This document can be cited as: Draine, B.T., and Flatau, P.J. 2003, "User Guide for the Discrete Dipole Approximation Code DDSCAT.6.0", http://arxiv.org/abs/astro-ph/0309069

User Guide for the Discrete Dipole Approximation Code DDSCAT.6.0

Bruce T. Draine
Princeton University Observatory
Princeton NJ 08544-1001
(draine@astro.princeton.edu)
and

Piotr J. Flatau
University of California, San Diego
Scripps Institution of Oceanography
La Jolla CA 92093-0221
(pflatau@ucsd.edu)

last revised: 2003 September 2

Abstract

DDSCAT.6.0 is a freely available software package which applies the "discrete dipole approximation" (DDA) to calculate scattering and absorption of electromagnetic waves by targets with arbitrary geometries and complex refractive index. The DDA approximates the target by an array of polarizable points. **DDSCAT.6.0** allows accurate calculations of electromagnetic scattering from targets with "size parameters" $2\pi a/\lambda < 15$ provided the refractive index m is not large compared to unity ($|m-1| < 1$). **DDSCAT.6.0** includes the option of using the `FFTW` (Fastest Fourier Transform in the West) package. **DDSCAT.6.0** also includes support for `MPI` (Message Passing Interface), permitting parallel calculations on multiprocessor systems.

The **DDSCAT** package is written in Fortran and is highly portable. The program supports calculations for a variety of target geometries (e.g., ellipsoids, regular tetrahedra, rectangular solids, finite cylinders, hexagonal prisms, etc.). Target materials may be both inhomogeneous and anisotropic. It is straightforward for the user to "import" arbitrary target geometries into the code, and relatively straightforward to add new target generation capability to the package. **DDSCAT** automatically calculates total cross sections for absorption and scattering and selected elements of the Mueller scattering intensity matrix for specified orientation of the target relative to the incident wave, and for specified scattering directions.

This User Guide explains how to use **DDSCAT.6.0** to carry out electromagnetic scattering calculations. CPU and memory requirements are described.

Contents

1 Introduction **4**

2 Applicability of the DDA **5**

3 DDSCAT.6.0 **6**
3.1 What Does It Calculate? 6
3.2 Application to Targets in Dielectric Media 9

4 What's New? **10**

5 Obtaining the Source Code **10**

6 Compiling and Linking **11**
6.1 Device Numbers `IDVOUT` and `IDVERR` 11
6.2 Subroutine `TIMEIT` 12
6.3 Optimization 12
6.4 Leaving `FFTW` Capability Disabled 12
6.5 Enabling `FFTW` Capability 13
6.6 Leaving the netCDF Capability Disabled 13
6.7 Enabling the netCDF Capability 14
6.8 Compiling and Linking... 14
6.9 Installation on Non-Unix Systems 14

7 Moving the Executable **15**

8 The Parameter File `ddscat.par` **16**

9 Running DDSCAT.6.0 Using the Sample `ddscat.par` File **16**

10 Output Files **17**
10.1 ASCII files 17
10.2 Binary Option 18
10.3 Machine-Independent Binary File Option: netCDF 18

11 Dipole Polarizabilities **19**

12 Dielectric Functions **19**

13 Choice of FFT Algorithm **19**

14 Choice of Iterative Algorithm **22**

15 Calculation of Radiative Force and Torque **23**

16 Memory Requirements **24**

17 Target Orientation **24**
17.1 Orientation of the Target in the Lab Frame 25
17.2 Orientation of the Incident Beam in the Target Frame 25
17.3 Sampling in Θ, Φ, and β 27

18 Orientational Averaging **28**
18.1 Randomly-Oriented Targets 28
18.2 Nonrandomly-Oriented Targets 28

19 Target Generation **29**

20 Scattering Directions **33**

21 Incident Polarization State **33**

22 Averaging over Scattering: $g(1) = \langle\cos\theta_s\rangle$, etc. **34**

23 Mueller Matrix for Scattering in Selected Directions **34**
23.1 Two Orthogonal Incident Polarizations (`IORTH=2`) 34
23.2 Stokes Parameters 36
23.3 Relation Between Stokes Parameters of Incident and Scattered Radiation: The Mueller Matrix 36
23.4 One Incident Polarization State Only (`IORTH=1`) 37

24 Using MPI (Message Passing Interface) **38**
24.1 Source Code for the MPI-compatible version of DDSCAT 38
24.2 Compiling and Linking the MPI-compatible version of DDSCAT 39
24.3 Code Execution Under MPI 39

25 Graphics and Postprocessing **40**
25.1 IDL 40

26 Miscellanea **40**

27 Finale **40**

28 Acknowledgments **41**

A Understanding and Modifying `ddscat.par` **43**

B `w`*xx*`r`*yy*`ori.avg` Files **45**

C `w`*xx*`r`*yy*`k`*zzz*`.sca` Files **46**

1 Introduction

DDSCAT.6.0 is a Fortran software package to calculate scattering and absorption of electromagnetic waves by targets with arbitrary geometries using the "discrete dipole approximation" (DDA). In this approximation the target is replaced by an array of point dipoles (or, more precisely, polarizable points); the electromagnetic scattering problem for an incident periodic wave interacting with this array of point dipoles is then solved essentially exactly. The DDA (sometimes referred to as the "coupled dipole approximation") was apparently first proposed by Purcell & Pennypacker (1973). DDA theory was reviewed and developed further by Draine (1988), Draine & Goodman (1993), and recently reviewed by Draine & Flatau (1994) and Draine (2000).

DDSCAT.6.0 is a Fortran implementation of the DDA developed by the authors. It is intended to be a versatile tool, suitable for a wide variety of applications ranging from interstellar dust to atmospheric aerosols. As provided, **DDSCAT.6.0** should be usable for many applications without modification, but the program is written in a modular form, so that modifications, if required, should be fairly straightforward.

The authors make this code openly available to others, in the hope that it will prove a useful tool. We ask only that:

- If you publish results obtained using **DDSCAT**, please consider acknowledging the source of the code.
- If you discover any errors in the code or documentation, please promptly communicate them to the authors.
- You comply with the "copyleft" agreement (more formally, the GNU General Public License) of the Free Software Foundation: you may copy, distribute, and/or modify the software identified as coming under this agreement. If you distribute copies of this software, you must give the recipients all the rights which you have. See the file `doc/copyleft` distributed with the DDSCAT software.

We also strongly encourage you to send email to the authors identifying yourself as a user of DDSCAT; this will enable the authors to notify you of any bugs, corrections, or improvements in DDSCAT.

The current version, **DDSCAT.6.0**, uses the DDA formulae from Draine (1988), with dipole polarizabilities determined from the Lattice Dispersion Relation (Draine & Goodman 1993). The code incorporates Fast Fourier Transform (FFT) methods (Goodman, Draine, & Flatau 1991).

This User Guide assumes that you have already obtained the Fortran source code for **DDSCAT.6.0** either by download from `http://www.astro.princeton.edu/~draine` or by following the instructions in the `README` file.[1] We refer you to the list of references at the end of this document for discussions of the theory and accuracy of the DDA [first see the recent reviews by Draine and Flatau (1994) and Draine (2000)]. In §4 we describe the principal changes between **DDSCAT.6.0** and the

[1] To obtain the `README` file:
(1) anonymous `ftp` to `astro.princeton.edu` ,
(2) `cd draine/scat/DDA/ver6`, and
(3) `get README`.

previous releases.[2] The succeeding sections contain instructions for:

- compiling and linking the code;
- running a sample calculation;
- understanding the output from the sample calculation;
- modifying the parameter file to do your desired calculations;
- specifying target orientation;
- changing the `DIMENSION`ing of the source code to accommodate your desired calculations.

The instructions for compiling, linking, and running will be appropriate for a UNIX system; slight changes will be necessary for non-UNIX sites, but they are quite minor and should present no difficulty.

Finally, the current version of this User Guide can be obtained from `http://arxiv.org/abs/astro-ph/0008151` – you will be offered the options of downloading

- Latex source
- Postscript
- Other formats – click on this to obtain the UserGuide as a PDF file.

2 Applicability of the DDA

The principal advantage of the DDA is that it is completely flexible regarding the geometry of the target, being limited only by the need to use an interdipole separation d small compared to (1) any structural lengths in the target, and (2) the wavelength λ. Numerical studies (Draine & Goodman 1993; Draine & Flatau 1994; Draine 2000) indicate that the second criterion is adequately satisfied if

$$|m|kd < 1 \quad , \tag{1}$$

where m is the complex refractive index of the target material, and $k \equiv 2\pi/\lambda$, where λ is the wavelength *in vacuo*. However, if accurate calculations of the scattering phase function (e.g., radar or lidar cross sections) are desired, a more conservative criterion

$$|m|kd < 0.5 \tag{2}$$

will ensure that differential scattering cross sections $dC_{\rm sca}/d\Omega$ are accurate to within a few percent of the average differential scattering cross section $C_{\rm sca}/4\pi$ (see Draine 2000).

Let V be the target volume. If the target is represented by an array of N dipoles, located on a cubic lattice with lattice spacing d, then

$$V = Nd^3 \quad . \tag{3}$$

We characterize the size of the target by the "effective radius"

$$a_{\rm eff} \equiv (3V/4\pi)^{1/3} \quad , \tag{4}$$

[2]The previous "official releases" were

- **DDSCAT.4b**,
- **DDSCAT.4c** –while never announced, `DDSCAT.4c` was made available to a number of interested users.
- **DDSCAT.5a7**
- **DDSCAT.5a8**
- **DDSCAT.5a9**
- **DDSCAT.5a10**

the radius of an equal volume sphere. A given scattering problem is then characterized by the dimensionless "size parameter"

$$x \equiv ka_{\rm eff} = \frac{2\pi a_{\rm eff}}{\lambda} \quad . \tag{5}$$

The size parameter can be related to N and $|m|kd$:

$$x \equiv \frac{2\pi a_{\rm eff}}{\lambda} = \frac{62.04}{|m|}\left(\frac{N}{10^6}\right)^{1/3} \cdot |m|kd \quad . \tag{6}$$

Equivalently, the target size can be written

$$a_{\rm eff} = 9.873\frac{\lambda}{|m|}\left(\frac{N}{10^6}\right)^{1/3} \cdot |m|kd \quad . \tag{7}$$

Practical considerations of CPU speed and computer memory currently available on scientific workstations typically limit the number of dipoles employed to $N < 10^6$ (see §16 for limitations on N due to available RAM); for a given N, the limitations on $|m|kd$ translate into limitations on the ratio of target size to wavelength.

For calculations of total cross sections $C_{\rm abs}$ and $C_{\rm sca}$, we require $|m|kd < 1$:

$$a_{\rm eff} < 9.88\frac{\lambda}{|m|}\left(\frac{N}{10^6}\right)^{1/3} \quad \text{or} \quad x < \frac{62.04}{|m|}\left(\frac{N}{10^6}\right)^{1/3} \quad . \tag{8}$$

For scattering phase function calculations, we require $|m|kd < 0.5$:

$$a_{\rm eff} < 4.94\frac{\lambda}{|m|}\left(\frac{N}{10^6}\right)^{1/3} \quad \text{or} \quad x < \frac{31.02}{|m|}\left(\frac{N}{10^6}\right)^{1/3} \quad . \tag{9}$$

It is therefore clear that the DDA is not suitable for very large values of the size parameter x, or very large values of the refractive index m. The primary utility of the DDA is for scattering by dielectric targets with sizes comparable to the wavelength. As discussed by Draine & Goodman (1993), Draine & Flatau (1994), and Draine (2000), total cross sections calculated with the DDA are accurate to a few percent provided $N > 10^4$ dipoles are used, criterion (1) is satisfied, and $|m - 1| < 2$.

Examples illustrating the accuracy of the DDA are shown in Figs. 1–2 which show overall scattering and absorption efficiencies as a function of wavelength for different discrete dipole approximations to a sphere, with N ranging from 304 to 59728. The DDA calculations assumed radiation incident along the (1,1,1) direction in the "target frame". Figs. 3–4 show the scattering properties calculated with the DDA for $x = ka = 7$. Additional examples can be found in Draine & Flatau (1994) and Draine (2000).

3 DDSCAT.6.0

3.1 What Does It Calculate?

DDSCAT.6.0, like previous versions of **DDSCAT**, solves the problem of scattering and absorption by an array of polarizable point dipoles interacting with a monochromatic plane wave incident from infinity. **DDSCAT.6.0** has the capability of automatically generating dipole array representations for a variety of target geometries (see §19) and can also accept dipole array representations of targets supplied by the user (although the dipoles must be located on a cubic lattice). The incident plane wave can have arbitrary elliptical polarization (see §21), and the target can be arbitrarily oriented relative to the incident radiation (see §17). The following quantities are calculated by **DDSCAT.6.0** :

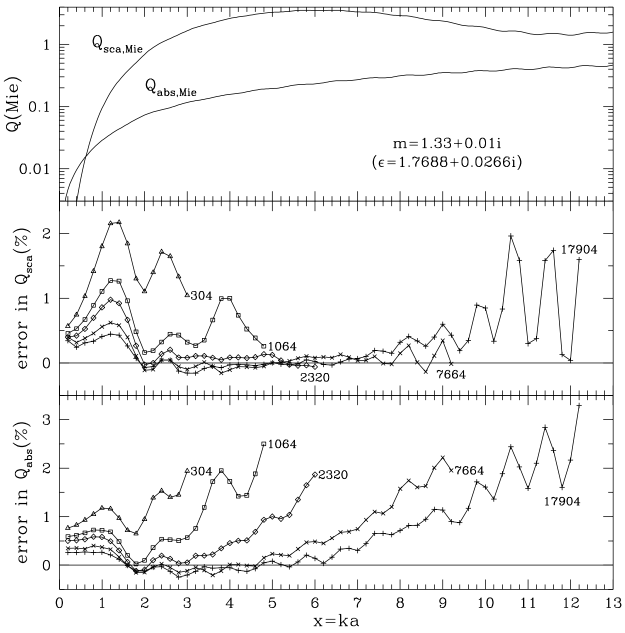

Figure 1: Scattering and absorption for a sphere with $m = 1.33 + 0.01i$. The upper panel shows the exact values of $Q_{\rm sca}$ and $Q_{\rm abs}$, obtained with Mie theory, as functions of $x = ka$. The middle and lower panels show fractional errors in $Q_{\rm sca}$ and $Q_{\rm abs}$, obtained using **DDSCAT** with polarizabilities obtained from the Lattice Dispersion Relation, and labelled by the number N of dipoles in each pseudosphere. After Fig. 1 of Draine & Flatau (1994).

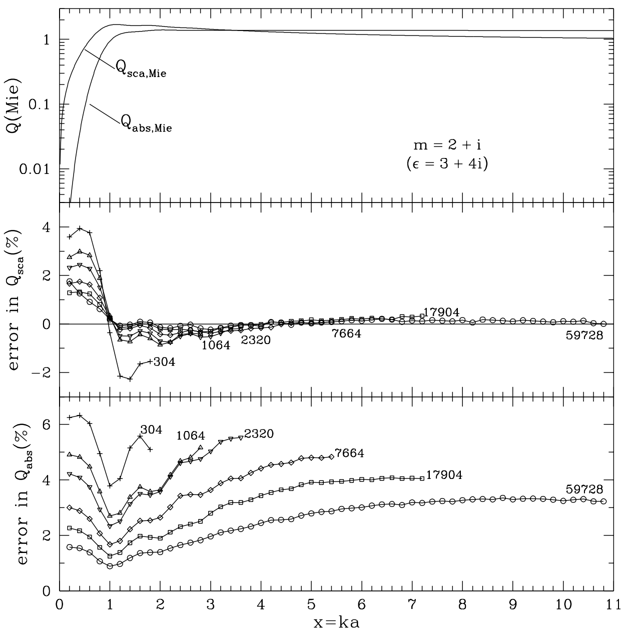

Figure 2: Same as Fig. 1, but for $m = 2 + i$. After Fig. 2 of Draine & Flatau (1994).

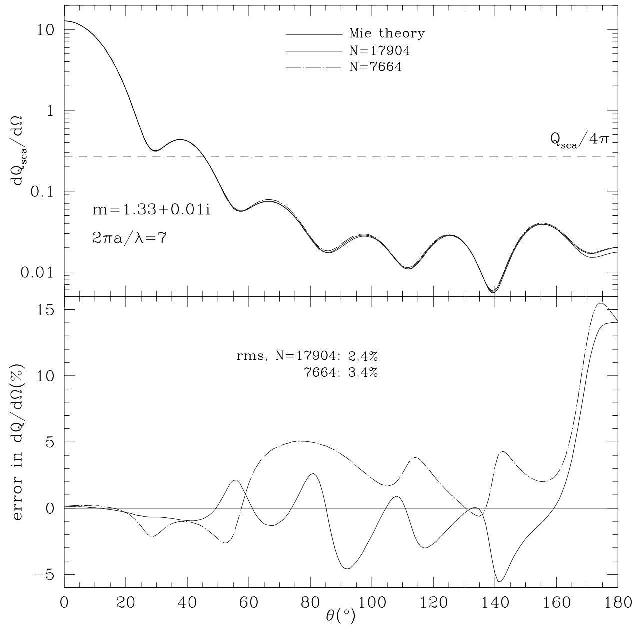

Figure 3: Differential scattering cross section for $m = 1.33 + 0.01i$ pseudosphere and $ka = 7$. Lower panel shows fractional error compared to exact Mie theory result. The $N = 17904$ pseudosphere has $|m|kd = 0.57$, and an rms fractional error in $d\sigma/d\Omega$ of 2.4%. After Fig. 5 of Draine & Flatau (1994).

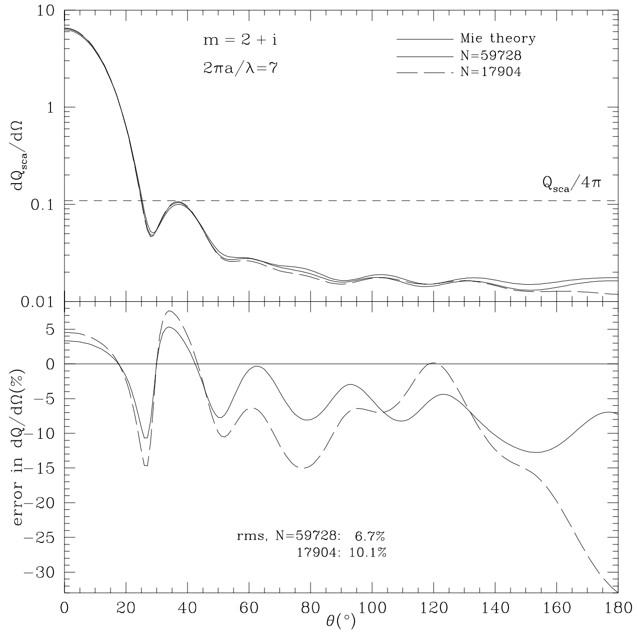

Figure 4: Same as Fig. 3 but for $m = 2 + i$. The $N = 59728$ pseudosphere has $|m|kd = 0.65$, and an rms fractional error in $d\sigma/d\Omega$ of 6.7%. After Fig. 8 of Draine & Flatau (1994).

