LMie

A Linearized Mie Scattering Implementation
Version 0.91
April 4, 2020

Greg McGarragh

This manual describes how to install and use LMie (A Linearized Mie Scattering Implementation) version 0.91.

Contents

1 Introduction to LMie

LMie (Linearized Mie) computes the scattering properties for polydisperse homogeneous spherical particles using Mie theory [Mie, 1908]. Six different particle size distributions are supported including power law, modified power law, gamma, modified gamma, log normal, and modified bimodal log normal. The outputs include the extinction, scattering, and backscattering cross sections; asymmetry parameter, elements of the normalized scattering matrix, coefficients for the expansion of the scattering matrix in either generalized spherical functions or Legendre functions, and finally, several quantities related to the numerical integration of the size distribution including the effective radius and effective variance.

LMie has two other features that sets it apart from other Mie scattering implementations. First, it has the option to analytically generate derivatives of the outputs with respect to input parameters such as size distribution parameters and the real and imaginary parts of the complex index of refraction. This makes LMie convenient for sensitivity studies and remote sensing retrievals of particle properties. Second, LMie is parallelized over the size distribution integration with OpenMP for shared memory systems such as multi-core workstations and with MPI for clusters.

LMie's core library is coded in C with a well defined application programming interface (API) and is thread safe so that multiple instances can be called safely in shared memory multi-processor environments. Interfaces are also provided for Fortran 77 and Fortran 90 and, alternatively, LMie can be executed independently as a stand alone program. Finally, example programs that call LMie are provided for each language interface.

For up to date information regarding LMie, to download the source code distribution, and/or to view the documentation please visit the LMie web page at

http://reef.atmos.colostate.edu/~gregm/lmie/

For questions or comments or to report a bug email Greg at gregm@atmos.colostate.edu. Bug reports are greatly appreciated! If you would like to report a bug please include sample code that reproduces the bug, along with the inputs and expected outputs.

1.1 License

1.2 Conventions used in this manual

Source code such as interface definitions and examples are typeset in `typewriter font`. Source code identifiers such as variable names and function names are also typeset in bold, while function argument types, modifiers, and names, are also typeset in italics. For example, a function name will be typeset in bold typewriter font as **`func_name`**, argument types in italic typewriter font as ***`int`***, and argument names in bold italic typewriter font as ***`arg_name`***.

Internet links are typeset in the standard color blue. Links that are local to the manual are typeset in a dark red except for citations that link to their corresponding bibliography entries which are in a dark green.

2 Building and Using LMie

This section discusses, first, the process of building (compiling) LMie including the core library, the language interfaces, the example programs, and the utility programs. Then the compilation details of using the LMie core library and the appropriate language interface in your programs is outlined.

2.1 Building LMie

2.1.1 GNU Make

The standard build system uses GNU Make (other versions of UNIX Make may work but are not tested). This should work on Linux, Unix, Mac OS, and on Windows using either Cygwin or MinGW (Minimalist GNU for Windows).

The first step is to configure the build for your environment. This includes setting the compiler commands and the associated options and setting the appropriate paths to external libraries. Settings are contained in the file `make.inc` in the LMie base directory.

Compiler and associated options are contained within the section identified as "Compiler and linker settings". The commands for the compilers to use are represented by the variables `CC`, `F77`, and `F90`, for the C, Fortran 77, and Fortran 90 compilers, respectively, and the associated options are represented by the variables `CCFLAGS`, `F77FLAGS`, and `F90FLAGS`. The default settings are appropriate for GCC (GNU Compiler Collection) versions 4.2 and greater and should not have to be modified unless other compilers are being used. Note that LMie is entirely C89/90 compliant except for the use of complex types. Therefore, the C compiler must be C99 compliant. As an alternative, all of LMie's C code may be built with a C++ compiler in which case the complex support is through the C++ standard library's complex class.

The next section of `make.inc` is for OpenMP settings. If multithreading is desired then uncomment the variable appendices in this section and set the appropriate OpenMP compiler flags for the `CCFLAGS`, `F77FLAGS`, and `F90FLAGS` variables.

The next section of `make.inc` is for MPI settings. If the MPI version of LMie is desired then an MPI implementation should be made available. The variable `MPI_INCDIR` should be set to the path to the directory with the MPI include files, the variable `MPI_LIBDIR` should be set to the path to the directory with the MPI library files, and the variable `MPI_LINK` should be set to the appropriate compiler link flag. The default settings are for a MPICH2 installation located at `$HOME/opt/mpich2/`.

