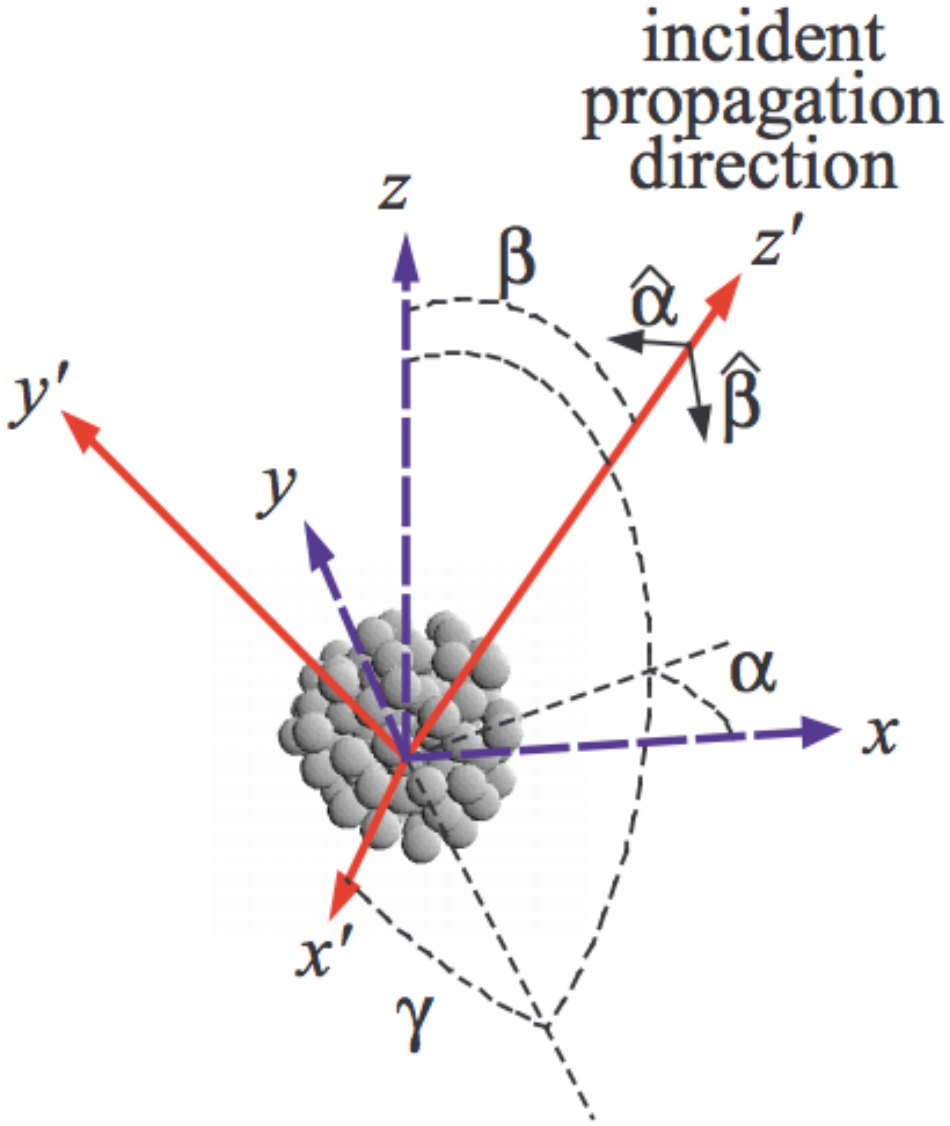


Figure 2: Target and incident field frames

Fig. 1.8: MSTM conventions

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

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

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

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

Symbols

C