- Absorption efficiency factor $Q_{\rm abs} \equiv C_{\rm abs}/\pi a_{\rm eff}^2$, where $C_{\rm abs}$ is the absorption cross section;
- Scattering efficiency factor $Q_{\rm sca} \equiv C_{\rm sca}/\pi a_{\rm eff}^2$, where $C_{\rm sca}$ is the scattering cross section;
- Extinction efficiency factor $Q_{\rm ext} \equiv Q_{\rm sca} + Q_{\rm abs}$;
- Phase lag efficiency factor $Q_{\rm pha}$, defined so that the phase-lag (in radians) of a plane wave after propagating a distance L is just $n_t Q_{\rm pha} \pi a_{\rm eff}^2 L$, where n_t is the number density of targets.
- The 4×4 Mueller scattering intensity matrix S_{ij} describing the complete scattering properties of the target for scattering directions specified by the user.
- Radiation force efficiency vector $\mathbf{Q}_{\rm rad}$ (see §15).
- Radiation torque efficiency vector $\mathbf{Q}_\Gamma$ (see §15).

3.2 Application to Targets in Dielectric Media

Let ω be the angular frequency of the incident radiation. For many applications, the target is essentially *in vacuo*, in which case the dielectric function ϵ which the user should supply to `DDSCAT` is the actual complex dielectric function $\epsilon_{\rm target}(\omega)$, or complex refractive index $m_{\rm target}(\omega) = \sqrt{\epsilon_{\rm target}}$ of the target material.

However, for many applications of interest (e.g., marine optics, or biological optics) the "target" body is embedded in a (nonabsorbing) dielectric medium, with (real) dielectric function $\epsilon_{\rm medium}(\omega)$, or (real) refractive index $m_{\rm medium}(\omega) = \sqrt{\epsilon_{\rm medium}}$. **DDSCAT** is fully applicable to these scattering problems, except that:

- The "dielectric function" or "refractive index" supplied to **DDSCAT** should be the *relative* dielectric function
$$\epsilon(\omega) = \frac{\epsilon_{\rm target}(\omega)}{\epsilon_{\rm medium}(\omega)} \tag{10}$$
or *relative* refractive index:
$$m(\omega) = \frac{m_{\rm target}(\omega)}{m_{\rm medium}(\omega)}. \tag{11}$$
- The wavelength λ specified in `ddscat.par` should be the wavelength *in the medium*:
$$\lambda = \frac{\lambda_{vac}}{m_{\rm medium}}, \tag{12}$$
where $\lambda_{vac} = 2\pi c/\omega$ is the wavelength *in vacuo*.

The absorption, scattering, extinction, and phase lag efficiency factors $Q_{\rm abs}$, $Q_{\rm sca}$, and $Q_{\rm ext}$ calculated by **DDSCAT** will then be equal to the physical cross sections for absorption, scattering, and extinction divided by $\pi a_{\rm eff}^2$ – e.g., the attenuation coefficient for radiation propagating through a medium with a density n_t of scatterers will be just $\alpha = n_t Q_{\rm ext} \pi a_{\rm eff}^2$. Similarly, the phase lag (in radians) after propagating a distance L will be $n_t Q_{\rm pha} \pi a_{\rm eff}^2$.

The elements S_{ij} of the 4×4 Mueller scattering matrix $\mathbf{S}$ calculated by **DDSCAT** will be correct for scattering in the medium:
$$\mathbf{I}_{\rm sca} = \left(\frac{\lambda}{2\pi r}\right)^2 \mathbf{S} \cdot \mathbf{I}_{in}, \tag{13}$$
where $\mathbf{I}_{in}$ and $\mathbf{I}_{\rm sca}$ are the Stokes vectors for the incident and scattered light (in the medium), r is the distance from the target, and λ is the wavelength in the medium (eq. 12). See §23 for a detailed discussion of the Mueller scattering matrix.

The time-averaged radiative force and torque (see §15) on a target in a dielectric medium are
$$\mathbf{F}_{\rm rad} = \mathbf{Q}_{pr} \pi a_{\rm eff}^2 u_{\rm rad} \quad , \tag{14}$$

$$\mathbf{\Gamma}_{\rm rad} = \mathbf{Q}_\Gamma \pi a_{\rm eff}^2 u_{\rm rad} \frac{\lambda}{2\pi} \quad , \tag{15}$$

where the time-averaged energy density is

$$u_{\rm rad} = \epsilon_{\rm medium} \frac{E_0^2}{8\pi} \quad , \tag{16}$$

where $E_0 \cos(\omega t + \phi)$ is the electric field of the incident plane wave in the medium.

4 What's New?

DDSCAT.6.0 differs from **DDSCAT.5a** in the following ways:

1. Via the parameter file `ddscat.par`, the user can now select up to 9 elements of the Muller scattering matrix S_{ij} to be printed out.
2. The maximum number of iterations allowed has been increased from 300 to 10000, since **DDSCAT** is now increasingly employed for computations that converge relatively slowly: large numbers of dipoles, large values of the scattering parameter, or large values of the refractive index. The number of iterations used for a solution is recorded in the `ww`aa`r`bb`k`ccc`.sca` output files.
3. A new target option is available: `LYRSLB` (a rectangular slab with up to 4 different material layers).
4. User must now specify (in `ddscat.par` which `IWAV`, `IRAD`, `IWAV` to start with. The "default" choice will be `0 0 0`, but when computations have been interupted by, e.g., power outage, there will be occasions when it is convenient to be able to resume with the first incomplete calculation.
5. A new FFT option is supported: `FFTW21`. This invokes the `FFTW` ("Fastest Fourier Transform in the West") package of Frigo and Johnson. The calls are compatible with versions 2.1.x of FFTW. Instructions on compiling and linking to include `FFTW` are given below (§6.5).
6. FFT options `BRENNR` and `TMPRTN` have been eliminated as they seem to offer no advantages relative to `GPFAFT` or `FFTW21`.
7. `MPI` is now supported. Users can now carry out parallel calculations using **DDSCAT.6.0** on multiprocessor systems in conformance with the `MPI` (Message Passing Interface) standards (§24). This of course requires that `MPI` be already installed on the system.

5 Obtaining the Source Code

The easiest way to download the source code is from the DDSCAT home page:
`http://www.astro.princeton.edu/~draine/DDSCAT.html`
where you can obtain the file `ddscat6.0.tar.gz` – a "gzipped tarfile" containing the complete source code and documentation. This can also be obtained by anonymous `ftp` from `astro.princeton.edu`, directory `draine/scat/ddscat/ver6.0`. Note that it is a compressed (binary) file.

The source code can be installed as follows. First, place the file `ddscat6.0.tar.gz` in the directory where you would like **DDSCAT** to reside. You should have at least 5 Mbytes of disk space available.

If you are on a Linux system, you should be able to type

```
tar xvzf ddscat6.0.tar.gz
```

which will "gunzip" the tarfile and then "extract" the files from the tarfile. If your version of "tar" doesn't support the "z" option (e.g., you are running Solaris) then try

```
zcat ddscat6.0.tar.gz | tar xvf -
```

If neither of the above work on your system, try the two-stage procedure

```
gunzip ddscat6.0.tar.gz
tar xvf ddscat6.0.tar
```

(The disadvantage of the two-stage procedure is that it uses more disk space, since after the second step you will have the uncompressed tarfile `ddscat6.0.tar` – about 3.8 Mbytes – in addition to all the files you have extracted from the tarfile – another 4.6 Mbytes).

Any of the above approaches should create subdirectories `src, doc, misc,` and `IDL`. The source code will be in subdirectory `src`, and documentation in subdirectory `doc`.

If you are running Windows on a PC, you will need the "`winzip`" program, which can be downloaded from `http://www.winzip.com`. `winzip` should be able to "unzip" the gzipped tarfile `ddscat6.0.tar.gz` and "extract" the various files from it automatically.

6 Compiling and Linking

In the discussion below, it will be assumed that the source code for **DDSCAT.6.0** has been installed in a directory `DDA/src`. To compile the code on a Unix system, position yourself in the directory `DDA/src`. The `Makefile` supplied has compiler options set as appropriate for use of the `g77` compiler under the Linux operating system. If you have a different Fortran compiler, you will probably need to edit `Makefile` to provide the desired compiler options. If you are using an operating system other than Linux, you may need to change the Makefile choice of `timeit`. `Makefile` contains samples of compiler options for selected operating systems, including Linux, HP AUX, IBM AIX, and SGI IRIX operating systems.

So far as we know, there are only two operating-system-dependent aspects of **DDSCAT.6.0**: (1) the device number to use for "standard output", and (2) the `TIMEIT` routine. There are, in addition, three installation-dependent aspects to the code:

- the procedure for linking to the `FFTW` library (see §6.5)
- the procedure for linking to the `netCDF` library (see §10.3 for discussion of the possibility of writing binary files using the machine-independent `netCDF` standard).
- the procedure for linking to the `MPI` library (see §24.2).

6.1 Device Numbers `IDVOUT` and `IDVERR`

The variables `IDVOUT` and `IDVERR` specify device numbers for "running output" and "error messages", respectively. Normally these would both be set to the device number for "standard output" (e.g., writing to the screen if running interactively). The variables `IDVOUT` and `IDVERR` are set by `DATA` statements in the "main" program `DDSCAT.f` and in the output routine `WRIMSG` (file `wrimsg.f`). Under Sun Fortran, `DATA IDVOUT/0/` results in unbuffered output to "standard output"; unbuffered output (if available) is nice so that if you choose to direct your output to a file (e.g., using `ddscat >& ddscat.out &`) the output file will contain up-to-date information. Other operating systems or compilers may not support this, and you may need to edit `DDSCAT.f` to change the two data statements to `DATA IDVOUT/6/` and `DATA IDVERR/6/`, and edit `wrimsg.f` to change `DATA IDVOUT/0/` to `DATA IDVOUT/6/`.

6.2 Subroutine TIMEIT

The only other operating system-dependent part of **DDSCAT.6.0** is the single subroutine TIMEIT. Several versions of TIMEIT are provided:

- timeit_sun.f uses the SunOS system call etime
- timeit_convex.f uses the Convex OS system call etime
- timeit_cray.f uses the system call second
- timeit_hp.f uses the HP-AUX system calls sysconf and times
- timeit_ibm6000.f uses the AIX system call mclock
- timeit_osf.f uses the DEC OSF system call etime
- timeit_sgi.f uses the IRIX system call etime
- timeit_vms.f uses the VMS system calls LIB$INIT_TIMER and LIB$SHOW_TIMER
- timeit_titan.f uses the system call cputim
- timeit_null.f is a dummy routine which provides no timing information, but can be used under any operating system.

You *must* compile and link one of the timeit_*xxx*.f routines, possibly after modifying it to work on your system; timeit_null.f is the easiest choice, but it will return no timing information.[3]

6.3 Optimization

The performance of **DDSCAT.6.0** will benefit from optimization during compilation and the user should enable the appropriate compiler flags (e.g., g77 -c -O, or pgf77 -c -fast).

There is one exception: the LAPACK routine SLAMC1, which is called to determine the number of bits of precision provided by the computer architectures, should **NOT** be optimized, because under some optimizers the resulting code will end up in an endless loop. Therefore the Makefile has a separate compilation flag fFLAGSslamc1 which should *not* specify optimization (e.g., just g77 -c, or pgf77 -c). [4]

6.4 Leaving FFTW Capability Disabled

FFTW[5] – "Fastest Fourier Transform in the West" – is a publicly available FFT implementation that performs very well (see §13). **DDSCAT.6.0** supports use of FFTW, but it is recommended that first-time users of DDSCAT first get the code up and running without FFTW support, using the built-in GPFA FFT routine written by Clive Temperton, which performs almost as well as FFTW. This is also the option you will have to follow if FFTW is not installed on your system (though you can install it yourself, if necessary – see §6.5 below).

The Makefile supplied with the **DDSCAT.6.0** distribution is initially set up to leave support for FFTW disabled by using a "dummy" subroutine dummycxfftw.f.

When you type make ddscat you will create a version of ddscat which will not recognize option FFTW21 – you *must* specify option GFPAFT.

[3]Non-Unix sites: The VMS-compatible version of TIMEIT is included in the file SRC9.FOR. For non-VMS sites, you will need to replace this version of TIMEIT with one of the other versions, which are to be found in the file MISC.FOR. If you are having trouble getting this to work, choose the "dummy" version of TIMEIT from MISC.FOR: this will return no timing information, but is platform-independent.

[4]Note that the LAPACK routines are only used for some target geometries, and even when used the computation time is insignificant.

[5]http://www.fftw.org

6.5 Enabling FFTW Capability

The FFTW ("Fastest Fourier Transform in the West") package of Matteo Frigo and Steven G. Johnson appears to be one of the fastest public-domain FFT packages available (see §13 for a performance comparison), and **DDSCAT.6.0** allows this package to be used. In order to support the FFTW21 option, however, it is necessary to first install the FFTW package on your system (FFTW 2.1.x – we have not yet implemented support for FFTW 3.0, which has a different interface).

FFTW offers two advantages:

- It appears, for some cases, to be somewhat faster than the GPFA FFT algorithm which is already included in the DDSCAT package (see §13 and Fig. 5).
- It does not restrict the dimensions of the "computational volume" to be factorizable as $2^i 3^j 5^k$, and therefore for some targets will use a smaller computational volume (and therefore require less memory).

If the FFTW package is not yet installed on your system, then

1. Download the FFTW 2.1.x package from http://www.fftw.org (as of this writing the latest version 2.1.x is fftw-2.1.5.tar.gz) into some convenient directory.
2. `tar xvfz fftw-2.1.5.tar.gz`
3. `cd fftw-2.1.5`
4. `./configure --enable-float --enable-type-prefix --prefix=PATH`
   where PATH is the path to the directory in which you would like the fftw library installed (you must have write permission, of course). If you choose to omit
   `--prefix=PATH`
   then fftw will be installed in the default directory /usr/local/, but unless you are root you may not have write permission to this directory.
5. `make`
6. `make install`

This should install the file libsfftw.a in the directory /usr/local/lib (or PATH/lib if you used option --prefix=PATH when running configure).

Once FFTW is installed, you will need to edit the Makefile:

- define `cxfftw = cxfftw`
- define LIBFFTW to give the correct path to libsfftw.a

Once the above is done, typing `make ddscat` should create an executable ddscat which will recognize option FFTW21 in the file ddscat.par. Note that you will be linking fortran to C, and different compilers may behave idiosyncratically. The Makefile has examples which work for g77 and pgf77. If you have trouble, consult with someone familiar with linking fortran and C modules on your system.

6.6 Leaving the netCDF Capability Disabled

netCDF[6] is a standard for data transport used at many sites (see §10.3 for more information on netCDF). The Makefile supplied with the distribution of **DDSCAT.6.0** is set up to link to a "dummy" subroutine dummywritenet.f instead of subroutine writenet.f, in order to minimize problems during initial compilation and linking. The "dummy" routine has no functionality, other than bailing out with a fatal error message if the user makes the mistake of trying to specify one of the netCDF options (ALLCDF

[6] http://www.unidata.ucasr.edu/packages/netcdf/

or ORICDF). First-time users of **DDSCAT.6.0** should *not* try to use the netCDF option – simply skip this section, and specify option NOTCDF in ddscat.par. After successfully using **DDSCAT.6.0** you can return to §6.7 to try to enable the netCDF capability.

6.7 Enabling the netCDF Capability

Subroutine WRITENET (file writenet.f) provides the capability to output binary data in the netCDF standard format (see §10.3). In order to use this routine (instead of dummywritenet.f), it is necessary to take the following steps:[7]

1. Have netCDF already installed on your system (check with your system administrator).
2. Find out where netcdf.inc is located and edit the include statement in writenet.f to specify the correct path to netcdf.inc.
3. Find out where the libnetcdf.a library is located, and edit the Makefile so that the variable LIBNETCDF specifies the correct path to this library.
4. Edit the Makefile to "comment out" (with a # symbol in column 1) the line writenet = dummywritenet and "uncomment" (remove the # symbol) the line writenet = writenet so that writenet.f will be compiled instead of the dummy routine dummywritenet.f.

6.8 Compiling and Linking...

After suitably editing the Makefile (while still positioned in DDA/src) simply type[8]

```
make ddscat
```

which should create an executable file DDA/src/ddscat . If the default (as-provided) Makefile is used, the g77 fortran compiler will be used to create an executable which:

- will *not* have MPI support;
- will *not* have FFTW support;
- will *not* have netCDF capability;
- will contain timing instructions compatible with Linux, Solaris 1.x or 2.x, as well as several other versions of Unix.

To add FFTW capability, see §6.5. To add netCDF capability, see §6.7. To add MPI support, see §24. To replace the Linux-compatible and Sun-compatible timing routine with another, see §6.2.

6.9 Installation on Non-Unix Systems

DDSCAT.5a is written in standard Fortran-77 plus the DO ... ENDDO extension which appears to be supported by all current Fortran-77 compatible compilers. It is possible to run **DDSCAT** on non-Unix systems. If the Unix "make" utility is not available, here in brief is what needs to be accomplished:

All of the necessary Fortran code to compile and link **DDSCAT.5a** is included in the following files: SRC0.FOR, SRC1.FOR, SRC2.FOR, SRC3.FOR, SRC4.FOR, SRC5.FOR, SRC6.FOR, SRC7.FOR, SRC8.FOR, SRC9.FOR, CGCOMMON.FOR, GPFA.FOR, LAPACK.FOR, and PIM.FOR. There is an additional file MISC.FOR, but this is not needed for "basic" use of the code (see below).

[7]Non-UNIX sites: You need to replace the "dummy" version of SUBROUTINE WRITENET in SRC1.FOR with the version provided in MISC.FOR. You will also need to consult your systems administrator to verify that the netCDF library has already been installed on your system, and to find out how to link to the library routines.

[8]Non-UNIX sites: see §6.9

The main program DDSCAT is found in SRC0.FOR. It calls a number of subroutines, which are included in the other *.FOR files.

Four of the subroutines exist in more than one version. The "default" version of each is located in the file SRC1.FOR:

- Select the version of SUBROUTINE CXFFTW from SRC1.FOR instead of the version in MISC.FOR. The resulting code will not support **FFTW** capability. If you later wish to add FFTW capability, you will first need to install the FFTW library on your system (see §6.5). After you have done so, you can use the version of SUBROUTINE CXFFTW provided in MISC.FOR.
- Select the version of SUBROUTINE WRITENET from SRC1.FOR instead of the version in MISC.FOR. The resulting code will not support **netCDF** capability. (If **netCDF** capability is required, you will need to install **netCDF** libraries on your system – see §6.7).
- Select the version of SUBROUTINE C3DFFT from SRC1.FOR (do not use the version in MISC.FOR)
- There is one version of SUBROUTINE TIMEIT included in SRC1.FOR, and a number of additional versions in MISC.FOR. Use the version from SRC1.FOR, which begins

```
      SUBROUTINE TIMEIT(CMSGTM,DTIME)
C
C     timeit_null
C
C This version of timeit is a dummy which does not provide any
C timing information.
```

  This version does not use any system calls, and therefore the code should compile and link without problems; however, when you run the code, it will not report any timing information reporting how much time was spent on different parts of the calculation.

  If you wish to obtain timing information, you will need to find out what system calls are supported by your operating system. You can look at the other versions of SUBROUTINE TIMEIT in MISC.FOR to see how this has been done under VMS and various version of Unix.

Once you have selected the appropriate versions of CXFFTW, WRITENET, C3DFFT, and TIMEIT, you can simply compile and link just as you would with any other Fortran code with a number of modules. You should end up with an executable with a (system-dependent) name like DDSCAT.EXE.

In addition to program DDSCAT, we provide two other Fortran 77 programs which may be useful. Program CALLTARGET can be used to call the target generation routines which may be helpful for testing purposes. Program TSTFFT is useful for comparing speeds of different FFT options.