Once the proper settings have been set in `make.inc` LMie may be compiled by executing

```
make
```

and for the MPI version of LMie

```
make mpi
```

2.1.2 Visual Studio

LMie may also be built on Windows using Visual Studio and, if the Fortran 77 and/or 90 interfaces are desired, with Intel's Visual Fortran Composer XE for Windows. Supported versions of Visual Studio are 2005, 2008, and 2010. Depending on which version is being used the LMie Visual Studio solution may be loaded from one of the following solution (`.sln`) files relative to the LMie base directory:

```
msvs_2005/lmie.sln
msvs_2008/lmie.sln
msvs_2010/lmie.sln
```

2.2 Using LMie in your code

To use LMie in your own code you have to include/use the appropriate header/module file and link with the appropriate LMie library files and any required external library files.

2.2.1 C/C++

The C interface is part of the core library in the `src/` directory. To use the C interface your code must include the following header file:

```
src/lmie_interface.h
```

and must link with the following libraries:

```
src/liblmie.a
misc/liblmie_misc.a
```

or when using Visual Studio the following libraries:

```
$(SolutionDir)/$(ConfigurationName)/liblmie.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_misc.lib
```

where the variable `$(SolutionDir)` is `msvs_2005`, `msvs_2008`, or `msvs_2010` and the variable `$(ConfigurationName)` is `Debug` or `Release`.

For example, if one has C code in a file named `my_code.c`, includes the LMie C interface header file with

```
#include <lmie_interface.h>
```

and uses GCC to compile and link the code the command may look like this

```
gcc -O2 my_code.c -I$(LMIE_HOME)/src -L$(LMIE_HOME)/src -L$(LMIE_HOME)/misc \
    -llmie -llmie_misc
```

where the variable `$(LMIE_HOME)` is the location of the LMie base directory. For more information, take a look at the build details for the C interface example program `examples/example_c.c`.

2.2.2 Fortran 77

To use the Fortran 77 interface your code must include the following file:

```
interfaces/lmie_int_f77.inc
```

and must link with the following libraries:

```
src/liblmie.a
misc/liblmie_misc.a
interfaces/liblmie_interfaces.a
```

or when using Visual Studio the following libraries:

```
$(SolutionDir)/$(ConfigurationName)/liblmie.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_misc.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_interfaces.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_interfaces_f.lib
```

where the variable `$(SolutionDir)` is `msvs_2005`, `msvs_2008`, or `msvs_2010` and the variable `$(ConfigurationName)` is `Debug` or `Release`. For more information, take a look at the build details for the Fortran 77 interface example program `examples/example_f77.f`.

2.2.3 Fortran 90

To use the Fortran 90 interface your code must `USE` the `LMIE_INT_F90` module and must link with the following libraries:

```
src/liblmie.a
misc/liblmie_misc.a
interfaces/liblmie_interfaces.a
```

or when using Visual Studio the following libraries:

```
$(SolutionDir)/$(ConfigurationName)/liblmie.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_misc.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_interfaces.lib
$(SolutionDir)/$(ConfigurationName)/liblmie_interfaces_f.lib
```

where the variable `$(SolutionDir)` is `msvs_2005`, `msvs_2008`, or `msvs_2010` and the variable `$(ConfigurationName)` is `Debug` or `Release`. For more information, take a look at the build details for the Fortran 90 interface example program `examples/example_f90.f90`.

3 LMie Inputs

This section gives a detailed description of LMie's inputs independent of the details of the input interfaces. See the appropriate interface section for details on actually supplying inputs to LMie.

calc_pf
: Flag indicating whether or not to compute the elements of the scattering phase matrix.

calc_gc
: Flag indicating whether or not to compute the coefficients for the expansion of the scattering phase matrix in terms of generalized spherical functions.

calc_lc
: Flag indicating whether or not to compute the coefficients for the expansion of the scattering phase matrix in terms of Legendre polynomials.

dist_type
: The particle size distribution to use. Currently seven size distributions are supported. Distributions 2 - 6 are given here as in chapter 5 of Mishchenko et al. [2002]

