MSTM

A multiple sphere T-matrix FORTRAN code for use on parallel computer clusters

Version 4.0

D. W. Mackowski
Department of Mechanical Engineering
Auburn University, Auburn, AL 36849, USA

January 30, 2022

About this document

This is the instruction manual for the MSTM Fortran–90 code. The code was originally released in January 2011. The current version of the manual (the one you are reading now) corresponds to version 4.0, and was released on January 30, 2022. This version is considerably different than previous ones, and those familiar with the use of the old codes are warned that their old input files will need some modification to work with the new code.

All queries regarding the code should be addressed to the author at mstm@auburn.edu.

Contents

1 Purpose

MSTM is a FORTRAN-90 code for calculating monochromatic electromagnetic scattering properties of multiple spherical domains. The algorithm applies the multiple sphere T matrix method, and the results can be considered exact to the truncation error of the vector spherical wave function (VSWF) expansions used to represent the fields. The code can

- calculate the cross sections, asymmetry parameters, and far–field scattering matrix elements for both fixed and random orientations with respect to the incident wave,
- be applied to arbitrary configurations of spheres located internally or externally to other spheres, the only restriction being that the surfaces of the spheres do not overlap,
- incorporate optically active refractive indices for all media, including the external medium,
- model both plane wave and Gaussian profile incident beams,
- include the presence of multiple plane boundaries that separate regions having different refractive index,
- have the spheres arranged in 2-D periodic lattice configurations in the lateral plane, and
- generate maps of the electric field distributions along any arbitrary plane and including points both within and external to the spheres.

The code incorporates message passing interface (MPI) commands to implement execution on distributed memory, multiple processor compute clusters.

The mathematical formulation is presented in [1, 2, 3, 4, 5, 6].

2 System configuration

All quantities used in the code are implicitly dimensionless. Lengths are scaled using the free space wavenumber $\mathrm{k}_0 = 2\pi/\lambda_0$, with λ_0 denoting the free space wavelength of the exciting incident wave, electric field amplitude is relative to the amplitude of the exciting wave, and refractive indices are relative to $\sqrt{\epsilon_0}$, where ϵ_0 is the permittivity of free space. A time harmonic factor of $\exp(-\mathrm{i}\omega t)$ is adopted.

The system consists of N_S spherical surfaces, with each characterized by a dimensionless radius a^i, a left–right pair of complex refractive indices $\mathbf{m}^i = (\mathsf{m}_L^i,\ \mathsf{m}_R^i)$ for the medium contacting the *inside* surface of the sphere, and a dimensionless position vector $\mathbf{r}^i = (x^i, y^i, z^i)$ that denotes the origin of the spherical surface relative to a common origin, for $i = 1, 2, \ldots N_S$. There is no restriction placed on the location of the sphere origins, with the sole exception that the surfaces of any two spheres cannot overlap. In other words, there can only be one value of $\mathbf{m}_i$ for each surface i.

The system within which the spheres are embedded consists of N_B+1 regions (or layers) separated by N_B parallel plane boundaries with surface normals in the z direction. The first boundary is fixed at $z = Z_1 = 0$, and the remainder located at $z = Z_2,\ Z_3, \ldots Z_{N_B}$, with all $Z_B \geq 0$. The media below $z = 0$ is denoted as layer $\ell = 0$, that between $z = 0$ and Z_2 as layer $\ell = 1$, and so on. Layers 0 and N_B are half spaces extending to - and + ∞. The material within each layer is isotropic and characterized by a complex refractive index $\mathsf{m}_\ell, \ell = 0,\ 1 \ldots N_B$. Spherical surfaces that have exterior surfaces in contact with the external layers are

referred to as exterior spheres, and $N_{S,E}$ denotes the number of exterior spheres: $1 \leq N_{S,E} \leq N_S$. Each of the exterior spheres will reside entirely within a single layer: they cannot cross the layer boundaries.