7 Moving the Executable

Now reposition yourself into the directory DDA (e.g., type cd ..), and copy the executable from src/ddscat to the DDA directory by typing

```
cp src/ddscat ddscat
```

This should copy the file DDA/src/ddscat to DDA/ddscat (you could, of course, simply create a symbolic link instead). Similarly, copy the sample parameter file ddscat.par and the file diel.tab to the DDA directory by typing

```
cp doc/ddscat.par ddscat.par
cp doc/diel.tab diel.tab
```

8 The Parameter File `ddscat.par`

The directory `DDA` should now contain a sample file `ddscat.par` which provides parameters to the program `ddscat`. As provided (see AppendixA), the file `ddscat.par` is set up to calculate scattering by a 8×6×4 rectangular array of 192 dipoles, with an effective radius $a_{\rm eff} = 1\mu{\rm m}$, at a wavelength of $6.2832\mu{\rm m}$ (for a "size parameter" $2\pi a_{\rm eff}/\lambda = 1$).

The dielectric function of the target material is provided in the file `diel.tab`, which is a sample file in which the refractive index is set to $m = 1.33 + 0.01i$ at all wavelengths; the name of this file is provided to `ddscat` by the parameter file `ddscat.par`.

The sample parameter file as supplied calls for the GPFA FFT routine (`GPFAFT`) of Temperton (1992) to be employed and the `PBCGST` iterative method to be used for solving the system of linear equations. (See section §13 and §14 for discussion of choice of FFT algorithm and choice of equation-solving algorithm.)

The sample parameter file specifies (via option `LATTDR`) that the "Lattice Dispersion Relation" (Draine & Goodman 1993) be employed to determine the dipole polarizabilities. See §11 for discussion of other options.

The sample parameter files specifies options `NOTBIN` and `NOTCDF` so that no binary data files and no NetCDF files will be created by `ddscat`.

The sample `ddscat.par` file specifies that the calculations be done for a single wavelength ($6.2832\mu{\rm m}$) and a single effective radius ($1.00\mu{\rm m}$). Note that in **DDSCAT.6.0** the "effective radius" $a_{\rm eff}$ is the radius of a sphere of equal volume – i.e., a sphere of volume Nd^3 , where d is the lattice spacing and N is the number of occupied (i.e., non-vacuum) lattice sites in the target. Thus the effective radius $a_{\rm eff} = (3N/4\pi)^{1/3}d$.

The incident radiation is always assumed to propagate along the x axis in the "Lab Frame". The sample `ddscat.par` file specifies incident polarization state $\hat{\mathbf{e}}_{01}$ to be along the y axis (and consequently polarization state $\hat{\mathbf{e}}_{02}$ will automatically be taken to be along the z axis). `IORTH=2` in `ddscat.par` calls for calculations to be carried out for both incident polarization states ($\hat{\mathbf{e}}_{01}$ and $\hat{\mathbf{e}}_{02}$ – see §21).

The target is assumed to have two vectors $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ embedded in it; $\hat{\mathbf{a}}_2$ is perpendicular to $\hat{\mathbf{a}}_1$. In the case of the 8×6×4 rectangular array of the sample calculation, the vector $\hat{\mathbf{a}}_1$ is along the "long" axis of the target, and the vector $\hat{\mathbf{a}}_2$ is along the "intermediate" axis. The target orientation in the Lab Frame is set by three angles: β, Θ, and Φ, defined and discussed below in §17. Briefly, the polar angles Θ and Φ specify the direction of $\hat{\mathbf{a}}_1$ in the Lab Frame. The target is assumed to be rotated around $\hat{\mathbf{a}}_1$ by an angle β. The sample `ddscat.par` file specifies $\beta = 0$ and $\Phi = 0$ (see lines in `ddscat.par` specifying variables `BETA` and `PHI`), and calls for three values of the angle Θ (see line in `ddscat.par` specifying variable `THETA`). **DDSCAT.6.0** chooses Θ values uniformly spaced in $\cos\Theta$; thus, asking for three values of Θ between 0 and 90° yields $\Theta = 0$, $60°$, and $90°$.

Appendix A provides a detailed description of the file `ddscat.par`.[9]

9 Running DDSCAT.6.0 Using the Sample `ddscat.par` File

To execute the program on a UNIX system (running either `sh` or `csh`), simply type

```
ddscat >& ddscat.out &
```

which will redirect the "standard output" to the file `ddscat.out`, and run the calculation in the background. The sample calculation (8x6x4=192 dipole target, 3 orientations, two incident polarizations, with scattering calculated for 14 distinct scattering directions), requires 1.0 cpu seconds on a Sun Ultra-170 workstation.

[9]Note that the structure of `ddscat.par` depends on the version of **DDSCAT** being used. Make sure you update old parameter files before using them with **DDSCAT.6.0** !

10 Output Files

10.1 ASCII files

If you run DDSCAT using the command

```
ddscat >& ddscat.out &
```

you will have various types of ASCII files when the computation is complete:

- a file `ddscat.out`;
- a file `mtable`;
- a file `qtable`;
- a file `qtable2`;
- files w*xx*r*yy*`ori.avg` (one, `w00r00ori.avg`, for the sample calculation);
- if `ddscat.par` specified `IWRKSC`=1, there will also be files w*xx*r*yy*k*zzz*`.sca` (3 for the sample calculation: `w00r00k000.sca`, `w00r00k001.sca`, `w00r00k002.sca`).

The file `ddscat.out` will contain any error messages generated as well as a running report on the progress of the calculation, including creation of the target dipole array. During the iterative calculations, $Q_{\rm ext}$, $Q_{\rm abs}$, and Q_{pha} are printed after each iteration; you will be able to judge the degree to which convergence has been achieved. Unless `TIMEIT` has been disabled, there will also be timing information.

The file `mtable` contains a summary of the dielectric constant used in the calculations.

The file `qtable` contains a summary of the orientationally-averaged values of $Q_{\rm ext}$, $Q_{\rm abs}$, $Q_{\rm sca}$, $g(1) = \langle\cos(\theta_s)\rangle$, and Q_{bk}. Here $Q_{\rm ext}$, $Q_{\rm abs}$, and $Q_{\rm sca}$ are the extinction, absorption, and scattering cross sections divided by $\pi a_{\rm eff}^2$. Q_{bk} is the differential cross section for backscattering (area per sr) divided by $\pi a_{\rm eff}^2$.

The file `qtable2` contains a summary of the orientationally-averaged values of Q_{pha}, Q_{pol}, and Q_{cpol}. Here Q_{pha} is the "phase shift" cross section divided by $\pi a_{\rm eff}^2$ (see definition in Draine 1988). Q_{pol} is the "polarization efficiency factor", equal to the difference between $Q_{\rm ext}$ for the two orthogonal polarization states. We define a "circular polarization efficiency factor" $Q_{cpol} \equiv Q_{pol}Q_{pha}$, since an optically-thin medium with a small twist in the alignment direction will produce circular polarization in initially unpolarized light in proportion to Q_{cpol}.

For each wavelength and size, **DDSCAT.6.0** produces a file with a name of the form w*xx*r*yy*`ori.avg`, where index *xx* (=00, 01, 02....) designates the wavelength and index *yy* (=00, 01, 02...) designates the "radius"; this file contains Q values and scattering information averaged over however many target orientations have been specified (see §17. The file `w00r00ori.avg` produced by the sample calculation is provided below in Appendix B.

In addition, if `ddscat.par` has specified `IWRKSC`=1 (as for the sample calculation), **DDSCAT.6.0** will generate files with names of the form w*xx*r*yy*k*zzz*`.avg`, where *xx* and *yy* are as before, and index *zzz* =(000,001,002...) designates the target orientation; these files contain Q values and scattering information for *each* of the target orientations. The structure of each of these files is very similar to that of the w*xx*r*yy*`ori.avg` files. Because these files may not be of particular interest, and take up disk space, you may choose to set `IWRKSC`=0 in future work. However, it is suggested that you run the sample calculation with `IWRKSC`=1.

The sample `ddscat.par` file specifies `IWRKSC`=1 and calls for use of 1 wavelength, 1 target size, and averaging over 3 target orientations. Running **DDSCAT.6.0** with the sample `ddscat.par` file will therefore generate files `w00r00k000.sca`, `w00r00k001.sca`, and `w00r00k002.sca` . To understand the information contained in one of these files, please consult Appendix C, which contains an example of the file `w00r00k000.sca` produced in the sample calculation.

10.2 Binary Option

It is possible to output an "unformatted" or "binary" file (`dd.bin`) with fairly complete information, including header and data sections. This is accomplished by specifying either `ALLBIN` or `ORIBIN` in `ddscat.par`.

Subroutine `writebin.f` provides an example of how this can be done. The "header" section contains dimensioning and other variables which do not change with wavelength, particle geometry, and target orientation. The header section contains data defining the particle shape, wavelengths, particle sizes, and target orientations. If `ALLBIN` has been specified, the "data" section contains, for each orientation, Mueller matrix results for each scattering direction. The data output is limited to actual dimensions of arrays; e.g. `nscat,4,4` elements of Muller matrix are written rather than `mxscat,4,4`. This is an important consideration when writing postprocessing codes.

A skeletal example of a postprocessing code was written by us (PJF) and is provided in subdirectory `DDA/IDL`. If you do plan to use the Interactive Data Language (IDL) for postprocessing, you may consider using the netCDF binary file option which offers substantial advantages over the FORTRAN unformatted write. More information about IDL is available at `http://www.rsinc.com/idl`. Unfortunately IDL requires a license and hence is not distributed with **DDSCAT**.

10.3 Machine-Independent Binary File Option: netCDF

The "unformatted" binary file is compact, fairly complete, and (in comparison to ASCII output files) is efficiently written from FORTRAN. However, binary files are not compatible between different computer architectures if written by "unformatted" `WRITE` from FORTRAN. Thus, you have to process the data on the same computer architecture that was used for the **DDSCAT** calculations. We provide an option of using netCDF with DDSCAT. The netCDF library[10] defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data. For more information, go to the `netcdf` website[10].

Several major graphics packages (for example IDL) have adopted netCDF as a standard for data transport. For this reason, and because of strong and free support of netCDF over the network by UNIDATA, we have implemented a netCDF interface in **DDSCAT**. This may become the method of choice for binary file storage of output from **DDSCAT**.

After the initial "learning curve" netCDF offers many advantages:

- It is easy to examine the contents of the file (using tools such as `ncdump`).
- Binary files are portable - they can be created on a supercomputer and processed on a workstation.
- Major graphics packages now offer netCDF interfaces.
- Data input and output are an order of magnitude faster than for ASCII files.
- Output data files are compact.

The disadvantages include:

- Need to have netCDF installed on your system.
- Lack of support of complex numbers.
- Nontrivial learning curve for using netCDF.

The public-domain netCDF software is available for many operating systems from `http://www.unidata.ucar.edu/packages/netcdf`. The steps necessary for enabling the netCDF capability in **DDSCAT.6.0** are listed above in §6.7. Once these have been successfully accomplished, and you are ready to run **DDSCAT** to produce netCDF output, you must choose either

[10] `http://www.unidata.ucar.edu/packages/netcdf/`

the ALLCDF or ORICDF option in ddscat.par; ALLCDF will result in a file which will include the Muller matrix for every wavelength, particle size, and orientation, whereas ORICDF will result in a file limited to just the orientational averages for each wavelength and target size.

11 Dipole Polarizabilities

Option LATTDR specifies that the "Lattice Dispersion Relation" (Draine & Goodman 1993) be employed to determine the dipole polarizabilities.; at this time this is the only supported option.

12 Dielectric Functions

In order to assign the appropriate dipole polarizabilities, **DDSCAT.6.0** must be given the dielectric constant of the material (or materials) of which the target of interest is composed. Unless the user wishes to use the dielectric function of either solid or liquid H_2O (see below), his information is supplied to **DDSCAT** through a table (or tables), read by subroutine DIELEC in file dielec.f, and providing either the complex refractive index $m = n + ik$ or complex dielectric function $\epsilon = \epsilon_1 + i\epsilon_2$ as a function of wavelength λ. Since $m = \epsilon^{1/2}$, or $\epsilon = m^2$, the user must supply either m or ϵ, but not both.

The table formatting is intended to be quite flexible. The first line of the table consists of text, up to 80 characters of which will be read and included in the output to identify the choice of dielectric function. (For the sample problem, it consists of simply the statement `m = 1.33 + 0.01i`.) The second line consists of 5 integers; either the second and third *or* the fourth and fifth should be zero.

- The first integer specifies which column the wavelength is stored in.
- The second integer specifies which column Re(m) is stored in.
- The third integer specifies which column Im(m) is stored in.
- The fourth integer specifies which column Re(ϵ) is stored in.
- The fifth integer specifies which column Im(ϵ) is stored in.

If the second and third integers are zeros, then DIELEC will read Re(ϵ) and Im(ϵ) from the file; if the fourth and fifth integers are zeros, then Re(m) and Im(m) will be read from the file.

The third line of the file is used for column headers, and the data begins in line 4. *There must be at least 3 lines of data:* even if ϵ is required at only one wavelength, please supply two additional "dummy" wavelength entries in the table so that the interpolation apparatus will not be confused.

In the event that the user wishes to compute scattering by targets with the refractive index of either solid or liquid H_2O, we have included two "built-in" dielectric functions. If H2OICE is specified on line 10 of ddscat.par, **DDSCAT** will use the dielectric function of H_2O ice at T=250K as compiled by Warren (1984). If H2OLIQ is specified on line 10 of ddscat.par, **DDSCAT** will use the dielectric function for liquid H_2O at T=280K using subroutine REFWAT written by Eric A. Smith. For more information see http://atol.ucsd.edu/∼pflatau/refrtab/index.htm.

13 Choice of FFT Algorithm

One major change in going from **DDSCAT.4b** to **4c** was modification of the code to permit use of the GPFA FFT algorithm developed by Dr. Clive Temperton (1992). In going from **DDSCAT.5a10** to 6.0 we introduce a new FFT option: the FFTW package of Frigo and Johnson. DDSCAT continues to offer Temperton's GPFA code as an alternative FFT implementations.

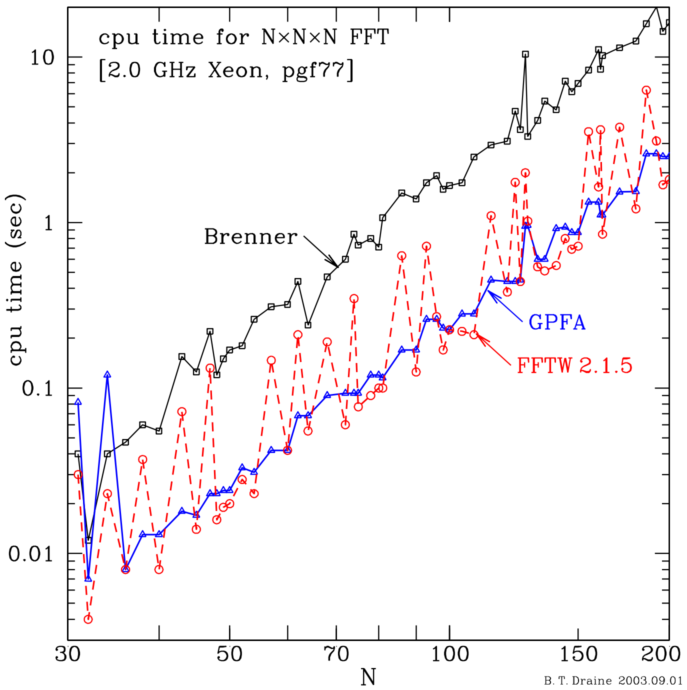

Figure 5: Comparison of cpu time required by 3 different FFT implementations. It is seen that the GPFA and FFTW implementations have comparable speeds, much faster than Brenner's FFT implementation.

To compare the performance of the GPFA algorithm and the FFTW code, we provide a program `TSTFFT` to compare the performance of different 3-D FFT implementations. The `TSTFFT` code requires the `FFTW` library (see §6.5). `TSTFFT` also provides timings for an FFT implementation due to Brenner (included in previous releases of **DDSCAT**).

To compile, link, and run this program on a Unix system,[11] position yourself in the `DDA/src` directory and type

```
make tstfft
tstfft
```

Output results will be written into a file `res.dat`. Table 1 is the `res.dat` file created when run on a 2.0 GHz Intel Xeon system, using the pgf77 compiler:

It is clear that both the GPFA code and the FFTW code are *much* faster than the Brenner FFT (by a factor of 7 for the 100×100×100 case). The FFTW code and GPFA code are quite comparable in performance – for some cases the GPFA code is faster, for other cases the FFTW code is faster. For target dimensions which are factorizable as $2^i3^j5^k$ (for integer i, j, k), the GPFA and FFTW codes have the same memory requirements. For targets with extents N_x, N_y, N_z which are not factorizable as $2^i3^j5^k$, the GPFA code needs to "extend" the computational volume to have values of N_x, N_y, and N_z which are factorizable by 2, 3, and 5. For these cases, GPFA requires somewhat more memory than FFTW. However, the fractional difference in required memory is not large, since integers factorizable as $2^i3^j5^k$ occur fairly frequently.[12]

[11]Non-Unix systems: the `TSTFFT` Fortran source code is in the file `MISC.FOR`.

[12]2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, 32, 36, 40, 45, 48, 50, 54, 60, 64, 72, 75, 80, 81, 90, 96, 100, 108, 120, 125, 128, 135, 144, 150, 160, 162, 180, 192, 200 are the integers ≤ 200 which are of the form $2^i3^j5^k$.

Table 1: FFT timing output from program `tstfft`

```
 CPU time (sec) for different 3-D FFT methods
 Machine=2.0 GHz Xeon, 256kB L2 cache, pgf77 compiler
 parameter LVR = 64
 LVR="length of vector registers" in gpfa2f,gpfa3f,gpfa5f
============================================================
                Brenner    Temperton Frigo & Johnson
  NX  NY  NZ    (FOURX)     (GPFA)      (FFTW)
  31  31  31      0.040       0.082       0.030
  32  32  32      0.012       0.007       0.004
  34  34  34      0.040       0.120       0.023
  36  36  36      0.047       0.008       0.008
  38  38  38      0.060       0.013       0.037
  40  40  40      0.055       0.013       0.008
  43  43  43      0.155       0.018       0.072
  45  45  45      0.125       0.017       0.014
  47  47  47      0.220       0.023       0.132
  48  48  48      0.120       0.023       0.016
  49  49  49      0.150       0.024       0.019
  50  50  50      0.170       0.024       0.020
  52  52  52      0.180       0.033       0.028
  54  54  54      0.260       0.031       0.023
  57  57  57      0.310       0.042       0.147
  60  60  60      0.320       0.042       0.042
  62  62  62      0.440       0.068       0.210
  64  64  64      0.240       0.068       0.055
  68  68  68      0.470       0.090       0.190
  72  72  72      0.600       0.093       0.060
  74  74  74      0.850       0.093       0.347
  75  75  75      0.730       0.093       0.077
  78  78  78      0.800       0.120       0.090
  80  80  80      0.710       0.120       0.100
  81  81  81      1.070       0.115       0.100
  86  86  86      1.510       0.170       0.630
  90  90  90      1.390       0.170       0.125
  93  93  93      1.740       0.260       0.720
  96  96  96      1.920       0.260       0.270
  98  98  98      1.590       0.230       0.170
 100 100 100      1.670       0.225       0.225
 104 104 104      1.740       0.280       0.220
 108 108 108      2.490       0.280       0.210
 114 114 114      2.940       0.450       1.100
 120 120 120      3.100       0.440       0.380
 123 123 123      4.710       0.440       1.750
 125 125 125      3.640       0.450       0.440
 127 127 127     10.400       0.950       2.000
 128 128 128      3.310       0.960       1.020
 132 132 132      4.140       0.600       0.540
 135 135 135      5.420       0.600       0.510
 140 140 140      4.800       0.920       0.550
 144 144 144      7.150       0.940       0.800
 147 147 147      6.170       0.870       0.690
 150 150 150      6.930       0.870       0.720
 155 155 155      8.350       1.330       3.540
 160 160 160     11.110       1.330       1.640
 161 161 161      8.450       1.110       3.640
 162 162 162     10.190       1.110       0.850
 171 171 171     11.380       1.530       3.770
 180 180 180     12.480       1.540       1.210
 186 186 186     15.920       2.600       6.310
 192 192 192     20.050       2.610       3.110
 196 196 196     14.270       2.500       1.690
 200 200 200     16.180       2.500       1.820
```

Based on Figure 5, we see that while for some cases `FFTW 2.1.5` is faster than the GPFA algorithm, the difference appears to be fairly marginal.