1. mono:
$$n(r) = 1 \tag{3.1}$$
2. gamma:
$$n(r) = Cr^{(1-3b)/b} \exp\left(-\frac{r}{ab}\right), b \in (0, 0.5) \tag{3.2}$$
3. modified gamma:
$$n(r) = Cr^{\alpha} \exp\left(-\frac{\alpha r^{\gamma}}{\gamma r_c^{\gamma}}\right) \tag{3.3}$$
4. power law:
$$n(r) = \begin{cases} Cr^{-3} & \text{if } r_1 \leq r \leq r_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$
5. modified power law:
$$n(r) = \begin{cases} C & \text{if } 0 \leq r \leq r_1 \\ C(r/r_1)^{\alpha} & \text{if } r_1 \leq r \leq r_2 \\ 0 & \text{if } r_2 \leq r \end{cases} \tag{3.5}$$
6. log normal:
$$n(r) = Cr^{-1} \exp\left[-\frac{(\ln r - \ln r_g)^2}{2 \ln^2 \sigma_g}\right] \tag{3.6}$$

7. modified bimodal log normal:

$$n(r) = Cr^{-4}\left\{\exp\left[-\frac{(\ln r - \ln r_{g1})^2}{2\ln^2\sigma_{g1}}\right] + \gamma\exp\left[-\frac{(\ln r - \ln r_{g2})^2}{2\ln^2\sigma_{g2}}\right]\right\} \tag{3.7}$$

where C is the normalization constant, r is particle radius, r_1 and r_2 are the minimum and maximum radii in the size distribution (input parameters `r1` and `r2`), and the other size distribution parameters are input via inputs `a1 - a5`.

n_int1
The number of subintervals from 0 to r_1 for the integration of size distribution. Only used for the power law size distribution.

n_int2
The number of subintervals from r_1 to r_2 for the integration of size distribution.

n_quad
The number of quadrature points to use per subinterval for the integration of size distribution.

n_angles
The number of scattering angles at which to compute the elements of the scattering matrix. Only used if input flag `calc_pf` is set.

n_derivs
The number of derivatives to be computed.

lambda
Wavelength of incident radiation, the units of which must be consistent with the input size distribution parameters `a1 - a5`.

mr
The real part of the complex index of refraction of the particle medium.

mi
The imaginary part of the complex index of refraction of the particle medium. Must be positive.

a1 - a5
Size distribution parameters, the units of which must be consistent with the wavelength of incident radiation `lambda`. The number of parameters used and their definition depend on the size distribution selected by input `dist_type`. The unused parameters are ignored.

1. mono:

   $\texttt{a1} = r$

2. gamma:

   $\texttt{a1} = a$
   $\texttt{a2} = b$

3. modified gamma:

$\texttt{a1} = \alpha$
$\texttt{a2} = r_c$
$\texttt{a3} = \gamma$

4. power law:

$\texttt{a1} = r_{\text{eff}}$
$\texttt{a2} = v_{\text{eff}}$

5. modified power law:

$\texttt{a1} = \alpha$

6. log normal:

$\texttt{a1} = r_g$
$\texttt{a2} = (\ln \sigma_g)^2$

7. modified bimodal log normal:

$\texttt{a1} = r_{g,1}$
$\texttt{a2} = (\ln \sigma_{g,1})^2$
$\texttt{a3} = r_{g,2}$
$\texttt{a4} = (\ln \sigma_{g,2})^2$
$\texttt{a5} = \gamma$

r1, **r2**

Minimum and maximum radii (r_1 and r_2) in the size distribution. In theory, **r1** and **r2** should be equal to zero and infinity, respectively. In practice, the user should extend their range until the calculated scattering characteristics converge to within a prescribed numerical accuracy. Unused for the mono (disperse) and power law size distributions.

accuracy

The numerical accuracy at which to compute the coefficients for the expansion of the scattering matrix in terms of generalized spherical functions (flag **calc_gc**) and/or Legendre polynomials (flag **calc_lc**).

lambda_l, **mr_l**, **mi_l**, **a1_l - a5_l**, **r1_l**, **r2_l**

Linearized inputs indicated by a "**_l**" appended to the name for the associated input base value. These inputs are one dimensional arrays of size **n_derivs** where each element is associated with one of **n_derivs** derivatives. As a simple example, if we have a log normal size distribution and we are interested in derivatives with respect to m_r, m_i, r_g, and $(\ln \sigma_g)^2$, in that order, for a total of 4 derivatives, the inputs would be set as

$\texttt{lambda\_l} = \{0, 0, 0, 0\}$
$\texttt{mr\_l} = \{1, 0, 0, 0\}$
$\texttt{mi\_l} = \{0, 1, 0, 0\}$
$\texttt{a1\_l} = \{0, 0, 1, 0\}$
$\texttt{a2\_l} = \{0, 0, 0, 1\}$
$\texttt{r1\_l} = \{0, 0, 0, 0\}$
$\texttt{r2\_l} = \{0, 0, 0, 0\}$

More complex inputs, with values other than unity and where more than one linearized input is set to nonzero for a particular derivative index, are possible if one has derivatives of the

inputs with respect to parameters further up the chain and is interested in Mie results with respect to these parameters.