The system of N_S spheres form a unit cell, and this unit cell may be repeated in the x, y plane to $\pm\infty$, with period W_x, W_y, to form a 2–D periodic lattice structure. For such conditions it is implied that the sphere system can be repeated periodically, with the given period W_x, W_y, without overlap of the spheres. For example, a single sphere system must have W_x, W_y both greater that $2a$.

The system can be excited by either a plane wave or a Gaussian beam. The incident direction of the beam can specified, via a polar β and azimuth α angle, or – for $N_B = 0$ – the code can perform a calculation of the random orientation properties of the sphere system.

3 Structure, compilation, and execution of the code

The code is organized into the following four components:

`mstm.f90:` Contains the main body of the code.

`mpidefs-parallel.f90:` A module which defines the MPI commands appearing in the `mstm-modules.f90` and `mstm-main.f90` code blocks for use on multiprocessor platforms.

`mpidefs-serial.f90:` A module which defines MPI commands for use on single processor (serial) platforms.

`mstm-intrinsics.f90:` Compiler–specific (non–standard Fortran) functions for command–line argument retrieval and system time operations. The users must modify this module to suit their specific compiler.

The actual file names included in the distribution may also include the version identifier, which will appear as `mstm-v4.0.f90`, etc. The version identifier will be omitted in the following discussion.

Compilation of the code using the GNU *gfortran* on a MS-Windows machine in serial mode would involve

```
gfortran  -o mstm.exe mstm-intrinsics.f90 mpidefs-serial.f90 mstm.f90
```

This places the executable in the file `mstm.exe`. Compilation using the MPICH2 package for execution on a parallel machine would use

```
mpif90 -I/opt/mpich2-1.2.1p1/include -g -o mstm.out
        mstm-intrinsics.f90 mpidefs-parallel.f90
        mstm.f90
```

and would put the executable in `mstm.out`.

Other compilers follow the same basic plan. It is important to compile the module files in the order they are given. And please remember that the `mstm-intrinsics.f90` must be modified to match the command–line recognition and retrieval intrinsic functions of the compiler. The distribution has the intrinsics set up for *gfortran*.

Properties of the sphere cluster and calculation variables and options are passed to the code during execution by use of an input file. An input file can be designated by a command line argument; on a serial machine, with the executable named `mstm.exe` and an input file named `mstm-01.inp`, the command to execute the code would appear as

```
mstm mstm-01.inp
```

On a parallel machine the execution command will appear in the shell script used to submit the job. On my windows machine I use ms-mpi with gfortran; installation and creation of the MPI library files is described in https://abhila.sh/writing/3/mpi_instructions.html. You might need to disable some of the gfortran error detection settings regarding subroutine arguments; such was my experience. Launching the code to run on 8 cores would involve

```
mpiexec -n 8 mstm mstm-01.inp
```

For optimum performance in parallel operation: the code should be run on some multiple of 4 processors, i.e., 4,8,12,...

The input file must be in the same directory as the executable. The default input file is mstm.inp; this file must be present if no command line argument is given.

4 Quick input file guide

For those of you who want to cut to the chase, this section offers the basics of input file structure and a selection of input file examples for the most common types of calculations.

4.1 Input file structure

The input file primarily consists of paired lines; the first line of a pair representing a parameter ID, and the second representing the value or option for that parameter. Order of the option pairs in important only for situations in which conflicting options are chosen; in this case, the last option pair appearing in the input file will be the one in effect. If a pair corresponding to a particular parameter is not present, the code will use the default value.

An example of an input file, showing the first few input parameters, is shown below.

```
number_spheres
100
sphere_position_file
ran100.pos
output_file
test.dat
length_scale_factor
2.d0
ref_index_scale_factor
(1.6d0,0.01d0)
mie_epsilon
1.d-3
random_orientation
.true.
end_of_options
```

Note that the parameter ID, i.e., `number_spheres` or `output_file`, must appear as written in the manual. Values appearing following the parameter id can be integers, floating point, complex floating point using the convention (real, imag), i.e., (1.6,0.01), logical using `t` or `f` or `.true.` or `.false.`, or character (no enclosing quotes).