The GPFA code contains a parameter `LVR` which is set in `data` statements in the routines `gpfa2f`, `gpfa3f`, and `gpfa5f`. `LVR` is supposed to be optimized to correspond to the "length of a vector register" on vector machines. As delivered, this parameter is set to 64, which is supposed to be appropriate for Crays other than the C90 (for the C90, 128 is supposed to be preferable).[13] The value of `LVR` is not critical for scalar machines, as long as it is fairly large. We found little difference between `LVR`=64 and 128 on a Sparc 10/51, on an Ultrasparc 170, and on an Intel Xeon cpu. You may wish to experiment with different `LVR` values on your computer architecture. To change `LVR`, you need to edit `gpfa.f` and change the three `data` statements where `LVR` is set.

The choice of FFT implementation is obtained by specifying one of:

- `FFTW21` to use the FFTW 2.1.x algorithm of Frigo and Johnson – **this is recommended, but requires that the FFTW 2.1.x library be installed on your system**.
- `GPFAFT` to use the GPFA algorithm (Temperton 1992) – **this is almost as fast as the FFTW package, and does not require that the FFTW package be installed on your system**;
- `CONVEX` to use the native FFT routine on a Convex system (this option is presumably obsolete, but is retained as an example of how to use a system-specific native routine).

14 Choice of Iterative Algorithm

As discussed elsewhere (e.g., Draine 1988), the problem of electromagnetic scattering of an incident wave $\mathbf{E}_{\rm inc}$ by an array of N point dipoles can be cast in the form

$$\mathbf{AP} = \mathbf{E} \tag{17}$$

where $\mathbf{E}$ is a $3N$-dimensional (complex) vector of the incident electric field $\mathbf{E}_{\rm inc}$ at the N lattice sites, $\mathbf{P}$ is a $3N$-dimensional (complex) vector of the (unknown) dipole polarizations, and $\mathbf{A}$ is a $3N \times 3N$ complex matrix.

Because $3N$ is a large number, direct methods for solving this system of equations for the unknown vector $\mathbf{P}$ are impractical, but iterative methods are useful: we begin with a guess (typically, $\mathbf{P} = 0$) for the unknown polarization vector, and then iteratively improve the estimate for $\mathbf{P}$ until equation (17) is solved to some error criterion. The error tolerance may be specified as

$$\frac{|\mathbf{A}^\dagger \mathbf{AP} - \mathbf{A}^\dagger \mathbf{E}|}{|\mathbf{A}^\dagger \mathbf{E}|} < h \quad , \tag{18}$$

where $\mathbf{A}^\dagger$ is the Hermitian conjugate of $\mathbf{A}$ [$(A^\dagger)_{ij} \equiv (A_{ji})^*$], and h is the error tolerance. We typically use $h = 10^{-5}$ in order to satisfy eq.(17) to high accuracy. The error tolerance h can be specified by the user (see Appendix A).

A major change in going from **DDSCAT.4b** to **5a** (and subsequent versions) was the implementation of several different algorithms for iterative solution of the system of complex linear equations. **DDSCAT.5a** and **DDSCAT.6.0** have been structured to permit solution algorithms to be treated in a fairly "modular" fashion, facilitating the testing of different algorithms. Many algorithms were compared by Flatau (1997)[14]; two of them performed well and are made available to the user in **DDSCAT.6.0**. The choice of algorithm is made by specifying one of the options:

[13] In place of "preferable" users are encouraged to read "necessary" – there is some basis for fearing that results computed on a C90 with `LVR` other than 128 run the risk of being incorrect! Please use `LVR=128` if running on a C90!

[14] A postscript copy of this report – file `cg.ps` – is distributed with the **DDSCAT.6.0** documentation.

- `PBCGST` – Preconditioned BiConjugate Gradient with STabilitization method from the Parallel Iterative Methods (PIM) package created by R. Dias da Cunha and T. Hopkins.
- `PETRKP` – the complex conjugate gradient algorithm of Petravic & Kuo-Petravic (1979), as coded in the Complex Conjugate Gradient package (CCGPACK) created by P.J. Flatau. This is the algorithm discussed by Draine (1988) and used in previous versions of **DDSCAT**.

Both methods work well. Our experience suggests that `PBCGST` is often faster than `PETRKP`, by perhaps a factor of two. We therefore recommend it, but include the `PETRKP` method as an alternative.

The Parallel Iterative Methods (PIM) by Rudnei Dias da Cunha (`rdd@ukc.ac.uk`) and Tim Hopkins (`trh@ukc.ac.uk`) is a collection of Fortran 77 routines designed to solve systems of linear equations on parallel and scalar computers using a variety of iterative methods (available at `http://www.mat.ufrgs.br/pim-e.html`). PIM offers a number of iterative methods, including

- Conjugate-Gradients, CG (Hestenes 1952),
- Bi-Conjugate-Gradients, BICG (Fletcher 1976),
- Conjugate-Gradients squared, CGS (Sonneveld 1989),
- the stabilised version of Bi-Conjugate-Gradients, BICGSTAB (van der Vorst 1991),
- the restarted version of BICGSTAB, RBICGSTAB (Sleijpen & Fokkema 1992)
- the restarted generalized minimal residual, RGMRES (Saad 1986),
- the restarted generalized conjugate residual, RGCR (Eisenstat 1983),
- the normal equation solvers, CGNR (Hestenes 1952 and CGNE (Craig 1955),
- the quasi-minimal residual, QMR (highly parallel version due to Bucker & Sauren 1996),
- transpose-free quasi-minimal residual, TFQMR (Freund 1992),
- the Chebyshev acceleration, CHEBYSHEV (Young 1981).

The source code for these methods is distributed with `DDSCAT` but only `PBCGST` and `PETRKP` can be called directly via `ddscat.par`. It is possible (and was done) to add other options by changing the code in `getfml.f` . A helpful introduction to conjugate gradient methods is provided by the report "Conjugate Gradient Method Without Agonizing Pain" by Jonathan R. Shewchuk, available as a postscript file: `ftp://REPORTS.ADM.CS.CMU.EDU/usr0/anon/1994/CMU-CS-94-125.ps`.

15 Calculation of Radiative Force and Torque

In addition to solving the scattering problem for a dipole array, **DDSCAT.6.0** can compute the three-dimensional force $\mathbf{F}_{\rm rad}$ and torque $\mathbf{\Gamma}_{\rm rad}$ exerted on this array by the incident and scattered radiation fields. This calculation is carried out, after solving the scattering problem, provided `DOTORQ` has been specified in `ddscat.par`. For each incident polarization mode, the results are given in terms of dimensionless efficiency vectors $\mathbf{Q}_{pr}$ and $\mathbf{Q}_{\Gamma}$, defined by

$$\mathbf{Q}_{pr} \equiv \frac{\mathbf{F}_{\rm rad}}{\pi a_{\rm eff}^2 u_{\rm rad}} \quad , \tag{19}$$

$$\mathbf{Q}_{\Gamma} \equiv \frac{k\mathbf{\Gamma}_{\rm rad}}{\pi a_{\rm a_eff}^2 u_{\rm rad}} \quad , \tag{20}$$

where $\mathbf{F}_{\rm rad}$ and $\mathbf{\Gamma}_{\rm rad}$ are the time-averaged force and torque on the dipole array, $k = 2\pi/\lambda$ is the wavenumber *in vacuo*, and $u_{\rm rad} = E_0^2/8\pi$ is the time-averaged energy density for an incident plane wave with amplitude $E_0\cos(\omega t + \phi)$. The radiation pressure efficiency vector can be written

$$\mathbf{Q}_{\rm pr} = Q_{\rm ext}\hat{\mathbf{k}} - Q_{\rm sca}\mathbf{g} \quad , \tag{21}$$

where $\hat{\mathbf{k}}$ is the direction of propagation of the incident radiation, and the vector **g** is the mean direction of propagation of the scattered radiation:

$$\mathbf{g} = \frac{1}{C_{\rm sca}}\int d\Omega \frac{dC_{\rm sca}(\hat{\mathbf{n}},\hat{\mathbf{k}})}{d\Omega}\hat{\mathbf{n}} \quad , \tag{22}$$

where $d\Omega$ is the element of solid angle in scattering direction $\hat{\mathbf{n}}$, and $dC_{\rm sca}/d\Omega$ is the differential scattering cross section.

Equations for the evaluation of the radiative force and torque are derived by Draine & Weingartner (1996). It is important to note that evaluation of $\mathbf{Q}_{pr}$ and $\mathbf{Q}_\Gamma$ involves averaging over scattering directions to evaluate the linear and angular momentum transport by the scattered wave. This evaluation requires appropriate choices of the parameters `ICTHM` and `IPHM` – see §22.

16 Memory Requirements

Since Fortran-77 does not allow for dynamic memory allocation, the executable has compiled into it the dimensions for a number of arrays; these array dimensions limit the size of the dipole array which the code can handle. The source code supplied to you (which can be used to run the sample calculation with 192 dipoles) is restricted to problems with targets with a maximum extent of 16 lattice spacings along the x-, y-, and z-directions (`MXNX=16,MXNY=16,MXNZ=16`; i.e, the target must fit within an 16×16×16=4096 cube) and involve at most 9 different dielectric functions (`MXCOMP=9`). With this dimensioning, the executable requires about 1.3 MB of memory to run on an Ultrasparc system; memory requirements on other hardware and with other compilers should be similar. To run larger problems, you will need to edit the code to change `PARAMETER` values and recompile.

All of the dimensioning takes place in the main program `DDSCAT` – this should be the only routine which it is necessary to recompile. All of the dimensional parameters are set in `PARAMETER` statements appearing before the array declarations. You need simply edit the parameter statements. Remember, of course, that the amount of memory allocated by the code when it runs will depend upon these dimensioning parameters, so do not set them to ridiculously large values! The parameters `MXNX`, `MXNY`, `MXNZ` specify the maximum extent of the target in the x-, y-, or z-directions. The parameter `MXCOMP` specifies the maximum number of different dielectric functions which the code can handle at one time. The comment statements in the code supply all the information you should need to change these parameters.

The memory requirement for **DDSCAT.6.0** (with the netCDF and FFTW options disabled) is approximately

$$(1690 + 0.764\texttt{MXNX} \times \texttt{MXNY} \times \texttt{MXNZ})\ \text{kbytes} \tag{23}$$

Thus a 32×32×32 calculation requires 22.5 MBytes, while a 48×48×48 calculation requires 73.0 MBytes. These values are for the `pgf77` compiler on an Intel Xeon system running the Linux operating system.

17 Target Orientation

Recall that we define a "Lab Frame" (LF) in which the incident radiation propagates in the $+x$ direction. In `ddscat.par` one specifies the first polarization state $\hat{\mathbf{e}}_{01}$ (which obviously must lie in the y, z plane in the LF); **DDSCAT** automatically constructs a second polarization state $\hat{\mathbf{e}}_{02} = \hat{\mathbf{x}} \times \hat{\mathbf{e}}_{01}^*$ orthogonal to $\hat{\mathbf{e}}_{01}$ (here $\hat{\mathbf{x}}$ is the unit vector in the $+x$ direction of the LF. For purposes of discussion we will always let unit vectors $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}}$ be the three coordinate axes of the LF. Users will often find it convenient to let polarization vectors $\hat{\mathbf{e}}_{01} = \hat{\mathbf{y}}$, $\hat{\mathbf{e}}_{02} = \hat{\mathbf{z}}$ (although this is not mandatory – see §21).

Recall that definition of a target involves specifying two unit vectors, $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$, which are imagined to be "frozen" into the target. We require $\hat{\mathbf{a}}_2$ to be orthogonal to $\hat{\mathbf{a}}_1$. Therefore we may define a "Target Frame" (TF) defined by the three unit vectors $\hat{\mathbf{a}}_1$, $\hat{\mathbf{a}}_2$, and $\hat{\mathbf{a}}_3 = \hat{\mathbf{a}}_1 \times \hat{\mathbf{a}}_2$.

For example, when **DDSCAT** creates a $8\times6\times4$ rectangular solid, it fixes $\hat{\mathbf{a}}_1$ to be along the longest dimension of the solid, and $\hat{\mathbf{a}}_2$ to be along the next-longest dimension.

Orientation of the target relative to the incident radiation can in principle be determined two ways:

1. specifying the direction of $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ in the LF, or
2. specifying the directions of $\hat{\mathbf{x}}$ (incidence direction) and $\hat{\mathbf{y}}$ in the TF.

DDSCAT.6.0 uses method 1.: the angles Θ, Φ, and β are specified in the file `ddscat.par`. The target is oriented such that the polar angles Θ and Φ specify the direction of $\hat{\mathbf{a}}_1$ relative to the incident direction $\hat{\mathbf{x}}$, where the $\hat{\mathbf{x}}$,$\hat{\mathbf{y}}$ plane has $\Phi = 0$. Once the direction of $\hat{\mathbf{a}}_1$ is specified, the angle β than specifies how the target is to rotated around the axis $\hat{\mathbf{a}}_1$ to fully specify its orientation. A more extended and precise explanation follows:

17.1 Orientation of the Target in the Lab Frame

DDSCAT uses three angles, Θ, Φ, and β, to specify the directions of unit vectors $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ in the LF (see Fig. 6).

Θ is the angle between $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{x}}$.

When $\Phi = 0$, $\hat{\mathbf{a}}_1$ will lie in the $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ plane. When Φ is nonzero, it will refer to the rotation of $\hat{\mathbf{a}}_1$ around $\hat{\mathbf{x}}$: e.g., $\Phi = 90°$ puts $\hat{\mathbf{a}}_1$ in the $\hat{\mathbf{x}}, \hat{\mathbf{z}}$ plane.

When $\beta = 0$, $\hat{\mathbf{a}}_2$ will lie in the $\hat{\mathbf{x}}, \hat{\mathbf{a}}_1$ plane, in such a way that when $\Theta = 0$ and $\Phi = 0$, $\hat{\mathbf{a}}_2$ is in the $\hat{\mathbf{y}}$ direction: e.g, $\Theta = 90°$, $\Phi = 0$, $\beta = 0$ has $\hat{\mathbf{a}}_1 = \hat{\mathbf{y}}$ and $\hat{\mathbf{a}}_2 = -\hat{\mathbf{x}}$. Nonzero β introduces an additional rotation of $\hat{\mathbf{a}}_2$ around $\hat{\mathbf{a}}_1$: e.g., $\Theta = 90°$, $\Phi = 0$, $\beta = 90°$ has $\hat{\mathbf{a}}_1 = \hat{\mathbf{y}}$ and $\hat{\mathbf{a}}_2 = \hat{\mathbf{z}}$.

Mathematically:

$$
\begin{aligned}
\hat{\mathbf{a}}_1 &= \hat{\mathbf{x}}\cos\Theta + \hat{\mathbf{y}}\sin\Theta\cos\Phi + \hat{\mathbf{z}}\sin\Theta\sin\Phi && (24)\\
\hat{\mathbf{a}}_2 &= -\hat{\mathbf{x}}\sin\Theta\cos\beta + \hat{\mathbf{y}}[\cos\Theta\cos\beta\cos\Phi - \sin\beta\sin\Phi] \\
&\quad +\hat{\mathbf{z}}[\cos\Theta\cos\beta\sin\Phi + \sin\beta\cos\Phi] && (25)\\
\hat{\mathbf{a}}_3 &= \hat{\mathbf{x}}\sin\Theta\sin\beta - \hat{\mathbf{y}}[\cos\Theta\sin\beta\cos\Phi + \cos\beta\sin\Phi] \\
&\quad -\hat{\mathbf{z}}[\cos\Theta\sin\beta\sin\Phi - \cos\beta\cos\Phi] && (26)
\end{aligned}
$$

or, equivalently:

$$
\begin{aligned}
\hat{\mathbf{x}} &= \hat{\mathbf{a}}_1\cos\Theta - \hat{\mathbf{a}}_2\sin\Theta\cos\beta + \hat{\mathbf{a}}_3\sin\Theta\sin\beta && (27)\\
\hat{\mathbf{y}} &= \hat{\mathbf{a}}_1\sin\Theta\cos\Phi + \hat{\mathbf{a}}_2[\cos\Theta\cos\beta\cos\Phi - \sin\beta\sin\Phi] \\
&\quad -\hat{\mathbf{a}}_3[\cos\Theta\sin\beta\cos\Phi + \cos\beta\sin\Phi] && (28)\\
\hat{\mathbf{z}} &= \hat{\mathbf{a}}_1\sin\Theta\sin\Phi + \hat{\mathbf{a}}_2[\cos\Theta\cos\beta\sin\Phi + \sin\beta\cos\Phi] \\
&\quad -\hat{\mathbf{a}}_3[\cos\Theta\sin\beta\sin\Phi - \cos\beta\cos\Phi] && (29)
\end{aligned}
$$

17.2 Orientation of the Incident Beam in the Target Frame

Under some circumstances, one may wish to specify the target orientation such that $\hat{\mathbf{x}}$ (the direction of propagation of the radiation) and $\hat{\mathbf{y}}$ (usually the first polarization direction) and $\hat{\mathbf{z}}$ (= $\hat{\mathbf{x}} \times \hat{\mathbf{y}}$) refer to