4 LMie Outputs

This section gives a detailed description of LMie's outputs independent of the details of the output interfaces. See the appropriate interface section for details on actually obtaining outputs from LMie.

r21, r22
: The actual minimum and maximum radii in the size distribution. Same as the inputs r1 and r2 for all but the mono (disperse) and power law size distributions.

norm
: Normalization constant C of the size distribution.

reff
: Effective radius of the size distribution:

$$r_{\text{eff}} = \frac{1}{G_{\text{avg}}} \int_{r_1}^{r_2} n(r) \pi r^2 dr, \tag{4.1}$$

where G_{avg} is defined under input gavg.

veff
: Effective variance of the size distribution:

$$v_{\text{eff}} = \frac{1}{G_{\text{avg}} r_{\text{eff}}^2} \int_{r_1}^{r_2} n(r)(r - r_{\text{eff}})^2 \pi r^2 dr, \tag{4.2}$$

where G_{avg} is defined under input gavg.

gavg
: The average projected area per particle of the size distribution:

$$G_{\text{avg}} = \int_{r_1}^{r_2} n(r) \pi r^2 dr. \tag{4.3}$$

vavg
: The average volume per particle of the size distribution:

$$V_{\text{avg}} = \int_{r_1}^{r_2} n(r) \frac{4}{3} \pi r^3 dr. \tag{4.4}$$

ravg

The average radius of the size distribution:

$$r_{\text{avg}} = \int_{r_1}^{r_2} n(r) r dr. \tag{4.5}$$

rvw

The volume-weighted average radius of the size distribution:

$$r_{\text{vwa}} = \int_{r_1}^{r_2} n(r) r \frac{4}{3} \pi r_3 dr. \tag{4.6}$$

cext

Ensemble averaged extinction cross section per particle.

csca

Ensemble averaged scattering cross section per particle.

cbak

Ensemble averaged backscattering cross section per particle.

g

Ensemble averaged asymmetry factor.

theta

Angles at which the elements of the ensemble averaged scattering phase matrix are computed as an array of size `n_angles`. Only valid if the input flag `calc_pf` is set.

pf

Elements of the ensemble averaged scattering phase matrix at the angles indicated by output `theta` as a $(6 \times$ `n_angles`$)$ array. Only valid if the input flag `calc_pf` is set.

The scattering phase matrix has the form appropriate for a macroscopically isotropic and mirror symmetric scattering medium given by

$$\mathbf{F}(\Theta) = \begin{bmatrix} a_1(\Theta) & b_1(\Theta) & 0 & 0 \\ b_1(\Theta) & a_2(\Theta) & 0 & 0 \\ 0 & 0 & a_3(\Theta) & b_2(\Theta) \\ 0 & 0 & -b_2(\Theta) & a_4(\Theta) \end{bmatrix}, \tag{4.7}$$

where $a_1(\Theta)$ is the scattering phase function. For the case of homogeneous isotropic spheres $a_1(\Theta) = a_2(\Theta)$ and $a_3(\Theta) = a_4(\Theta)$ but for compatibility reasons the output is for all six elements. The elements map to the output array `pf` (in C syntax) as

$a_1(\Theta_i) =$ `pf[0][i]`
$a_2(\Theta_i) =$ `pf[1][i]`
$a_3(\Theta_i) =$ `pf[2][i]`
$a_4(\Theta_i) =$ `pf[3][i]`
$b_1(\Theta_i) =$ `pf[4][i]`
$b_2(\Theta_i) =$ `pf[5][i]`

where

$\Theta_i =$ theta[i]

and $0 \leq$ i $<$ n_angles.

n_coef

The number of expansion coefficients (and associated derivatives) output in the gc and lc arrays, and when derivatives are to be computed, the gc_l and lc_l arrays. This is the number of coefficients required to accurately represent the scattering phase matrix $\mathbf{F}(\Theta)$ to the accuracy given as the input accuracy.

gc

Coefficients for the expansion of the ensemble averaged scattering phase matrix (equation 4.7) in terms of generalized spherical functions as a ($6 \times$ n_coef) array. Only valid if the input flag calc_gc is set.