The MSTM code has many input file options and parameters. Most of you will never need 90% of them. A comprehensive list of the options is given in Sec. (5). The examples given below deal with the most common calculation scenarios. You should, however, first read Sec. (5.1) for a description of how the sphere positions and properties are specified.

4.2 Calculation of random orientation properties of sphere clusters

Random orientation calculations only make sense for systems without plane boundaries and without periodicity. The following calculates the random orientation properties of an aggregate of 100 spheres using the analytical T matrix orientation averaging algorithm [1]. The positions are given in `frac_agg.pos` in the 3-column format, which assumes unit sphere radii. The calculations set the dimensionless sphere radius to 0.5 and the refractive index to 1.6 +0.1i.

```
output_file
example-1.dat
print_sphere_data
f
number_spheres
100
sphere_data_input_file
frac_agg.pos
random_orientation
t
length_scale_factor
0.5d0
ref_index_scale_factor
(1.6d0,.1d0)
calculate_scattering_matrix
t
scattering_map_increment
1.d0
end_of_options
```

The option `scattering_map_increment` sets the angular intervals at which the scattering matrix is printed, in this case 1 degree (which is the default).

Sometimes it works faster to perform the orientation averaging via a monte carlo integration over incident directions. This is accomplished by replacing

```
output_file
example-1.dat
random_orientation
t
```

with

```
output_file
example-1a.dat
incidence_average
t
number_incident_directions
100
```

The output will now correspond to the average over `number_incident_directions` randomly–sampled incident directions; adjust this number accordingly based on the convergence of the averages.

4.3 Fixed orientation calculations

The following sets up a calculation for a sphere–within–a–sphere system exposed to an incident plane wave

```
output_file
example-2.dat
number_spheres
2
sphere_data
0.d0,0.d0,0.d0,20.d0,(1.54d0,0.d0)
0.d0,0.d0,19.d0,1.d0,(.15d0,3.d0)
end_of_sphere_data
print_sphere_data
t
incident_beta_deg
30.d0
incident_alpha_deg
0.d0
length_scale_factor
1.d0
calculate_scattering_matrix
t
incident_frame
t
scattering_map_model
0
scattering_map_increment
1.d0
end_of_options
```

Note that the inner sphere (with properties characteristic of gold) is positioned against the inside surface of the outer sphere (glass). `incident_beta_deg` and `incident_alpha_deg` are the incident polar and azimuth angles: for this case incidence is in the z direction.

The logical option `incident_frame`, when true, makes the scattering matrix correspond to the usual scattering angle definition, so that the scattering matrix at $\theta = 0$ corresponds to the incident direction, and

$\theta = \beta,\ \phi = 180^\circ$ would point in the z direction in the sphere coordinate system. Making this option false would define the scattering matrix with respect to the sphere coordinate system, so that $\theta = 0$ would point in the z direction, and $\theta = \beta,\ \phi = \alpha$ would point in the incident direction.

The scattering matrix for a fixed incidence will, in general, be a function of the polar θ and azimuth ϕ angles. The option `scattering_map_model=0` prints the scattering matrix at discrete values of θ over a circle. The plane of the circle (which sets ϕ) contains the z axis and the incident direction (when both are the same, the $x - z$ plane is used). Setting `scattering_map_model=1` will print the full 2-D scattering matrix as a pair of 2-D arrays in $k_x = \sin\theta\cos\phi,\ k_y = \sin\theta\sin\phi$, for forward and backwards directions ($\theta <, > 90$). Both k_x and k_y run from -1 to 1, and the number of points in each direction is set by the integer parameter `scattering_map_dimension`.