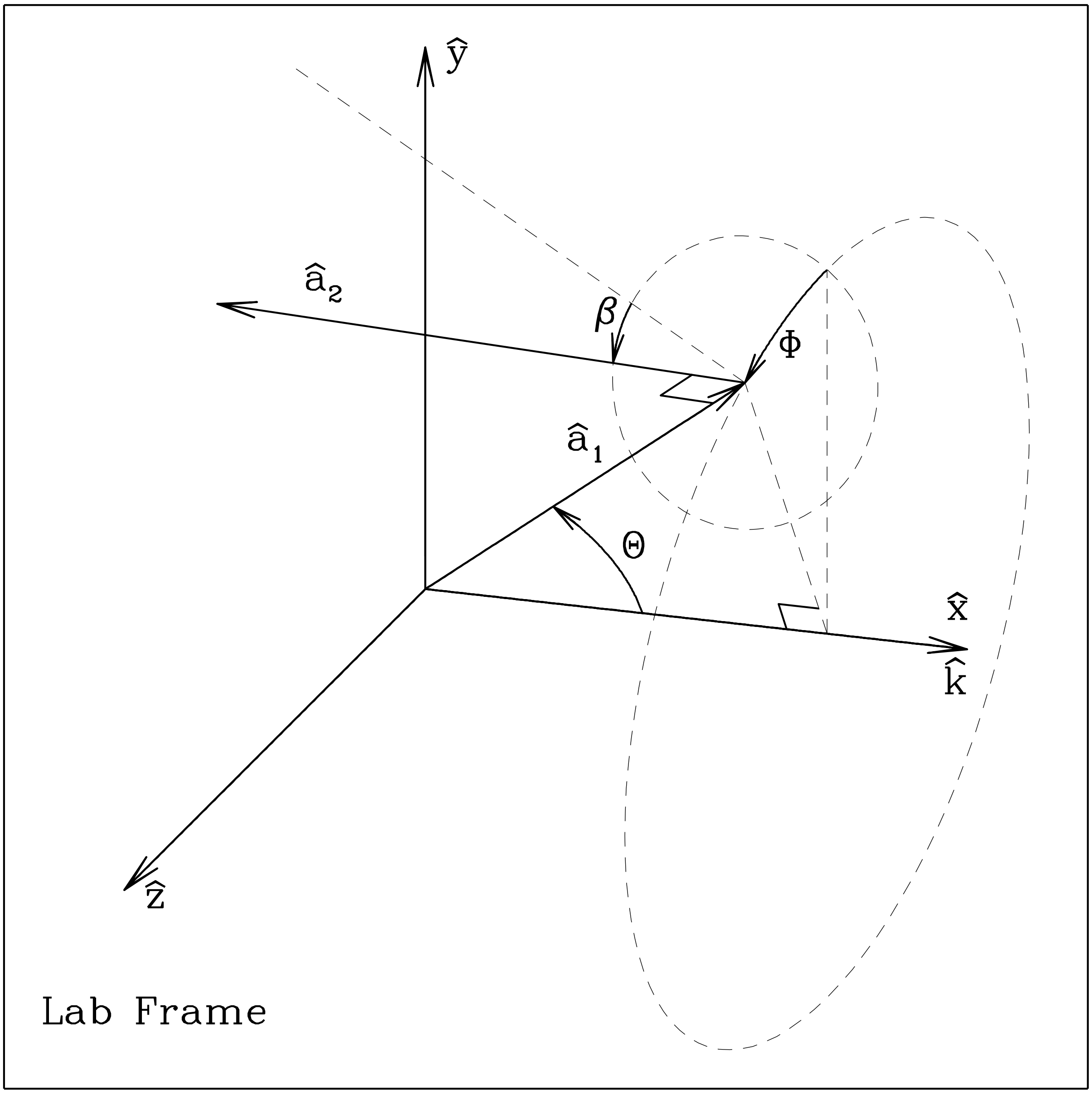

Figure 6: Target orientation in the Lab Frame. $\hat{\mathbf{x}}$ is the direction of propagation of the incident radiation, and $\hat{\mathbf{y}}$ is the direction of the "real" component of the first incident polarization mode. In this coordinate system, the orientation of target axis $\hat{\mathbf{a}}_1$ 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 $\hat{\mathbf{a}}_1$,$\hat{\mathbf{x}}$ plane.

certain directions in the TF. Given the definitions of the LF and TF above, this is simply an exercise in coordinate transformation. For example, one might wish to have the incident radiation propagating along the (1,1,1) direction in the TF (example 14 below). Here we provide some selected examples:

1. $\hat{\mathbf{x}} = \hat{\mathbf{a}}_1, \hat{\mathbf{y}} = \hat{\mathbf{a}}_2, \hat{\mathbf{z}} = \hat{\mathbf{a}}_3 : \Theta = 0, \Phi + \beta = 0$
2. $\hat{\mathbf{x}} = \hat{\mathbf{a}}_1, \hat{\mathbf{y}} = \hat{\mathbf{a}}_3, \hat{\mathbf{z}} = -\hat{\mathbf{a}}_2 : \Theta = 0, \Phi + \beta = 90°$
3. $\hat{\mathbf{x}} = \hat{\mathbf{a}}_2, \hat{\mathbf{y}} = \hat{\mathbf{a}}_1, \hat{\mathbf{z}} = -\hat{\mathbf{a}}_3 : \Theta = 90°, \beta = 180°, \Phi = 0$
4. $\hat{\mathbf{x}} = \hat{\mathbf{a}}_2, \hat{\mathbf{y}} = \hat{\mathbf{a}}_3, \hat{\mathbf{z}} = \hat{\mathbf{a}}_1 : \Theta = 90°, \beta = 180°, \Phi = 90°$
5. $\hat{\mathbf{x}} = \hat{\mathbf{a}}_3, \hat{\mathbf{y}} = \hat{\mathbf{a}}_1, \hat{\mathbf{z}} = \hat{\mathbf{a}}_2 : \Theta = 90°, \beta = -90°, \Phi = 0$
6. $\hat{\mathbf{x}} = \hat{\mathbf{a}}_3, \hat{\mathbf{y}} = \hat{\mathbf{a}}_2, \hat{\mathbf{z}} = -\hat{\mathbf{a}}_1 : \Theta = 90°, \beta = -90°, \Phi = -90°$
7. $\hat{\mathbf{x}} = -\hat{\mathbf{a}}_1, \hat{\mathbf{y}} = \hat{\mathbf{a}}_2, \hat{\mathbf{z}} = -\hat{\mathbf{a}}_3 : \Theta = 180°, \beta + \Phi = 180°$
8. $\hat{\mathbf{x}} = -\hat{\mathbf{a}}_1, \hat{\mathbf{y}} = \hat{\mathbf{a}}_3, \hat{\mathbf{z}} = \hat{\mathbf{a}}_2 : \Theta = 180°, \beta + \Phi = 90°$
9. $\hat{\mathbf{x}} = -\hat{\mathbf{a}}_2, \hat{\mathbf{y}} = \hat{\mathbf{a}}_1, \hat{\mathbf{z}} = \hat{\mathbf{a}}_3 : \Theta = 90°, \beta = 0, \Phi = 0$
10. $\hat{\mathbf{x}} = -\hat{\mathbf{a}}_2, \hat{\mathbf{y}} = \hat{\mathbf{a}}_3, \hat{\mathbf{z}} = -\hat{\mathbf{a}}_1 : \Theta = 90°, \beta = 0, \Phi = -90°$
11. $\hat{\mathbf{x}} = -\hat{\mathbf{a}}_3, \hat{\mathbf{y}} = \hat{\mathbf{a}}_1, \hat{\mathbf{z}} = -\hat{\mathbf{a}}_2 : \Theta = 90°, \beta = -90°, \Phi = 0$
12. $\hat{\mathbf{x}} = -\hat{\mathbf{a}}_3, \hat{\mathbf{y}} = \hat{\mathbf{a}}_2, \hat{\mathbf{z}} = \hat{\mathbf{a}}_1 : \Theta = 90°, \beta = -90°, \Phi = 90°$
13. $\hat{\mathbf{x}} = (\hat{\mathbf{a}}_1 + \hat{\mathbf{a}}_2)/\sqrt{2}, \hat{\mathbf{y}} = \hat{\mathbf{a}}_3, \hat{\mathbf{z}} = (\hat{\mathbf{a}}_1 - \hat{\mathbf{a}}_2)/\sqrt{2}$: $\Theta = 45°, \beta = 180°, \Phi = 90°$
14. $\hat{\mathbf{x}} = (\hat{\mathbf{a}}_1 + \hat{\mathbf{a}}_2 + \hat{\mathbf{a}}_3)/\sqrt{3}, \hat{\mathbf{y}} = (\hat{\mathbf{a}}_1 - \hat{\mathbf{a}}_2)/\sqrt{2}, \hat{\mathbf{z}} = (\hat{\mathbf{a}}_1 + \hat{\mathbf{a}}_2 - 2\hat{\mathbf{a}}_3)/\sqrt{6}$: $\Theta = 54.7356°, \beta = 135°$, $\Phi = 30°$.

17.3 Sampling in Θ, Φ, and β

The present version, **DDSCAT.6.0**, chooses the angles β, Θ, and Φ to sample the intervals (`BETAMI,BETAMX`), (`THETMI,THETMX`), (`PHIMIN,PHIMAX`), where `BETAMI`, `BETAMX`, `THETMI`, `THETMX`, `PHIMIN`, `PHIMAX` are specified in `ddscat.par` . The prescription for choosing the angles is to:

- uniformly sample in β;
- uniformly sample in Φ;
- uniformly sample in $\cos\Theta$.

This prescription is appropriate for random orientation of the target, within the specified limits of β, Φ, and Θ.

Note that when **DDSCAT.6.0** chooses angles it handles β and Φ differently from Θ.[15] The range for β is divided into `NBETA` intervals, and the midpoint of each interval is taken. Thus, if you take `BETAMI`=0, `BETAMX`=90, `NBETA`=2 you will get $\beta = 22.5°$ and $67.5°$. Similarly, if you take `PHIMIN`=0, `PHIMAX`=180, `NPHI`=2 you will get $\Phi = 45°$ and $135°$.

Sampling in Θ is done quite differently from sampling in β and Φ. First, as already mentioned above, **DDSCAT.6.0** samples uniformly in $\cos\Theta$, not Θ. Secondly, the sampling depends on whether `NTHETA` is even or odd.

- If `NTHETA` is odd, then the values of Θ selected include the extreme values `THETMI` and `THETMX`; thus, `THETMI`=0, `THETMX`=90, `NTHETA`=3 will give you $\Theta = 0, 60°, 90°$.
- If `NTHETA` is even, then the range of $\cos\Theta$ will be divided into `NTHETA` intervals, and the midpoint of each interval will be taken; thus, `THETMI`=0, `THETMX`=90, `NTHETA`=2 will give you $\Theta = 41.41°$ and $75.52°$ [$\cos\Theta = 0.25$ and 0.75].

[15]This is a change from **DDSCAT.4a**!!.

The reason for this is that if odd NTHETA is specified, than the "integration" over $\cos\Theta$ is performed using Simpson's rule for greater accuracy. If even NTHETA is specified, then the integration over $\cos\Theta$ is performed by simply taking the average of the results for the different Θ values.

If averaging over orientations is desired, it is recommended that the user specify an *odd* value of NTHETA so that Simpson's rule will be employed.

18 Orientational Averaging

DDSCAT has been constructed to facilitate the computation of orientational averages. How to go about this depends on the distribution of orientations which is applicable.

18.1 Randomly-Oriented Targets

For randomly-oriented targets, we wish to compute the orientational average of a quantity $Q(\beta,\Theta,\Phi)$:

$$\langle Q\rangle = \frac{1}{8\pi^2}\int_0^{2\pi} d\beta \int_{-1}^{1} d\cos\Theta \int_0^{2\pi} d\Phi\; Q(\beta,\Theta,\Phi) \quad . \tag{30}$$

To compute such averages, all you need to do is edit the file `ddscat.par` so that DDSCAT knows what ranges of the angles β, Θ, and Φ are of interest. For a randomly-oriented target with no symmetry, you would need to let β run from 0 to 360°, Θ from 0 to 180°, and Φ from 0 to 360°.

For targets with symmetry, on the other hand, the ranges of β, Θ, and Φ may be reduced. First of all, remember that averaging over Φ is relatively "inexpensive", so when in doubt average over 0 to 360°; most of the computational "cost" is associated with the number of different values of (β,Θ) which are used. Consider a cube, for example, with axis $\hat{\mathbf{a}}_1$ normal to one of the cube faces; for this cube β need run only from 0 to 90°, since the cube has fourfold symmetry for rotations around the axis $\hat{\mathbf{a}}_1$. Furthermore, the angle Θ need run only from 0 to 90°, since the orientation (β,Θ,Φ) is indistinguishable from (β, 180° $-\Theta$, 360° $-\Phi$).

For targets with symmetry, the user is encouraged to test the significance of β,Θ,Φ on targets with small numbers of dipoles (say, of the order of 100 or so) but having the desired symmetry.

It is important to remember that **DDSCAT.4b** handled even and odd values of NTHETA differently – see §8 above! For averaging over random orientations odd values of NTHETA are to be preferred, as this will allow use of Simpson's rule in evaluating the "integral" over $\cos\Theta$.

18.2 Nonrandomly-Oriented Targets

Some special cases (where the target orientation distribution is uniform for rotations around the x axis = direction of propagation of the incident radiation), one may be able to use **DDSCAT.6.0** with appropriate choices of input parameters. More generally, however, you will need to modify subroutine ORIENT to generate a list of NBETA values of β, NTHETA values of Θ, and NPHI values of Φ, plus two weighting arrays `WGTA(1-NTHETA,1-NPHI)` and `WGTB(1-NBETA)`. Here WGTA gives the weights which should be attached to each (Θ,Φ) orientation, and WGTB gives the weight to be attached to each β orientation. Thus each orientation of the target is to be weighted by the factor WGTA×WGTB. For the case of random orientations, **DDSCAT.6.0** chooses Θ values which are uniformly spaced in $\cos\Theta$, and β and Φ values which are uniformly spaced, and therefore uses uniform weights

```
WGTB=1./NBETA
```

When NTHETA is even, **DDSCAT** sets

```
WGTA=1./(NTHETA×NPHI)
```

but when NTHETA is odd, **DDSCAT** uses Simpson's rule when integrating over Θ and

WGTA= (1/3 or 4/3 or 2/3)/(NTHETA$\times$NPHI)

Note that the program structure of **DDSCAT** may not be ideally suited for certain highly oriented cases. If, for example, the orientation is such that for a given Φ value only one Θ value is possible (this situation might describe ice needles oriented with the long axis perpendicular to the vertical in the Earth's atmosphere, illuminated by the Sun at other than the zenith) then it is foolish to consider all the combinations of Θ and Φ which the present version of **DDSCAT** is set up to do. We hope to improve this in a future version of **DDSCAT**.

19 Target Generation

DDSCAT contains routines to generate dipole arrays representing targets of various geometries, including spheres, ellipsoids, rectangular solids, cylinders, hexagonal prisms, tetrahedra, two touching ellipsoids, and three touching ellipsoids. The target type is specified by variable CSHAPE on line 9 of ddscat.par, up to 6 target shape parameters (SHPAR1, SHPAR2, SHPAR3, ...) on line 10, and the lattice spacing DX DY DZ on line 11 (for a cubic lattice, set DX DY DZ to 1. 1. 1. on line 11). The target geometry is most conveniently described in a coordinate system attached to the target which we refer to as the "Target Frame" (TF), so in this section *only* we will let x,y,z be coordinates in the Target Frame. Once the target is generated, the orientation of the target in the Lab Frame is accomplished as described in §17.

Target geometries currently supported include:[16]

- ELLIPS – **Homogeneous, isotropic ellipsoid.**
  "Lengths" SHPAR1, SHPAR2, SHPAR3 in the x, y, z directions in the TF. $(x/\mathtt{SHPAR1})^2 + (y/\mathtt{SHPAR2})^2 + (z/\mathtt{SHPAR3})^2 = d^2/4$, where d is the interdipole spacing.
  The target axes are set to $\hat{\mathbf{a}}_1 = (1, 0, 0)$ and $\hat{\mathbf{a}}_2 = (0, 1, 0)$ in the TF.
  User must set NCOMP=1 on line 9 of ddscat.par.
  A **homogeneous, isotropic sphere** is obtained by setting SHPAR1 = SHPAR2 = SHPAR3 = diameter/d.
- ANIELL – **Homogeneous, anisotropic ellipsoid.**
  SHPAR1, SHPAR2, SHPAR3 have same meaning as for ELLIPS. Target axes $\hat{\mathbf{a}}_1 = (1, 0, 0)$ and $\hat{\mathbf{a}}_2 = (0, 1, 0)$ in the TF. It is assumed that the dielectric tensor is diagonal in the TF. User must set NCOMP=3 and provide xx, yy, zz elements of the dielectric tensor.
- CONELL – **Two concentric ellipsoids.**
  SHPAR1, SHPAR2, SHPAR3 specify the lengths along x, y, z axes (in the TF) of the *outer* ellipsoid; SHPAR4, SHPAR5, SHPAR6 are the lengths, along the x, y, z axes (in the TF) of the *inner* ellipsoid. The "core" within the inner ellipsoid is composed of isotropic material 1; the "mantle" between inner and outer ellipsoids is composed of isotropic material 2. Target axes $\hat{\mathbf{a}}_1 = (1, 0, 0)$, $\hat{\mathbf{a}}_2 = (0, 1, 0)$ in TF. User must set NCOMP=2 and provide dielectric functions for "core" and "mantle" materials.
- CYLNDR – **Homogeneous, isotropic cylinder.**
  Length/d=SHPAR1, diameter/d=SHPAR2, with cylinder axis $= \hat{\mathbf{a}}_1 = (1, 0, 0)$ and $\hat{\mathbf{a}}_2 = (0, 1, 0)$ in the TF. User must set NCOMP=1.

[16] In DDSCAT.6.0 not all of the target generation routines have been modified for use with a noncubic lattice. Those routines which have not yet been modified (TAR2EL, TAR2SP, TAR3EL, TARCEL, TARHEX, TARTET include code to ensure that they are not used with a noncubic lattice; if called with a noncubic lattice, an error message is generated (e.g. "tarhex does not support noncubic lattice") and execution is terminated.

- UNICYL – **Homogeneous cylinder with unixial anisotropic dielectric tensor.**
SHPAR1, SHPAR2 have same meaning as for CYLNDR. Cylinder axis = $\hat{\mathbf{a}}_1 = (1,0,0)$, $\hat{\mathbf{a}}_2 = (0,1,0)$. It is assumed that the dielectric tensor ϵ is diagonal in the TF, with $\epsilon_{yy} = \epsilon_{zz}$. User must set NCOMP=2. Dielectric function 1 is for $\mathbf{E} \parallel \hat{\mathbf{a}}_1$ (cylinder axis), dielectric function 2 is for $\mathbf{E} \perp \hat{\mathbf{a}}_1$.

- RCTNGL – **Homogeneous, isotropic, rectangular solid.**
x, y, z lengths/d = SHPAR1, SHPAR2, SHPAR3. Target axes $\hat{\mathbf{a}}_1 = (1,0,0)$ and $\hat{\mathbf{a}}_2 = (0,1,0)$ in the TF. User must set NCOMP=1.

- HEXGON – **Homogeneous, isotropic hexagonal prism.**
Length/d = SHPAR1, hexagon side/d = SHPAR2. Target axis $\hat{\mathbf{a}}_1 = (1,0,0)$ in the TF is along the prism axis, and target axis $\hat{\mathbf{a}}_2 = (0,1,0)$ in the TF is normal to one of the rectangular faces of the hexagonal prism. User must set NCOMP=1.

- TETRAH – **Homogeneous, isotropic tetrahedron.**
SHPAR1=length/d of one edge. Orientation: one face parallel to y,z plane (in the TF), opposite "vertex" is in $+x$ direction, and one edge is parallel to z axis (in the TF). Target axes $\hat{\mathbf{a}}_1 = (1,0,0)$ [emerging from one vertex] and $\hat{\mathbf{a}}_2 = (0,1,0)$ [emerging from an edge] in the TF. User must set NCOMP=1.

- TWOSPH – **Two touching homogeneous, isotropic spheroids, with distinct compositions.**
First spheroid has length SHPAR1 along symmetry axis, diameter SHPAR2 perpendicular to symmetry axis. Second spheroid has length SHPAR3 along symmetry axis, diameter SHPAR4 perpendicular to symmetry axis. Contact point is on line connecting centroids. Line connecting centroids is in x direction. Symmetry axis of first spheroid is in y direction. Symmetry axis of second spheroid is in direction $\hat{\mathbf{y}}\cos(\mathtt{SHPAR5}) + \hat{\mathbf{z}}\sin(\mathtt{SHPAR5})$, where $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are basis vectors in TF, and SHPAR5 is in degrees. If SHPAR6=0., then target axes $\hat{\mathbf{a}}_1 = (1,0,0)$, $\hat{\mathbf{a}}_2 = (0,1,0)$. If SHPAR6=1., then axes $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ are set to principal axes with largest and 2nd largest moments of inertia assuming spheroids to be of uniform density. User must set NCOMP=2 and provide dielectric function files for each spheroid.