The expansion has the form given by

$$\begin{aligned}
a_1(\Theta) &= \sum_{l=0}^{N} \beta_l P^l_{0,0}(\cos\Theta), \\
a_2(\Theta) + a_3(\Theta) &= \sum_{l=2}^{N} (\alpha_l + \zeta_l) P^l_{2,2}(\cos\Theta), \\
a_3(\Theta) - a_3(\Theta) &= \sum_{l=2}^{N} (\alpha_l - \zeta_l) P^l_{2,-2}(\cos\Theta), \\
a_4(\Theta) &= \sum_{l=0}^{N} \delta_l P^l_{0,0}(\cos\Theta), \\
b_1(\Theta) &= \sum_{l=0}^{N} \gamma_l P^l_{0,2}(\cos\Theta), \\
b_2(\Theta) &= \sum_{l=0}^{N} \varepsilon_l P^l_{0,2}(\cos\Theta),
\end{aligned}$$

where $P^l_{m,n}(\cos\Theta)$ are generalized spherical functions. It is worth noting that $a_1(\Theta)$ is equal to the so called scattering phase function $P(\cos\Theta)$ and the generalized spherical function $P^l_{0,0}(\cos\Theta)$ is equal to the Legendre polynomial $P_l(\cos\Theta)$. The expansion coefficients are the so "Greek constants" [Siewert, 1982] of the matrix

$$\mathbf{B}_l = \begin{bmatrix} \beta_l & \gamma_l & 0 & 0 \\ \gamma_l & \alpha_l & 0 & 0 \\ 0 & 0 & \zeta_l & -\varepsilon_l \\ 0 & 0 & \varepsilon_l & \delta_l \end{bmatrix}, \tag{4.8}$$

which is a common form of input for many vector radiative transfer models. The expansion coefficients map to the output array gc (in C syntax) as

$\beta_l =$ gc[0][l]
$\alpha_l =$ gc[1][l]
$\zeta_l =$ gc[2][l]
$\delta_l =$ gc[3][l]
$\gamma_l = -$gc[4][l]
$\varepsilon_l = -$gc[5][l]

where $0 \leq$ l $<$ n_coef.

lc

Coefficients for the expansion of the ensemble averaged scattering phase matrix (equation 4.7) in terms of Legendre polynomials as a ($6 \times$ n_coef) array. Only valid if the input flag calc_lc is set.

The expansion has the form given by

$$
\begin{aligned}
a_1(\Theta) &= \sum_{l=0}^{N} \beta_{1,l} P_l(\cos\Theta), \\
a_2(\Theta) &= \sum_{l=0}^{N} \beta_{2,l} P_l(\cos\Theta), \\
a_3(\Theta) &= \sum_{l=0}^{N} \beta_{3,l} P_l(\cos\Theta), \\
a_4(\Theta) &= \sum_{l=0}^{N} \beta_{4,l} P_l(\cos\Theta), \\
b_1(\Theta) &= \sum_{l=0}^{N} \beta_{5,l} P_l(\cos\Theta), \\
b_2(\Theta) &= \sum_{l=0}^{N} \beta_{6,l} P_l(\cos\Theta),
\end{aligned}
$$

where $P_l(\cos\Theta)$ are Legendre polynomials of degree l. It is worth noting that $a_1(\Theta)$ is equal to the so called scattering phase function $P(\cos\Theta)$. The expansion coefficients map to the output array lc (in C syntax) as

$\beta_{1,l} =$ lc[0][l]
$\beta_{2,l} =$ lc[1][l]
$\beta_{3,l} =$ lc[2][l]
$\beta_{4,l} =$ lc[3][l]
$\beta_{5,l} =$ lc[4][l]
$\beta_{6,l} =$ lc[5][l]

where $0 \leq$ l $<$ n_coef.

r21_l, r22_l, reff_l, veff_l, gavg_l, vavg_l, ravg_l, rvw_l, cext_l, csca_l, cbak_l, g_l

Linearized outputs indicated by a "_l" appended to the name for the associated output base value. These outputs are one dimensional arrays of size n_derivs where each element is associated with one of n_derivs derivatives.

pf_l, **gc_l**, **lc_l**

Linearized outputs indicated by a "_l" appended to the name for the associated output base value. These outputs are three dimensional arrays of size (n_derivs $\times 6 \times$ n_angles) for pf_l and (n_derivs $\times 6 \times$ n_coef) for gc_l and lc_l, where each subarray of the first dimension is associated with one of n_derivs derivatives.

5 LMie C Interface

The LMie C interface is made available by including `src/lmie_interface.h` in your code. It consists of some enumeration constants, input and output structures as typedefs, and functions for managing memory used in the structures and to call the actual Mie scattering algorithm.