- TWOELL – **Two touching, homogeneous, isotropic ellipsoids, with distinct compositions.**
SHPAR1, SHPAR2, SHPAR3=x-length/d, y-length/d, z-length/d of one ellipsoid. The two ellipsoids have identical shape, size, and orientation, but distinct dielectric functions. The line connecting ellipsoid centers is along the x-axis in the TF. Target axes $\hat{\mathbf{a}}_1 = (1,0,0)$ [along line connecting ellipsoids] and $\hat{\mathbf{a}}_2 = (0,1,0)$. User must set NCOMP=2 and provide dielectric function file names for both ellipsoids. Ellipsoids are in order of increasing x: first dielectric function is for ellipsoid with center at negative x, second dielectric function for ellipsoid with center at positive x.

- TWOAEL – **Two touching, homogeneous, anisotropic ellipsoids, with distinct compositions.**
Geometry as for TWOELL; SHPAR1, SHPAR2, SHPAR3 have same meanings as for TWOELL. Target axes $\hat{\mathbf{a}}_1 = (1,0,0)$ and $\hat{\mathbf{a}}_2 = (0,1,0)$ in the TF. It is assumed that (for both ellipsoids) the dielectric tensor is diagonal in the TF. User must set NCOMP=6 and provide xx, yy, zz components of dielectric tensor for first ellipsoid, and xx, yy, zz components of dielectric tensor for second ellipsoid (ellipsoids are in order of increasing x).

- THRELL – **Three touching homogeneous, isotropic ellipsoids of equal size and orientation, but distinct compositions.**
SHPAR1, SHPAR2, SHPAR3 have same meaning as for TWOELL. Line connecting ellipsoid centers is parallel to x axis. Target axis $\hat{\mathbf{a}}_1 = (1,0,0)$ along line of ellipsoid centers, $\hat{\mathbf{a}}_2 = (0,1,0)$. User must set NCOMP=3 and provide (isotropic) dielectric functions for first, second, and third ellipsoid.

- THRAEL – **Three touching homogeneous, anisotropic ellipsoids with same size and orientation but distinct dielectric tensors.**
SHPAR1, SHPAR2, SHPAR3 have same meanings as for THRELL. Target axis $\hat{\mathbf{a}}_1 = (1,0,0)$ along line of ellipsoid centers, $\hat{\mathbf{a}}_2 = (0,1,0)$. It is assumed that dielectric tensors are all diagonal in the TF. User must set NCOMP=9 and provide xx, yy, zz elements of dielectric tensor for first ellipsoid, xx, yy, zz elements for second ellipsoid, and xx, yy, zz elements for third ellipsoid (ellipsoids are in order of increasing x).
- BLOCKS – **Homogeneous target constructed from cubic "blocks".**
Number and location of blocks are specified in separate file blocks.par with following structure:
  one line of comments (may be blank)
  PRIN (= 0 or 1 – see below)
  N (= number of blocks)
  B (= width/d of one block)
  x y z (= position of 1st block in units of Bd)
  x y z (= position of 2nd block)
  ...
  x y z (= position of Nth block)
If PRIN=0, then $\hat{\mathbf{a}}_1 = (1,0,0)$, $\hat{\mathbf{a}}_2 = (0,1,0)$. If PRIN=1, then $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ are set to principal axes with largest and second largest moments of inertia, assuming target to be of uniform density. User must set NCOMP=1.
- DW1996 – **13 block target used by Draine & Weingartner (1996).**
Single, isotropic material. Target geometry was used in study by Draine & Weingartner (1996) of radiative torques on irregular grains. $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ are principal axes with largest and second-largest moments of inertia. User must set NCOMP=1. Target size is controlled by shape parameter SHPAR(1) = width of one block in lattice units.
- NSPHER – **Multisphere target consisting of the union of N spheres of single isotropic material.**
Spheres may overlap if desired. The relative locations and sizes of these spheres are defined in an external file, whose name (enclosed in quotes) is passed through ddscat.par. The length of the file name should not exceed 13 characters. The pertinent line in ddscat.par should read
SHPAR(1) SHPAR(2) *'filename'* (quotes must be used)
where SHPAR(1) = target diameter in x direction (in Target Frame) in units of d
SHPAR(2) = 0 to have $a_1 = (1,0,0)$, $a_2 = (0,1,0)$ in Target Frame.
SHPAR(2) = 1 to use principal axes of moment of inertia tensor for a_1 (largest I) and a_2 (intermediate I).
filename is the name of the file specifying the locations and relative sizes of the spheres.
The overall size of the multisphere target (in terms of numbers of dipoles) is determined by parameter SHPAR(1), which is the extent of the multisphere target in the x-direction, in units of the lattice spacing d. The file *'filename'* should have the following structure:
  N (= number of spheres)
  one line of comments (may be blank)
  x_1 y_1 z_1 a_1 (arb. units)
  x_2 y_2 z_2 a_2 (arb. units)
  ...
  x_N y_N z_N a_N (arb. units)
where x_j, y_j, z_j are the coordinates of the center, and a_j is the radius of sphere j.
Note that x_j, y_j, z_j, a_j ($j = 1, ..., N$) establish only the *shape* of the $N-$sphere target. For instance, a target consisting of two touching spheres with the line between centers parallel to the x

axis could equally well be described by lines 3 and 4 being

0 0 0 0.5
1 0 0 0.5

or

0 0 0 1
2 0 0 1

The actual size (in physical units) is set by the value of $a_{\rm eff}$ specified in `ddscat.par`, where, as always, $a_{\rm eff} \equiv (3V/4\pi)^{1/3}$, where V is the volume of material in the target.
User must set `NCOMP`=1.

- `PRISM3` – **Triangular prism of homogeneous, isotropic material.**
`SHPAR1, SHPAR2, SHPAR3, SHPAR4` $= a/d, b/a, c/a, L/a$
The triangular cross section has sides of width a, b, c. L is the length of the prism. d is the lattice spacing. The triangular cross-section has interior angles α, β, γ (opposite sides a, b, c) given by $\cos\alpha = (b^2+c^2-a^2)/2bc$, $\cos\beta = (a^2+c^2-b^2)/2ac$, $\cos\gamma = (a^2+b^2-c^2)/2ab$. In the Target Frame, the prism axis is in the $\hat{\mathbf{x}}$ direction, the normal to the rectangular face of width a is (0,1,0), the normal to the rectangular face of width b is $(0,-\cos\gamma,\sin\gamma)$, and the normal to the rectangular face of width c is $(0,-\cos\gamma,-\sin\gamma)$.
- `LYRSLB` – Multilayer rectangular slab with overall x, y, z lengths a_x = `SHPAR1`$\times d$, a_y = `SHPAR2`$\times d$, a_z = `SHPAR3`$\times d$, with the boundary between layers 1 and 2 occuring at $x_1 = \texttt{SHPAR4} \times a_x$, boundary between layers 2 and 3 occuring at $x_2 = \texttt{SHPAR5} \times a_x$ [if `SHPAR5`>0], and boundary between layers 3 and 4 occuring at $x_3 = \texttt{SHPAR6} \times a_x$ [if `SHPAR6`>0], To create a simple bi-layer slab, just set `SHPAR5`=`SHPAR6`=0. User must set `NCOMP`=2,3, or 4 and provide dielectric function files for each of the two layers.
- `FRMFIL` – **List of dipole locations and "compositions" obtained from a file.**
This option causes **DDSCAT** to read the target geometry information from a file `shape.dat` instead of automatically generating one of the geometries listed above. The `shape.dat` file is read by routine `REASHP` (file `reashp.f`). The user can customize `REASHP` as needed to conform to the manner in which the target geometry is stored in file `shape.dat`. However, as supplied, `REASHP` expects the file `shape.dat` to have the following structure:
  - one line containing a description; the first 67 characters will be read and printed in various output statements.
  - `N` = number of dipoles in target
  - a_{1x} a_{1y} a_{1z} = x,y,z components of $\mathbf{a_1}$
  - a_{2x} a_{2y} a_{2z} = x,y,z components of $\mathbf{a_2}$
  - (line containing comments)
  - $dummy$ `IXYZ(1,1) IXYZ(1,2) IXYZ(1,3) ICOMP(1,1) ICOMP(1,2) ICOMP(1,3)`
  - $dummy$ `IXYZ(2,1) IXYZ(2,2) IXYZ(2,3) ICOMP(2,1) ICOMP(2,2) ICOMP(2,3)`
  - $dummy$ `IXYZ(3,1) IXYZ(3,2) IXYZ(3,3) ICOMP(3,1) ICOMP(3,2) ICOMP(3,3)`
  - ...
  - $dummy$ `IXYZ(J,1) IXYZ(J,2) IXYZ(J,3) ICOMP(J,1) ICOMP(J,2) ICOMP(J,3)`
  - ...
  - $dummy$ `IXYZ(N,1) IXYZ(N,2) IXYZ(N,3) ICOMP(N,1) ICOMP(N,2) ICOMP(N,3)`

The user should be able to easily modify these routines, or write new routines, to generate targets with other geometries. The user should first examine the routine `target.f` and modify it to call any new target generation routines desired. Alternatively, targets may be generated separately, and the target description (locations of dipoles and "composition" corresponding to x,y,z dielectric properties at each dipole site) read in from a file by invoking the option FRMFIL in ddscat.f.

It is often desirable to be able to run the target generation routines without running the entire **DDSCAT** code. We have therefore provided a program CALLTARGET which allows the user to generate targets interactively; to create this executable just type[17]

```
make calltarget .
```

The program calltarget is to be run interactively; the prompts are self-explanatory. You may need to edit the code to change the device number IDVOUT as for DDSCAT (see §6.1 above).

After running, calltarget will leave behind an ASCII file target.out which is a list of the occupied lattice sites in the last target generated. The format of target.out is the same as the format of the shape.dat files read if option FRMFIL is used (see above). Therefore you can simply

```
mv target.out shape.dat
```

and then use DDSCAT with the option FRMFIL in order to input a target shape generated by CALLTARGET.

20 Scattering Directions

DDSCAT calculates scattering in selected directions, and elements of the scattering matrix are reported in the output files *wxxryykzzz*.sca . The scattering direction is specified through angles θ_s and ϕ_s (not to be confused with the angles Θ and Φ which specify the orientation of the target relative to the incident radiation!).

The scattering angle θ_s is simply the angle between the incident beam (along direction $\hat{\mathbf{x}}$) and the scattered beam ($\theta_s = 0$ for forward scattering, $\theta_s = 180°$ for backscattering).

The scattering angle ϕ_s specifies the orientation of the "scattering plane" relative to the $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ plane in the Lab Frame. When $\phi_s = 0$ the scattering plane is assumed to coincide with the $\hat{\mathbf{x}}, \hat{\mathbf{y}}$ plane. When $\phi_s = 90°$ the scattering plane is assumed to coincide with the $\hat{\mathbf{x}}, \hat{\mathbf{z}}$ plane. Within the scattering plane the scattering directions are specified by $0 \leq \theta_s \leq 180°$.

Scattering directions for which the scattering properties are to be calculated are set in the file ddscat.par by specifying one or more scattering planes (determined by the value of ϕ_s) and for each scattering plane, the number and range of θ_s values. The only limitation is that the number of scattering directions not exceed the parameter MXSCA in DDSCAT.f (in the code as distributed it is set to MXSCA=1000).

21 Incident Polarization State

Recall that the "Lab Frame" is defined such that the incident radiation is propagating along the $\hat{\mathbf{x}}$ axis. **DDSCAT.6.0** allows the user to specify a general elliptical polarization for the incident radiation, by specifying the (complex) polarization vector $\hat{\mathbf{e}}_{01}$. The orthonormal polarization state $\hat{\mathbf{e}}_{02} = \hat{\mathbf{x}} \times \hat{\mathbf{e}}_{01}^*$ is generated automatically if ddscat.par specifies IORTH=2.

For incident linear polarization, one can simply set $\hat{\mathbf{e}}_{01} = \hat{\mathbf{y}}$ by specifying (0,0) (1,0) (0,0) in ddscat.par; then $\hat{\mathbf{e}}_{02} = \hat{\mathbf{z}}$ For polarization mode $\hat{\mathbf{e}}_{01}$ to correspond to right-handed circular polarization, set $\hat{\mathbf{e}}_{01} = (\hat{\mathbf{y}} + i\hat{\mathbf{z}})/\sqrt{2}$ by specifying (0,0) (1,0) (0,1) in ddscat.par (**DDSCAT.6.0** automatically takes care of the normalization of $\hat{\mathbf{e}}_{01}$); then $\hat{\mathbf{e}}_{02} = (-i\hat{\mathbf{y}} + \hat{\mathbf{z}})/\sqrt{2}$, corresponding to left-handed circular polarization.

[17] Non-Unix sites: The source code for CALLTARGET is in the file MISC.FOR. You must compile and link this to ERRMSG, GETSET, REASHP, TAR2EL, TAR2SP, TAR3EL, TARBLOCKS, TARCEL, TARCYL, TARELL, TARGET, TARHEX, TARREC, TARTET, and WRIMSG. The source code for ERRMSG is in SRC2.FOR, GETSET is in SRC5.FOR, and the rest are in SRC8.FOR and SRC9.FOR .

22 Averaging over Scattering: $g(1) = \langle\cos\theta_s\rangle$, etc.

DDSCAT automatically carries out numerical integration of various scattering properties. In particular, it calculates the mean value $g(1) = \langle\cos\theta_s\rangle$ for the scattered intensity for each incident polarization state. This is accomplished by evaluating the scattered intensity for `ICTHM` different values of θ_s (including $\theta_s = 0$ and $\theta_s = \pi$), and taking a weighted sum. For each value of θ_s except 0 and π, the scattering intensity is evaluated for `IPHM` different values of the scattering angle ϕ_s. The angular integration over θ_s is accomplished using Simpson's rule (with uniform intervals in $\cos\theta_s$), so `ICTHM` should be an *odd* number. The angular integration over ϕ_s is accomplished by sampling uniformly in ϕ_s with uniform weighting; `IPHM` can be either even or odd.

The following quantities are evaluated by this angular integration:

- $\mathbf{g} = \langle\cos\theta_s\rangle\hat{\mathbf{x}} + \langle\sin\theta_s\cos\phi_s\rangle\hat{\mathbf{y}} + \langle\sin\theta_s\sin\phi_s\rangle\hat{\mathbf{z}}$ (see §15);
- $\mathbf{Q}_\Gamma$ (see §15).

It is important that the user recognize that accurate evaluation of these angular averages requires adequate sampling over scattering angles. For small values of the size parameter $x = 2\pi a_{\rm eff}/\lambda$, the angular distribution of scattered radiation has a dipolar character and the sampling in θ_s and ϕ_s does not need to be very fine, so `ICTHM` and `IPHM` need not be large. For larger values of the size parameter x, however, higher multipoles in the scattered radiation field become important, and finer sampling in θ_s and ϕ_s is required. We do not have any foolproof prescription to offer, since the scattering pattern will depend upon the target geometry and dielectric constant in addition to overall size parameter. However, as a very rough guide, we suggest that the user specify values of `ICTHM` and `IPHM` satisfying

$$\begin{aligned} \texttt{ICTHM} &> 5(1+x) \quad , & (31)\\ \texttt{IPHM} &> 2(1+x) \quad . & (32) \end{aligned}$$

The sample `ddscat.par` file supplied has `ICTHM` = 33 and `IPHM` = 12; the above criteria would suggest that this would be suitable for $x < 5$.

The cpu time required for evaluation of these angular averages is proportional to $[2+\texttt{IPHM}(\texttt{ICTHM}-2)]$. Since the computational time spent in evaluating these angular integrals can be a significant part of the total, it is important to choose values of `ICTHM` and `IPHM` which will provide a suitable balance between accuracy (in this part of the overall calculation) and cpu time.

Within one scattering plane, the scattered intensity tends to have approximately $(1 + x)$ peaks for $0 \leq \theta_s \leq \pi$, so that the above prescription for `ICTHM` would have at least 5 sampling points per maximum. The angular distribution over ϕ_s is usually not as structured as that over θ_s so we suggest that `IPHM` need not be as large as `ICTHM`. We have refrained from "hard-wiring" the values of `ICTHM` and `IPHM` because we are not confident of the reliability of the recommended criteria (31,32) – it is up to the user to specify appropriate values of `ICTHM` and `IPHM` according to the requirements of the problem being addressed.

23 Mueller Matrix for Scattering in Selected Directions

23.1 Two Orthogonal Incident Polarizations (`IORTH=2`)

DDSCAT.6.0 internally computes the scattering properties of the dipole array in terms of a complex scattering matrix $f_{ml}(\theta_s, \phi_s)$ (Draine 1988), where index $l = 1, 2$ denotes the incident polarization state, $m = 1, 2$ denotes the scattered polarization state, and θ_s,ϕ_s specify the scattering direction. Normally **DDSCAT** is used with `IORTH=2` in `ddscat.par`, so that the scattering problem will be solved for both incident polarization states ($l = 1$ and 2); in this subsection it will be assumed that this is the case.

Incident polarization states $l = 1, 2$ correspond to polarization states $\hat{\mathbf{e}}_{01}, \hat{\mathbf{e}}_{02}$; recall that polarization state $\hat{\mathbf{e}}_{01}$ is user-specified, and $\hat{\mathbf{e}}_{02} = \hat{\mathbf{x}} \times \hat{\mathbf{e}}_{01}^*$. Scattered polarization state $m = 1$ corresponds to linear polarization of the scattered wave parallel to the scattering plane ($\hat{\mathbf{e}}_1 = \hat{\mathbf{e}}_{\parallel s} = \hat{\theta}_s$) and $m = 2$ corresponds to linear polarization perpendicular to the scattering plane (in the $+\hat{\phi}_s$ direction). The scattering matrix f_{ml} was defined (Draine 1988) so that the scattered electric field $\mathbf{E}_s$ is related to the incident electric field $\mathbf{E}_i(0)$ at the origin (where the target is assumed to be located) by

$$\begin{pmatrix} \mathbf{E}_s \cdot \hat{\theta}_s \\ \mathbf{E}_s \cdot \hat{\phi}_s \end{pmatrix} = \frac{\exp(i\mathbf{k}\cdot\mathbf{r})}{kr} \begin{pmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{01} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{02} \end{pmatrix} \quad . \tag{33}$$

The 2×2 complex *amplitude scattering matrix* (with elements S_1, S_2, S_3, and S_4) is defined so that (see Bohren & Huffman 1983)

$$\begin{pmatrix} \mathbf{E}_s \cdot \hat{\theta}_s \\ -\mathbf{E}_s \cdot \hat{\phi}_s \end{pmatrix} = \frac{\exp(i\mathbf{k}\cdot\mathbf{r})}{-ikr} \begin{pmatrix} S_2 & S_3 \\ S_4 & S_1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\parallel} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\perp} \end{pmatrix} \quad , \tag{34}$$

where $\hat{\mathbf{e}}_{i\parallel}, \hat{\mathbf{e}}_{i\perp}$ are (real) unit vectors for incident polarization parallel and perpendicular to the scattering plane (with the customary definition of $\hat{\mathbf{e}}_{i\perp} = \hat{\mathbf{e}}_{i\parallel} \times \hat{\mathbf{x}}$).