In a typical usage the steps taken to use LMie will be:

1. Allocate memory used by the `lmie_in_data` input structure with **`lmie_in_alloc()`**.
2. If derivatives are to be computed, zero out all the linearized inputs in the `lmie_in_data` input structure with **`lmie_in_zero_derivs()`**.
3. Set the input values in the `lmie_in_data` input structure including any nonzero linearized values when derivatives are to be computed
4. Call **`lmie_solution()`** to calculate the Mie scattering results.
5. Read/use the output values from the `lmie_out_data` output structure.
6. Free memory allocated to the `lmie_out_data` output structure with **`lmie_out_free()`**.
7. If another set of results is required and the number of derivatives has not changed loop back to step 2.
8. Free memory allocated to the `lmie_in_data` input structure with **`lmie_in_free()`**

See `examples/example_c.c` for an actual example of LMie usage in C.

5.1 Size distributions

Size distributions are indicated by the following enumeration constants:

SIZE_DIST_MONO
SIZE_DIST_GAMMA
SIZE_DIST_MODIFIED_GAMMA
SIZE_DIST_POWER_LAW
SIZE_DIST_MODIFIED_POWER_LAW
SIZE_DIST_LOG_NORMAL
SIZE_DIST_MODIFIED_BIMODAL_LOG_NORMAL

5.2 Input and output structures

The LMie input structure typedef is:

```
typedef struct {
   int calc_gc;
   int calc_lc;
   int calc_pf;
   enum size_dist_type dist_type;
   int n_int1;
   int n_int2;
   int n_quad;
   int n_angles;
   int n_derivs;
   double lambda;
   double mr;
   double mi;
   double a1;
   double a2;
   double a3;
   double a4;
   double a5;
   double r1;
   double r2;
   double *lambda_l;
   double *mr_l;
   double *mi_l;
   double *a1_l;
   double *a2_l;
   double *a3_l;
   double *a4_l;
   double *a5_l;
   double *r1_l;
   double *r2_l;
   double accuracy;
} lmie_in_data;
```

The LMie output structure typedef is:

```
typedef struct {
   int n_coef;
   double r1;
   double r2;
   double norm;
   double reff;
   double veff;
```

```
    double gavg;
    double vavg;
    double ravg;
    double rvw;
    double cext;
    double csca;
    double cbak;
    double g;
    double **gc;
    double **lc;
    double *theta;
    double **pf;
    double *r1_l;
    double *r2_l;
    double *norm_l;
    double *reff_l;
    double *veff_l;
    double *gavg_l;
    double *vavg_l;
    double *ravg_l;
    double *rvw_l;
    double *cext_l;
    double *csca_l;
    double *cbak_l;
    double *g_l;
    double ***gc_l;
    double ***lc_l;
    double ***pf_l;
} lmie_in_data;
```

5.3 Function interface

`int` **`lmie_in_alloc`**`(`*`lmie_in_data`* ***`in`***`,` *`int`* ***`n_derivs`***`)`

Description:

Allocate memory used by an `lmie_in_data` input structure. This function should be called before populating an `lmie_in_data` input structure before the first call to **`lmie_solution()`**. After calling this function all pointers within the `lmie_in_data` input structure may be used as valid arrays over subsequent calls to **`lmie_solution()`** as long as ***`n_derivs`*** does not change. Once use of the `lmie_in_data` input structure is finished **`lmie_in_free()`** must be called to free the allocated memory.

Arguments:

`in`	an `lmie_in_data` input structure for which to allocate memory
`n_derivs`	the number of derivatives for which to allocate memory

Return value:

Zero with successful completion or otherwise on error.

`void` **`lmie_in_free`**(*`lmie_in_data *`**`in`**`, int `**`flag`*)

Description:
Free memory allocated to an `lmie_in_data` input structure by **`lmie_in_alloc()`**. This function should be called after use of an `lmie_in_data` input structure is finished.

Arguments:
`in` an `lmie_in_data` input structure for which to free memory
`flag` zero if the number of derivatives is zero and nonzero otherwise

Return value:
None.

`void` **`lmie_out_free`**(*`lmie_out_data *`**`out`**`, int `**`flag`*)

Description:
Free memory allocated to an `lmie_out_data` output structure by **`lmie_solution()`**. This function should be called after use of the output data within the `lmie_out_data` output structure is finished and before another call to **`lmie_solution()`** otherwise the allocated memory from the previous call will be leaked.