From (33,34) we may write

$$\begin{pmatrix} S_2 & S_3 \\ S_4 & S_1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\parallel} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\perp} \end{pmatrix} = -i \begin{pmatrix} f_{11} & f_{12} \\ -f_{21} & -f_{22} \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{01}^* \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{02}^* \end{pmatrix} \quad . \tag{35}$$

Let

$$a \equiv \hat{\mathbf{e}}_{01}^* \cdot \hat{\mathbf{y}} \quad , \tag{36}$$
$$b \equiv \hat{\mathbf{e}}_{01}^* \cdot \hat{\mathbf{z}} \quad , \tag{37}$$
$$c \equiv \hat{\mathbf{e}}_{02}^* \cdot \hat{\mathbf{y}} \quad , \tag{38}$$
$$d \equiv \hat{\mathbf{e}}_{02}^* \cdot \hat{\mathbf{z}} \quad . \tag{39}$$

Note that since $\hat{\mathbf{e}}_{01}, \hat{\mathbf{e}}_{02}$ could be complex (i.e., elliptical polarization), the quantities a, b, c, d are complex. Then

$$\begin{pmatrix} \hat{\mathbf{e}}_{01}^* \\ \hat{\mathbf{e}}_{02}^* \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \hat{\mathbf{y}} \\ \hat{\mathbf{z}} \end{pmatrix} \tag{40}$$

and eq. (35) can be written

$$\begin{pmatrix} S_2 & S_3 \\ S_4 & S_1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\parallel} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\perp} \end{pmatrix} = i \begin{pmatrix} -f_{11} & -f_{12} \\ f_{21} & f_{22} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{y}} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{z}} \end{pmatrix} \quad . \tag{41}$$

The incident polarization states $\hat{\mathbf{e}}_{i\parallel}$ and $\hat{\mathbf{e}}_{i\perp}$ are related to $\hat{\mathbf{y}}, \hat{\mathbf{z}}$ by

$$\begin{pmatrix} \hat{\mathbf{y}} \\ \hat{\mathbf{z}} \end{pmatrix} = \begin{pmatrix} \cos\phi_s & \sin\phi_s \\ \sin\phi_s & -\cos\phi_s \end{pmatrix} \begin{pmatrix} \hat{\mathbf{e}}_{i\parallel} \\ \hat{\mathbf{e}}_{i\perp} \end{pmatrix} ; \tag{42}$$

substituting (42) into (41) we obtain

$$\begin{pmatrix} S_2 & S_3 \\ S_4 & S_1 \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\parallel} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\perp} \end{pmatrix} = i \begin{pmatrix} -f_{11} & -f_{12} \\ f_{21} & f_{22} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \cos\phi_s & \sin\phi_s \\ \sin\phi_s & -\cos\phi_s \end{pmatrix} \begin{pmatrix} \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\parallel} \\ \mathbf{E}_i(0) \cdot \hat{\mathbf{e}}_{i\perp} \end{pmatrix} \tag{43}$$

Eq. (43) must be true for all $\mathbf{E}_i(0)$, so we obtain an expression for the complex scattering amplitude matrix in terms of the f_{ml}:

$$\begin{pmatrix} S_2 & S_3 \\ S_4 & S_1 \end{pmatrix} = i \begin{pmatrix} -f_{11} & -f_{12} \\ f_{21} & f_{22} \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \cos\phi_s & \sin\phi_s \\ \sin\phi_s & -\cos\phi_s \end{pmatrix} \quad . \tag{44}$$

This provides the 4 equations used in subroutine GETMUELLER to compute the amplitude scattering matrix elements:

$$S_1 = -i\left[f_{21}(b\cos\phi_s - a\sin\phi_s) + f_{22}(d\cos\phi_s - c\sin\phi_s)\right] \quad , \tag{45}$$
$$S_2 = -i\left[f_{11}(a\cos\phi_s + b\sin\phi_s) + f_{12}(c\cos\phi_s + d\sin\phi_s)\right] \quad , \tag{46}$$
$$S_3 = i\left[f_{11}(b\cos\phi_s - a\sin\phi_s) + f_{12}(d\cos\phi_s - c\sin\phi_s)\right] \quad , \tag{47}$$
$$S_4 = i\left[f_{21}(a\cos\phi_s + b\sin\phi_s) + f_{22}(c\cos\phi_s + d\sin\phi_s)\right] \quad . \tag{48}$$

23.2 Stokes Parameters

It is both convenient and customary to characterize both incident and scattered radiation by 4 "Stokes parameters" – the elements of the "Stokes vector". There are different conventions in the literature; we adhere to the definitions of the Stokes vector (I,Q,U,V) adopted in the excellent treatise by Bohren & Huffman (1983), to which the reader is referred for further detail. Here are some examples of Stokes vectors $(I,Q,U,V) = (1,Q/I,U/I,V/I)I$:

- $(1,0,0,0)I$: unpolarized light (with intensity I);
- $(1,1,0,0)I$: 100% linearly polarized with **E** parallel to the scattering plane;
- $(1,-1,0,0)I$: 100% linearly polarized with **E** perpendicular to the scattering plane;
- $(1,0,1,0)I$: 100% linearly polarized with **E** at +45° relative to the scattering plane;
- $(1,0,-1,0)I$: 100% linearly polarized with **E** at -45° relative to the scattering plane;
- $(1,0,0,1)I$: 100% right circular polarization (*i.e.,* negative helicity);
- $(1,0,0,-1)I$: 100% left circular polarization (*i.e.,* positive helicity).

23.3 Relation Between Stokes Parameters of Incident and Scattered Radiation: The Mueller Matrix

It is convenient to describe the scattering properties in terms of the 4×4 Mueller matrix relating the Stokes parameters (I_i, Q_i, U_i, V_i) and (I_s, Q_s, U_s, V_s) of the incident and scattered radiation:

$$\begin{pmatrix} I_s \\ Q_s \\ U_s \\ V_s \end{pmatrix} = \frac{1}{k^2 r^2} \begin{pmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{pmatrix} \begin{pmatrix} I_i \\ Q_i \\ U_i \\ V_i \end{pmatrix} \quad . \tag{49}$$

Once the amplitude scattering matrix elements are obtained, the Mueller matrix elements can be computed (Bohren & Huffman 1983):

$$S_{11} = \left(|S_1|^2 + |S_2|^2 + |S_3|^2 + |S_4|^2\right)/2 \quad ,$$
$$S_{12} = \left(|S_2|^2 - |S_1|^2 + |S_4|^2 - |S_3|^2\right)/2 \quad ,$$
$$S_{13} = \mathrm{Re}\left(S_2 S_3^* + S_1 S_4^*\right) \quad ,$$
$$S_{14} = \mathrm{Im}\left(S_2 S_3^* - S_1 S_4^*\right) \quad ,$$

$$
\begin{aligned}
S_{21} &= \left(|S_2|^2 - |S_1|^2 + |S_3|^2 - |S_4|^2\right)/2 \quad , \\
S_{22} &= \left(|S_1|^2 + |S_2|^2 - |S_3|^2 - |S_4|^2\right)/2 \quad , \\
S_{23} &= \mathrm{Re}\left(S_2 S_3^* - S_1 S_4^*\right) \quad , \\
S_{24} &= \mathrm{Im}\left(S_2 S_3^* + S_1 S_4^*\right) \quad , \\
S_{31} &= \mathrm{Re}\left(S_2 S_4^* + S_1 S_3^*\right) \quad , \\
S_{32} &= \mathrm{Re}\left(S_2 S_4^* - S_1 S_3^*\right) \quad , \\
S_{33} &= \mathrm{Re}\left(S_1 S_2^* + S_3 S_4^*\right) \quad , \\
S_{34} &= \mathrm{Im}\left(S_2 S_1^* + S_4 S_3^*\right) \quad , \\
S_{41} &= \mathrm{Im}\left(S_4 S_2^* + S_1 S_3^*\right) \quad , \\
S_{42} &= \mathrm{Im}\left(S_4 S_2^* - S_1 S_3^*\right) \quad , \\
S_{43} &= \mathrm{Im}\left(S_1 S_2^* - S_3 S_4^*\right) \quad , \\
S_{44} &= \mathrm{Re}\left(S_1 S_2^* - S_3 S_4^*\right) \quad .
\end{aligned} \tag{50}
$$

These matrix elements are computed in `DDSCAT` and passed to subroutine `WRITESCA` which handles output of scattering properties. Although the Muller matrix has 16 elements, only 9 are independent.

The user can select up to 9 distinct Muller matrix elements to be printed out in the output files w*xx*r*yy*k*zzz*`.sca` and w*xx*r*yy*`ori.avg`; this choice is made by providing a list of indices in `ddscat.par` (see Appendix A).

If the user does not provide a list of elements, `WRITESCA` will provide a "default" set of 6 selected elements: S_{11}, S_{21}, S_{31}, S_{41} (these 4 elements describe the intensity and polarization state for scattering of unpolarized incident radiation), S_{12}, and S_{13}.

In addition, `WRITESCA` writes out the linear polarization P of the scattered light for incident unpolarized light:

$$P = \frac{(S_{21}^2 + S_{31}^2)^{1/2}}{S_{11}} \quad . \tag{51}$$

23.4 One Incident Polarization State Only (`IORTH=1`)

In some cases it may be desirable to limit the calculations to a single incident polarization state – for example, when each solution is very time-consuming, and the target is known to have some symmetry so that solving for a single incident polarization state may be sufficient for the required purpose. In this case, set `IORTH=1` in `ddscat.par`.

When `IORTH=1`, only f_{11} and f_{21} are available; hence, **DDSCAT** cannot automatically generate the Mueller matrix elements. In this case, the output routine `WRITESCA` writes out the quantities $|f_{11}|^2$, $|f_{21}|^2$, $\mathrm{Re}(f_{11} f_{21}^*)$, and $\mathrm{Im}(f_{11} f_{21}^*)$ for each of the scattering directions.

Note, however, that if `IPHI` is greater than 1, **DDSCAT** will automatically set `IORTH=2` even if `ddscat.par` specified `IORTH=1`: this is because when more than one value of the target orientation angle Φ is required, there is no additional "cost" to solve the scattering problem for the second incident polarization state, since when solutions are available for two orthogonal states for some particular target orientation, the solution may be obtained for another target orientation differing only in the value of Φ by appropriate linear combinations of these solutions. Hence we may as well solve the "complete" scattering problem so that we can compute the complete Mueller matrix.

24 Using MPI (Message Passing Interface)

Many (probably most) users of **DDSCAT** will wish to calculate scattering and absorption by a given target for more than one orientation relative to the incident radiation (e.g., when calculating orientational averages). `DDSCAT` can automatically carry out the calculations over different target orientations. Normally, these calculations will be carried out sequentially. However, on a multiprocessor system [either a symmetric multiprocessor (SMP) system, or a cluster of networked computers] with `MPI` (Message Passing Interface) software installed, **DDSCAT.6.0** allows the user to carry out parallel calculations of scattering by different target orientations, with the results gathered together and with orientational averages calculated just as they are when sequential calculations are carried out. Note that the MPI-capable executable can also be used for ordinary serial calculations using a single cpu.

More than one implementation of `MPI` exists. `MPI` support within **DDSCAT.6.0** is compliant with MPI-1.2 and MPI-2 standards[18], and should be usable under any implementation of `MPI` that is compatible with those standards.

At Princeton University Department of Astrophysical Sciences we are using `MPICH`[19], a publicly-available implementation of `MPI`.

24.1 Source Code for the MPI-compatible version of DDSCAT

In order to use DDSCAT with MPI, the user must first make sure that the correct version of the code is being used. To prepare the MPI-capable version:

1. In `DDSCAT.f` :
   - Make sure the line
     ```
             INCLUDE 'mpif.h'
     ```
     is not commented out.
   - Make sure the line
     ```
     C      INTEGER MPI_COMM_WORLD
     ```
     is commented out.
2. In `mpisubs.f, SUBROUTINE COLSUM` :
   - Make sure the line
     ```
             INCLUDE 'mpif.h'
     ```
     is not commented out.
   - Make sure the line
     ```
     C      INTEGER MPI_COMM_WORLD,MPI_SUM,MPI_REAL,MPI_COMPLEX
     ```
     is commented out.
3. In `mpisubs.f, SUBROUTINE SHARE1` :
   - Make sure the line
     ```
             INCLUDE 'mpif.h'
     ```
     is not commented out.
   - Make sure the lines
     ```
     C      INTEGER MPI_CHARACTER,MPI_COMM_WORLD,MPI_INTEGER,
     C     &    MPI_INTEGER2,MPI_REAL,MPI_COMPLEX
     ```
     are commented out.
4. In `mpisubs.f, SUBROUTINE SHARE2` :

[18] `http://www.mpi-forum.org/`
[19] `http://www.mcs.anl.gov/mpi/mpich`

- Make sure the line
  `INCLUDE 'mpif.h'`
  is not commented out.
- Make sure the line
  `C      INTEGER MPI_COMM_WORLD,MPI_REAL,MPI_COMPLEX`
  is commented out.

24.2 Compiling and Linking the MPI-compatible version of DDSCAT

In the `Makefile`, enable the following definitions:

```
MPIsrc     = mpi_subs.f
MPIobj     = mpi_subs.o
```

At Princeton University Dept. of Astrophysical Sciences we are currently using `mpich-1.2.4`[20]. `mpich` provides a "compiler wrapper" `mpif77` that is supposed to automatically set the variables `MPICH_F77` and `MPICH_F77LINKER` to values appropriate for use of the `pgf77` compiler from PGI[21] which is installed on our beowulf cluster. Other compilers, e.g., the Intel f77 compiler, can also be employed.

24.3 Code Execution Under MPI

Local installations of `MPI` will vary – you should consult with someone familiar with way `MPI` is installed and used on your system.

At Princeton University Dept. of Astrophysical Sciences we use `PBS` (Portable Batch System)[22] to schedule jobs. `MPI` jobs are submitted using `PBS` by first creating a shell script such as the following example file `pbs.submit`:

```
#!/bin/bash
#PBS -l nodes=2:ppn=1
#PBS -l mem=1200MB,pmem=300MB
#PBS -m bea
#PBS -j oe
cd $PBS_O_WORKDIR
/usr/local/bin/mpiexec ddscat
```

The lines beginning with `#PBS -l` specify the required resources:
`#PBS -l nodes=2:ppn=1` specifies that 2 nodes are to be used, with 1 processor per node.
`#PBS -l mem=1200MB,pmem=300MB` specifies that the total memory required (`mem`) is 1200MB, and the maximum physical memory used by any single process (`pmem`) is 300MB. The actual definition of `mem` is not clear, but in practice it seems that it should be set equal to 2×(`nodes`)×(`ppn`)×(`pmem`).
`#PBS -m bea` specifies that PBS should send email when the job begins (`b`), and when it ends (`e`) or aborts (`a`).
`#PBS -j oe` specifies that the output from stdout and stderr will be merged, intermixed, as stdout.
This example assumes that the executable `dscat` is located in the same directory where the code is to execute and write its output. If `ddscat` is located in another directory, simply give the full pathname. to it. The `qsub` command is used to submit the `PBS` job:

```
% qsub pbs.submit
```

[20]`http://www-unix.mcs.anl.gov/mpi/mpich/`
[21]`http://www.pgroup.com`
[22]`http://www.openpbs.org`

As the calculation proceeds, the usual output files will be written to this directory: for each wavelength, target size, and target orientation, there will be a file w*aa*r*bb*k*ccc*.sca, where *aa=00, 01, 02, ...* specifies the wavelength, *bb=00, 01, 02, ...* specifies the target size, and *cc=000, 001, 002, ...* specifies the orientation. For each wavelength and target size there will also be a file w*aa*r*bb*ori.avg with orientationally-averaged quantities. Finally, there will also be tables qtable, and qtable2 with orientationally-averaged cross sections for each wavelength and target size.

In addition, each processor employed will write to its own log file ddscat.log_*nnn*, where *nnn=000, 001, 002, ...* These files contain information concerning cpu time consumed by different parts of the calculation, convergence to the specified error tolerance, etc. If you are uncertain about how the calculation proceeded, examination of these log files is recommended.

25 Graphics and Postprocessing

25.1 IDL

At present, we do not offer a comprehensive package for **DDSCAT** data postprocessing and graphical display in IDL. However, there are several developments worth mentioning: First, we offer several output capabilities from within **DDSCAT**: ASCII (see §10.1), FORTRAN unformatted binary (see §10.2), and netCDF portable binary (see §10.3). Second, we offer several skeleton IDL utilities:

- bhmie.pro is our translation to IDL of the popular Bohren-Huffman code which calculates efficiencies for spherical particles using Mie theory.
- readbin.pro reads the FORTRAN unformatted binary file written by routine writebin.f. The variables are stored in a common block.
- readnet.pro reads NetCDF portable binary file and should be the method of choice for IDL users. It offers random data access.
- mie.pro is an example of an interface to binary files and the Bohren-Huffman code. It plots a comparison of **DDSCAT** results with scattering by equivalent radius spheres.

At present the IDL code is experimental.

26 Miscellanea

Additional source code, refractive index files, etc., contributed by users will be located in the directory DDA/misc. These routines and files should be considered to be **not supported** by Draine and Flatau – *caveat receptor!* These routines and files should be accompanied by enough information (e.g., comments in source code) to explain their use.

27 Finale

This User Guide is somewhat inelegant, but we hope that it will prove useful. The structure of the ddscat.par file is intended to be simple and suggestive so that, after reading the above notes once, the user may not have to refer to them again.

The file rel_notes in DDA/doc lists known bugs in **DDSCAT.6.0** . Up-to-date release notes will be available at the **DDSCAT** web site,

http://www.astro.princeton.edu/~draine/DDSCAT.html

Users are encouraged to provide B. T. Draine (draine@astro.princeton.edu) with their email address; bug reports, and any new releases of **DDSCAT**, will be made known to those who do!

P. J. Flatau maintains the WWW page "SCATTERLIB - Light Scattering Codes Library" with URL `http://atol.ucsd.edu/~pflatau`. The SCATTERLIB Internet site is a library of light scattering codes. Emphasis is on providing source codes (mostly FORTRAN). However, other information related to scattering on spherical and non-spherical particles is collected: an extensive list of references to light scattering methods, refractive index, etc. This URL page contains section on the discrete dipole approximation.

Concrete suggestions for improving **DDSCAT** (and this User Guide) are welcomed. If you wish to cite this User Guide, we suggest the following citation:

Draine, B.T., & Flatau, P.J. 2000, "User Guide for the Discrete Dipole Approximation Code DDSCAT (Version 6.0)", http://arxiv.org/abs/astro-ph/??????

Finally, the authors have one special request: We would very much appreciate preprints and (especially!) reprints of any papers which make use of **DDSCAT**!

28 Acknowledgments

- The routine `ESELF` making use of the FFT was originally written by Jeremy Goodman, Princeton University Observatory.
- The FFT routine `FOURX` (no longer used by **DDSCAT.6.0**, but included for comparison of FFT routines by `TSTFFT`) is based on a FFT routine written by Norman Brenner (Brenner 1969).
- The routine `REFICE` was written by Steven B. Warren, based on Warren (1984).
- The routine `REFWAT` was written by Eric A. Smith.
- The `GPFAPACK` package was written by Clive Temperton (1992), and generously made available by him for use with `DDSCAT`.
- We make use of routines from the `LAPACK`[23] package (Anderson *et al.* 1995), the result of work by Jack Dongarra and others at the Univ. of Tennessee, Univ. of California Berkeley, NAG Ltd., Courant Institute, Argonne National Lab, and Rice University.
- The `FFTW`[24] package, to which we link, was written by Matteo Frigo and Steven G. Johnson (`fftw@fftw.org`).
- Much of the work involved in modifying `DDSCAT` to use `MPI` was done by Matthew Collinge, Princeton University.

We are deeply indebted to all of these authors for making their code available.

We wish also to acknowledge bug reports and suggestions from **DDSCAT** users, including Henrietta Lemke, Timo Nousianen, and Mike Wolff.

Development of **DDSCAT** was supported in part by National Science Foundation grants AST-8341412, AST-8612013, AST-9017082, AST-9319283, AST-9616429, AST-9988126 to BTD, in part by support from the Office of Naval Research Young Investigator Program to PJF, in part by DuPont Corporate Educational Assistance to PJF, and in part by the United Kingdom Defence Research Agency.

[23] `http://netlib.org`
[24] `http://www.fftw.org`

References

[1] Anderson, E., *et al.* 1995, *LAPACK Users' Guide*, (Philadelphia, SIAM).

[2] Bohren, C.F., and Huffman, D.R. 1983, *Absorption and Scattering of Light by Small Particles* (New York: Wiley-Interscience).

[3] Brenner, N.M. 1969, *IEEE Trans. Audio Electroacoust.* **AU-17**, 128.

[4] Draine, B.T. 1988, "The Discrete-Dipole Approximation and Its Application to Interstellar Graphite Grains", *Astrophysical J.*, **333**, 848-872.

[5] Draine, B.T. 2000, "The Discrete-Dipole Approximation for Light Scattering by Irregular Targets", in *Light Scattering by Nonspherical Particles: Theory, Measurements, and Geophysical Applications*, ed. M.I. Mishchenko, J.W. Hovenier, and L.D. Travis (N.Y.: Academic Press), 131-145.

[6] Draine, B.T., and Flatau, P.J. 1994, "Discrete-dipole approximation for scattering calculations", *J. Opt. Soc. Am. A*, **11**, 1491-1499.

[7] Draine, B.T., and Goodman, J.J. 1993, "Beyond Clausius-Mossotti: Wave Propagation on a Polarizable Point Lattice and the Discrete Dipole Approximation", *Astrophysical J.*, **405**, 685-697.

[8] Draine, B.T., and Weingartner, J.C. 1996, "Radiative Torques on Interstellar Grains: I. Superthermal Rotation", *Astrophysical J.*, **470**, 551-565.

[9] Draine, B.T., and Weingartner, J.C. 1997, "Radiative Torques on Interstellar Grains: II. Alignment with the Magnetic Field", *Astrophysical J.*, **480**, 633-646.

[10] Flatau, P.J. 1997, "Improvements in the discrete dipole approximation method of computing scattering and absorption", *Optics Lett.*, **22**, 1205-1207

[11] Flatau, P.J., Stephens, G.L., and Draine, B.T. 1990, "Light scattering by rectangular solids in the discrete-dipole approximation: a new algorithm exploiting the Block-Toeplitz structure", *J. Opt. Soc. Am. A*, **7**, 593-600.

[12] Goedecke, G.H., and O'Brien, S.G. 1988, "Scattering by irregular inhomogeneous particles via the digitized Green's function algorithm", *Appl. Opt.*, **28**, 2431-2438.

[13] Goodman, J.J., Draine, B.T., and Flatau, P.J. 1991, "Application of FFT Techniques to the Discrete Dipole Approximation, *Opt. Lett.*, **16**, 1198-1200.

[14] Hage, J.I., and Greenberg, J.M. 1990, "A Model for the Optical Properties of Porous Grains", *Astrophysical J.*, **361**, 251-259.

[15] Petravic, M., & Kuo-Petravic, G. 1979, "An ILUCG algorithm which minimizes in the Euclidean norm", *J. Computational Phys.*, 32, 263-269.

[16] Purcell, E.M., and Pennypacker, C.R. 1973, "Scattering and Absorption of Light by Nonspherical Dielectric Grains", Astrophysical J., **186**, 705-714.

[17] Temperton, C.J. 1983, "Self-sorting mixed-radix Fast Fourier Transforms", *J. Comp. Phys.*, **52**, 1-23.

[18] 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

[19] Warren, S.G. 1984, "Optical constants of ice from the ultraviolet to the microwave", *Appl. Opt.*, **23**, 1206-1225.

A Understanding and Modifying `ddscat.par`

In order to use DDSCAT to perform the specific calculations of interest to you, it will be necessary to modify the `ddscat.par` file. Here we list the sample `ddscat.par` file, followed by a discussion of how to modify this file as needed. Note that all numerical input data in DDSCAT is read with free-format `READ(IDEV,*)...` statements. Therefore you do not need to worry about the precise format in which integer or floating point numbers are entered on a line. The crucial thing is that lines in `ddscat.par` containing numerical data have the correct number of data entries, with any informational comments appearing *after* the numerical data on a given line.

```
' =================== Parameter file ==================='
'**** PRELIMINARIES ****'
'NOTORQ'= CMTORQ*6 (DOTORQ, NOTORQ) -- either do or skip torque calculations
'PBCGST'= CMDSOL*6 (PBCGST, PETRKP) -- select solution method
'GPFAFT'= CMETHD*6 (GPFAFT, FFTW21, CONVEX)
'LATTDR'= CALPHA*6 (LATTDR is only supported option now)
'NOTBIN'= CBINFLAG (ALLBIN, ORIBIN, NOTBIN)
'NOTCDF'= CNETFLAG (ALLCDF, ORICDF, NOTCDF)
'RCTNGL'= CSHAPE*6 (FRMFIL,ELLIPS,CYLNDR,RCTNGL,HEXGON,TETRAH,UNICYL,UNIELL)
8. 6. 4. = shape parameters PAR1, PAR2, PAR3
1         = NCOMP = number of dielectric materials
'TABLES'= CDIEL*6 (TABLES,H2OICE,H2OLIQ; if TABLES, then filenames follow...)
'diel.tab' = name of file containing dielectric function
'**** CONJUGATE GRADIENT DEFINITIONS ****'
0       = INIT (TO BEGIN WITH |X0> = 0)
1.00e-5 = ERR = MAX ALLOWED (NORM OF |G>=AC|E>-ACA|X>)/(NORM OF AC|E>)
'**** ANGLES FOR CALCULATION OF CSCA, G'
33      = ICTHM (number of theta values for evaluation of Csca and g)
12      = IPHM (number of phi values for evaluation of Csca and g)
'**** Wavelengths (micron) ****'
6.283185 6.283185 1 'INV' = wavelengths (first,last,how many,how=LIN,INV,LOG)
'**** Effective Radii (micron) **** '
1. 1. 1  'LIN' = eff. radii (first, last, how many, how=LIN,INV,LOG)
'**** Define Incident Polarizations ****'
(0,0) (1.,0.) (0.,0.) = Polarization state e01 (k along x axis)
2 = IORTH  (=1 to do only pol. state e01; =2 to also do orth. pol. state)
1 = IWRKSC (=0 to suppress, =1 to write ".sca" file for each target orient.
'**** Prescribe Target Rotations ****'
 0.   0.  1  = BETAMI, BETAMX, NBETA (beta=rotation around a1)
 0.  90.  3  = THETMI, THETMX, NTHETA (theta=angle between a1 and k)
 0.   0.  1  = PHIMIN, PHIMAX, NPHI (phi=rotation angle of a1 around k)
'**** Specify first IWAV, IRAD, IORI (normally 0 0 0) ****'
0   0   0    = first IWAV, first IRAD, first IORI (0 0 0 to begin fresh)
'**** Select Elements of S_ij Matrix to Print ****'
6 = NSMELTS = number of elements of S_ij to print (not more than 9)
11 12 21 22 31 41 = indices ij of elements to print
'**** Specify Scattered Directions ****'
0.  0. 180. 30 = phi, thetan_min, thetan_max, dtheta (in degrees) for plane A
90. 0. 180. 30 = phi, ... for plane B
```

Lines	Comments
1-2	comment lines – no need to change.
3	`NOTORQ` if torque calculation is not required;
	`DOTORQ` if torque calculation is required.
4	`PBCGST` is recommended; other option is `PETRKP` (see §14).
5	`GPFAFT` is supplied as default, but `FFTW21` is recommended if `DDSCAT` has been compiled with `FFTW` support (see §§6.5, 13); only other option is `CONVEX` (valid only on a Convex computer).
6	`LATTDR` (Draine & Goodman [2000] Lattice Dispersion Relation is only option now supported) (see §11).
7	`ALLBIN` for full unformatted binary dump (§10.2);
	`ORIBIN` for unformatted binary dump of orientational averages only;
	`NOTBIN` for no unformatted binary output.
8	`ALLCDF` for output in netCDF format (must have netCDF option enabled; cf. §10.3);
	`ORICDF` for orientational averages in netCDF format (must have netCDF option enabled).
	`NOTCDF` for no output in netCDF format;
9	specify choice of target shape (see §19 for description of options `RCTNGL`, `ELLIPS`, `TETRAH`, ...)
10	shape parameters SHPAR1, SHPAR2, SHPAR3, ... (see §19).
11	number of different dielectric constant tables (§12).
12	`TABLES` – dielectric function to be provided via input table.
13	name(s) of dielectric constant table(s) (one per line).
14	comment line – no need to change.
15	`0` is recommended value of parameter `INIT`.
16	`ERR` = error tolerance h: maximum allowed value of $\|A^\dagger E - A^\dagger AP\|/\|A^\dagger E\|$ [see eq.(18)].
17	comment line – no need to change.
18	`ICTHM` – number of θ_s values for angular averages (§22).
19	`IPHM` – number of ϕ_s values for angular averages.
20	comment line – no need to change.
21	λ – first, last, how many, how chosen.
22	comment line – no need to change.
23	$a_{\rm eff}$ – first, last, how many, how chosen.
24	comment line – no need to change.
25	specify x,y,z components of (complex) incident polarization $\hat{\mathbf{e}}_{01}$ (§21)
26	`IORTH` = 1 to do one polarization state only;
	2 to do second (orthogonal) incident polarization as well.
27	`IWRKSC` = 0 to suppress writing of ".sca" files;
	2 to enable writing of ".sca" files.
28	comment line – no need to change.
29	β (see §17) – first, last, how many .
30	Θ – first, last, how many.
31	Φ – first, last, how many.
32	comment line – no need to change.
33	`IWAV0 IRAD0 IORI0` – specify starting values of integers `IWAV IRAD IORI` (normally `0 0 0`).
34	comment line – no need to change.
35	N_S = number of scattering matrix elements (must be ≤ 9)
36	indices ij of N_S elements of the scattering matrix S_{ij}
37	comment line – no need to change.
38	ϕ_s for first scattering plane, $\theta_{s,min}$, $\theta_{s,max}$, how many θ_s values;
39,...	ϕ_s for 2nd,... scattering plane, ...

B wxxryyori.avg Files

The file w00r00ori.avg contains the results for the first wavelength (w00) and first target radius (r00) averaged over orientations (ori.avg). The w00r00ori.avg file generated by the sample calculation should look like the following:

```
 DDSCAT --- DDSCAT.6.0 [03.07.13]
 TARGET --- Rectangular prism; NX,NY,NZ=   8   6   4
 GPFAFT --- method of solution
 LATTDR --- prescription for polarizabilies
 RCTNGL --- shape
    192 = NAT0 = number of dipoles

  AEFF=   1.00000 = effective radius (physical units)
  WAVE=   6.28319 = wavelength (physical units)
K*AEFF=   1.00000 = 2*pi*aeff/lambda
n= ( 1.3300 ,  0.0100),  eps.= (  1.7688 ,  0.0266)  |m|kd=  0.3716 for subs. 1
   TOL= 1.000E-05 = error tolerance for CCG method
 ICTHM= 33 = theta values used in comp. of Qsca,g
 IPHIM= 12 = phi values used in comp. of Qsca,g
 ( 1.00000  0.00000  0.00000) = target axis A1 in Target Frame
 ( 0.00000  1.00000  0.00000) = target axis A2 in Target Frame
 ( 0.27942  0.00000  0.00000) = k vector (latt. units) in Lab Frame
 ( 0.00000, 0.00000)( 1.00000, 0.00000)( 0.00000, 0.00000)=inc.pol.vec. 1 in LF
 ( 0.00000, 0.00000)( 0.00000, 0.00000)( 1.00000, 0.00000)=inc.pol.vec. 2 in LF
   0.000   0.000 = beta_min, beta_max ;  NBETA = 1
   0.000  90.000 = theta_min, theta_max; NTHETA= 3
   0.000   0.000 = phi_min, phi_max   ;   NPHI = 1
 Results averaged over   3 target orientations
                   and   2 incident polarizations
          Qext       Qabs       Qsca     g(1)=<cos>     Qbk        Qpha
 JO=1:  1.3296E-01 3.2687E-02 1.0028E-01  2.3451E-01 5.5520E-03 4.8356E-01
 JO=2:  9.0628E-02 2.3972E-02 6.6655E-02  2.6726E-01 3.4303E-03 4.1588E-01
 mean:  1.1180E-01 2.8330E-02 8.3466E-02  2.4758E-01 4.4912E-03 4.4972E-01
 Qpol=  4.2337E-02                                      dQpha= 6.7685E-02
         Qsca*g(1)   Qsca*g(2)   Qsca*g(3)   iter  mxiter
 JO=1:  2.3516E-02  1.1256E-03  3.2952E-10      3    576
 JO=2:  1.7814E-02  2.7678E-03  7.6056E-10      3    576
 mean:  2.0665E-02  1.9467E-03  5.4504E-10
            ** Mueller matrix elements for selected scattering directions **
theta   phi    Pol.    S_11       S_12       S_21       S_22       S_31       S_41
  0.0   0.0  0.15320  5.167E-02  7.916E-03  7.916E-03  5.167E-02  8.165E-11  2.565E-11
 30.0   0.0  0.01117  4.046E-02  4.518E-04  4.518E-04  4.046E-02  7.106E-12  2.170E-11
 60.0   0.0  0.47184  2.169E-02 -1.023E-02 -1.023E-02  2.169E-02  3.183E-11 -5.271E-11
 90.0   0.0  0.99888  1.193E-02 -1.192E-02 -1.192E-02  1.193E-02  5.394E-12  5.443E-11
120.0   0.0  0.48743  1.170E-02 -5.702E-03 -5.702E-03  1.170E-02 -3.811E-11  3.570E-11
150.0   0.0  0.06956  1.403E-02  9.758E-04  9.758E-04  1.403E-02 -4.999E-11 -1.284E-11
180.0   0.0  0.23621  1.411E-02  3.333E-03  3.333E-03  1.411E-02 -1.220E-11 -4.503E-12
  0.0  90.0  0.15320  5.167E-02 -7.916E-03 -7.916E-03  5.167E-02 -7.224E-10  2.865E-11
 30.0  90.0  0.28550  4.487E-02 -1.281E-02 -1.280E-02  4.486E-02 -5.036E-04  8.979E-05
 60.0  90.0  0.67857  3.040E-02 -2.063E-02 -2.062E-02  3.039E-02 -7.990E-04  1.643E-04
 90.0  90.0  0.99926  1.991E-02 -1.991E-02 -1.989E-02  1.989E-02 -7.361E-04  1.827E-04
120.0  90.0  0.72285  1.626E-02 -1.176E-02 -1.175E-02  1.625E-02 -4.415E-04  1.359E-04
150.0  90.0  0.35833  1.478E-02 -5.295E-03 -5.293E-03  1.478E-02 -1.716E-04  6.554E-05
180.0  90.0  0.23621  1.411E-02 -3.333E-03 -3.333E-03  1.411E-02  3.323E-10 -2.245E-11
```

C `wxxryykzzz.sca` Files

The `w00r00k000.sca` file contains the results for the first wavelength (`w00`), first target radius (`r00`), and first orientation (`k000`). The `w00r00k000.sca` file created by the sample calculation should look like the following:

```
 DDSCAT --- DDSCAT.6.0 [03.07.13]
 TARGET --- Rectangular prism; NX,NY,NZ=   8   6   4
 GPFAFT --- method of solution
 LATTDR --- prescription for polarizabilies
 RCTNGL --- shape
    192 = NAT0 = number of dipoles

  AEFF=   1.00000 = effective radius (physical units)
  WAVE=   6.28319 = wavelength (physical units)
K*AEFF=   1.00000 = 2*pi*aeff/lambda
n= ( 1.3300 ,  0.0100),  eps.= (  1.7688 ,  0.0266)  |m|kd=  0.3716 for subs. 1
   TOL= 1.000E-05 = error tolerance for CCG method
 ICTHM= 33 = theta values used in comp. of Qsca,g
 IPHIM= 12 = phi values used in comp. of Qsca,g
 ( 1.00000  0.00000  0.00000) = target axis A1 in Target Frame
 ( 0.00000  1.00000  0.00000) = target axis A2 in Target Frame
 ( 0.27942  0.00000  0.00000) = k vector (latt. units) in TF
 ( 0.00000, 0.00000)( 1.00000, 0.00000)( 0.00000, 0.00000)=inc.pol.vec. 1 in TF
 ( 0.00000, 0.00000)( 0.00000, 0.00000)( 1.00000, 0.00000)=inc.pol.vec. 2 in TF
 BETA =  0.000 = rotation of target around A1
 THETA=  0.000 = angle between A1 and k
  PHI =  0.000 = rotation of A1 around k
         Qext       Qabs       Qsca     g(1)=<cos>     Qbk        Qpha
 JO=1:  1.1104E-01 3.0279E-02 8.0766E-02  3.5035E-01 1.8580E-03 4.6680E-01
 JO=2:  8.6508E-02 2.4413E-02 6.2095E-02  3.6498E-01 1.3620E-03 4.1968E-01
 mean:  9.8776E-02 2.7346E-02 7.1430E-02  3.5671E-01 1.6100E-03 4.4324E-01
 Qpol=  2.4537E-02                                      dQpha= 4.7115E-02
        Qsca*g(1)   Qsca*g(2)   Qsca*g(3)   iter  mxiter
 JO=1:  2.8296E-02 -5.5908E-10 -1.2161E-09      3    576
 JO=2:  2.2663E-02  3.0799E-09  1.4463E-09      3    576
 mean:  2.5480E-02  1.2604E-09  1.1509E-10
            ** Mueller matrix elements for selected scattering directions **
theta   phi    Pol.    S_11        S_12        S_21        S_22        S_31        S_41
  0.0   0.0  0.10772  8.312E-03  8.954E-04  8.954E-04  8.312E-03 -1.837E-11  5.826E-12
 30.0   0.0  0.02296  6.734E-03 -1.546E-04 -1.546E-04  6.734E-03  7.358E-13  9.139E-12
 60.0   0.0  0.48383  3.651E-03 -1.766E-03 -1.766E-03  3.651E-03 -5.798E-12 -1.051E-11
 90.0   0.0  0.99794  1.764E-03 -1.760E-03 -1.760E-03  1.764E-03 -6.232E-12  7.791E-12
120.0   0.0  0.53222  1.251E-03 -6.658E-04 -6.658E-04  1.251E-03 -1.813E-12 -4.691E-13
150.0   0.0  0.00167  9.868E-04 -1.644E-06 -1.644E-06  9.868E-04 -5.853E-12 -2.595E-12
180.0   0.0  0.15403  8.430E-04  1.298E-04  1.298E-04  8.430E-04  3.319E-13 -3.020E-12
  0.0  90.0  0.10772  8.312E-03 -8.954E-04 -8.954E-04  8.312E-03 -7.445E-11  9.616E-12
 30.0  90.0  0.24082  7.141E-03 -1.720E-03 -1.720E-03  7.141E-03 -7.673E-11 -6.021E-12
 60.0  90.0  0.64798  4.560E-03 -2.955E-03 -2.955E-03  4.560E-03 -2.574E-11  3.576E-12
 90.0  90.0  0.99942  2.566E-03 -2.565E-03 -2.565E-03  2.566E-03 -5.111E-12 -5.125E-12
120.0  90.0  0.68918  1.631E-03 -1.124E-03 -1.124E-03  1.631E-03  8.146E-12 -8.819E-13
150.0  90.0  0.28491  1.065E-03 -3.034E-04 -3.034E-04  1.065E-03  9.598E-12  1.278E-12
180.0  90.0  0.15403  8.430E-04 -1.298E-04 -1.298E-04  8.430E-04  7.621E-12 -1.666E-12
```