Arguments:
`out` an `lmie_out_data` output structure for which to free memory
`flag` zero if the number of derivatives is zero and nonzero otherwise

Return value:
None.

`void` **`lmie_in_zero_derivs`**(*`lmie_in_data *`**`in`**`, int `**`n_derivs`*)

Description:
Sets all the linearized inputs (derivatives) in an `lmie_in_data` input structure to zero. This function is not necessary and is provided as convenience since in many cases most of the linearized inputs are set to zero. This function must be called after a call to **`lmie_in_alloc()`**.

Arguments:
`in` an `lmie_in_data` input structure for which to zero the linearized inputs

Return value:
None.

`int` **`lmie_solution`**(*`lmie_in_data *`**`in`**`, lmie_out_data *`**`out`**`, int `**`alloc_out`**`, int `**`verbose`**`, int `**`n_threads`**`, int `**`use_mpi`*)

Description:
Run the actual Mie scattering algorithm using the inputs in the supplied input structure and put the outputs in the supplied output structure.

Arguments:

in	the `lmie_in_data` input structure
out	the `lmie_out_data` output structure
alloc_out	Flag indicating whether or not memory should be allocated for the `lmie_out_data` output structure. This should only be set to "false" if enough memory has already been allocated for the `lmie_out_data` output structure, either by the user or from a subsequent call to **`lmie_solution()`**. The amount of memory used is dependent on the input value `r2` so that an initial call to **`lmie_solution()`** using a maximum value for `r2` will allocate enough memory for subsequent calls with a lesser or equal value of `r2`.
n_threads	Number of threads to use. For minimum run times the number of threads should be set to the number of cores available on the machine being used.
use_mpi	A flag indicating whether or not to use MPI. This requires the execution of two or more separate calling processes using the appropriate MPI execution program. MPI parallelization is above the threaded parallelization in which case ***n_threads*** indicates the number of threads to use *per* MPI process.
verbose	A flag indicating whether or not to print information useful for debugging. Currently only information useful for debugging calls using MPI is printed.

Return value:

Zero with successful completion or otherwise on error.

5.4 Example C program using LMie

An example program using the C interface is at

```
examples/example_c.c
```

and when the LMie code is compiled properly the C example program will be compiled as

```
examples/example_c
```

6 LMie Fortran 77 Interface

6.1 Size distributions

6.2 Function interface

6.3 Example Fortran 77 program using LMie

An example program using the Fortran 77 interface is at

```
examples/example_f77.f
```

and when the LMie code is compiled properly the Fortran 77 example program will be compiled as

```
examples/example_f77
```

7 LMie Fortran 90 Interface

7.1 Size distributions

7.2 Input and output structures

7.3 Function interface

7.4 Example Fortran 90 program using LMie

An example program using the Fortran 90 interface is at

```
examples/example_f90.f90
```

and when the LMie code is compiled properly the Fortran 90 example program will be compiled as

```
examples/example_f90
```

8 LMie command line program: calllmie

`calllmie` is a command line program that takes inputs from the command line, runs LMie, and outputs results.

The path to `calllmie` is

```
utils/calllmie
```

or when using Visual Studio

```
$(SolutionDir)/$(ConfigurationName)/calllmie.exe
```

where the variable `$(SolutionDir)` is `msvs_2005`, `msvs_2008`, or `msvs_2010` and the variable `$(ConfigurationName)` is `Debug` or `Release`.

An example run of `calllmie` is contained in the following text file:

```
examples/calllmie.txt
```

Please see the text file for more details about the example run. The `calllmie` input format for the command line and output format are outlined in the next two sections.

8.1 Calllmie input format

8.1.1 General flags

`-help`
Print a message linking the user to this documentation and exit.

`-n_threads <n_threads>`
The number of threads to use. This is an optional flag with a default value of 1.

`-verbose`
Print information useful for debugging. Currently only information useful for debugging runs using MPI is printed.

`-version`
Print version information and exit.

8.1.2 Input flags

-accuracy <accuracy>
Set the accuracy.

-derivs <n_derivs>
Turn on derivative output and set the number of derivatives to be calculated n_derivs.

-dist <NAME> <a1> ... <a5> [r1] [r2]
Set the size distribution name and the required input parameters a1 - a5 and, if required, r1 and r2. The size distribution name NAME is the name of one of the supported size distributions with underscores in place for spaces. For example, if one had a log normal size distribution with a mean radius of 0.39μm and a standard deviation of 2.0μm ranging from 0.005 to 50μm the flag and required parameters would be given as

```
-dist log_normal 0.39 0.48 0.005 50.0
```

where $(\ln 1.0)^2 = 0.48$.

-dist_l <a1_l1,a2_l2,...> ... <a5_l1,a5_l2,...> [r1_l1,r1_l2,...] [r2_l1,r2_l2,...]
Set linearized parameters a1_l - a5_l and, if required r1_l and r2_l, for the size distribution given by the **-dist** flag. Flag **-dist** must be given on the command line before **-dist_l**. Each argument is a comma-separated list (with no spaces) of values for each derivative. For example, if one had a log normal size distribution and derivatives with respect to r_g, $(\ln \sigma_g)^2$, m_r, and m_i, in that order, the flag and required arguments would be given as

```
-dist log_normal 1.0,0.0,0.0,0.0 0.0,1.0,0.0,0.0 \
                 0.0,0.0,0.0,0.0 0.0,0.0,0.0,0.0
```

where each of the 4 lists have a length of 4. This flag is only valid when **-derivs** is given and is optional with default values all equal to zero.

-lambda <lambda>
Set the wavelength of incident radiation lambda.

-lambda_l <lambda_l_1,lambda_l_2,...>
Set the linearized wavelength of incident radiation lambda_l. This flag is only valid when **-derivs** is given and is optional with default values all equal to zero.

-m <mr> <mi>
Set the real (**mr**) and imaginary (**mi**) parts of the complex index of refraction of the particle medium.

-m_l <m_r_l1,m_r_l2,...> <m_i_l1,m_i_l2,...>
Set the linearized real (**mr**) and imaginary (**mi**) parts of the complex index of refraction of the particle medium. Each argument is a comma-separated list (with no spaces) of values for each derivative. For example, if one had a log normal size distribution and derivatives with respect to r_g, $(\ln \sigma_g)^2$, m_r, and m_i, in that order, the flag and required arguments would be given as

```
-m_l 0.0,0.0,1.0,0.0 0.0,0.0,0.0,1.0
```

where each of the 2 lists have a length of 4. This flag is only valid when **-derivs** is given and is optional with default values all equal to zero.

-n_int1 <n_int1>
Set the number of subintervals from 0 to r_1 (**n_int1**) for the integration of size distribution. Only used for the power law size distribution.

-n_int2 <n_int2>
Set the number of subintervals from r_1 to r_2 (**n_int2**) for the integration of size distribution.

-n_quad <n_quad>
Set the number of quadrature points to use per subinterval (**n_quad**) for the integration of size distribution.

-n_angles <n_angles>
Set the number of scattering angles (**n_angles**) at which to compute the elements of the scattering matrix. Only used if output flag **-pf** is given.

8.1.3 Output flags

-pf,-gc,-lc <FILE> [FILE_L1,FILE_L2,...]
Write the scalar outputs along with the elements of the scattering phase matrix (**-pf**) or the coefficients for the expansion of the scattering phase matrix in terms of generalized spherical functions (**-gc**) or Legendre polynomials (**-lc**). The first argument FILE is the output file for the full quantities while the second argument is a comma-separated list (with no spaces) of output files (FILE_L1, FILE_L2, ...) for each derivative and is only required if the option **-derivs** is given. For example, if one had a log normal size distribution and derivatives with respect to r_g, $(\ln \sigma_g)^2$, m_r, and m_i, in that order, the flag and required arguments for output with the coefficients for the expansion of the scattering phase matrix in terms of generalized spherical functions could be given as

```
-gc output.gc "output_l_r_g.gc,output_l_ln_r_g_2.gc, \
               output_l_m_r.gc,output_l_ln_m_i.gc"
```

If the '-' symbol is given in place of a file name the associated output will be sent to standard output. Multiple types of output may be output by giving more than one of **-pf**, **-gc**, or **-lc** but the scalar outputs will be identical in each case.

8.2 Calllmie output format

The calllmie output format is self-explanatory.

Bibliography

Gustav Mie. Beiträge zur optik trüber medien, speziell kolloidaler metallösungen. *Annalen der Physik*, 25(4):377–445, 1908.

Michael I. Mishchenko, Larry D. Travis, and Andrew A. Lacis. *Scattering, Absorption, and Emission of Light by Small Particles*. Cambridge University Press, Cambridge, 2002. http://www.giss.nasa.gov/~crmim/publications/book_2.pdf.

C. E. Siewert. On the phase matrix basic to scattering of polarized light. *Astronomy and Astrophysics*, 109:195–200, 1982.