4.4 Plane boundaries

A two–component film (or layer) is now added to the previous system.

```
output_file
example-3.dat
number_spheres
2
sphere_data
0.d0,0.d0,-20.d0,20.d0,(1.54d0,0.d0)
0.d0,0.d0,-1.d0,1.d0,(.15d0,3.d0)
end_of_sphere_data
number_plane_boundaries
3
layer_thickness
1.d0,10.d0
layer_ref_index
(1.d0,0.d0),(1.8,0.d0),(1.54,0.d0),(1.d0,0.d0)
print_sphere_data
t
incident_beta_deg
30.d0
incident_alpha_deg
0.d0
incident_frame
f
length_scale_factor
1.d0
calculate_scattering_matrix
t
scattering_map_model
0
scattering_map_increment
1.d0
end_of_options
```

Observe that the first plane boundary is located, by definition, at $z = 0$, and the z positions of the spheres have to accommodate this convention: the large sphere, with radius 20 and z position -20, is touching the lower surface of the first boundary. No part of a sphere may cross a plane boundary; it is up to the user to get this correct. The option `incident_frame` is set to `.false.`, indicating that the scattering matrix polar angle θ will be relative to the z axis of the system; this is more appropriate as the plane boundary imposes a real (as opposed to virtual) reference frame on the system.

The following lines would model a unit radius bubble centered in a glass layer of thickness 2:

```
number_spheres
1
sphere_data
0.d0,0.d0,1.d0,1.d0,(1.d0,0.d0)
end_of_sphere_data
number_plane_boundaries
2
layer_thickness
2.d0
layer_ref_index
(1.d0,0.d0),(1.54,0.d0),(1.d0,0.d0)
```

Note that the "sphere" has a unit refractive index, i.e. that of air. It is enclosed in the glass (ref. index =1.54) layer.

4.5 Periodic systems

Periodic systems are completely compatible with plane boundary systems. A 2-D periodic array of the bubbles-in-glass would be specified by

```
number_spheres
1
sphere_data
0.d0,0.d0,1.d0,1.d0,(1.d0,0.d0)
end_of_sphere_data
number_plane_boundaries
2
layer_thickness
2.d0
layer_ref_index
(1.d0,0.d0),(1.54,0.d0),(1.d0,0.d0)
periodic_lattice
t
cell_width
3.d0,3.d0
```

The minimum cell width for this case would be 2, i.e., twice the sphere radius.

A close–packed plane layer of glass spheres, in contact with a glass half–space, would be specified by

```
number_spheres
2
sphere_data
-1.d0,0.d0,-1.d0,1.d0,(1.54d0,0.d0)
0.d0,1.73205d0,-1.d0,1.d0,(1.54d0,0.d0)
end_of_sphere_data
number_plane_boundaries
1
layer_ref_index
(1.d0,0.d0),(1.54,0.d0)
periodic_lattice
t
cell_width
2.d0,3.4641d0
```

Note that the y position of the 2nd sphere is $\sqrt{3}$, and the unit cell has dimensions (2,2$\sqrt{3}$).

The output file for the periodic system will include the reflectance, absorptance, and transmittance of the lattice, with all three adding up to unity. The scattering matrix for periodic systems, when calculated, will be listed at the discrete reciprocal lattice directions corresponding to the lattice cell width and the incident plane wave direction.

4.6 Multiple and looped calculations using a single input file

Results for different incident directions can be obtained simply by running a new job with the changed input file. Alternatively, the following commands can replace `end_of_options` in the above:

```
new_run
incident_beta_deg
40.d0
length_scale_factor
2.d0
end_of_options
```

This runs the same job over again, with all parameters the same except incident beta of 40 degrees and a length scale factor of 2. You can add as many `new_run` statements as you like, and output is appended to the output file unless a new `output_file` is listed the following the `new_run`. The last statement in the input file must always be `end_of_options`/

You can also make the calculations perform in loops. For example:

```
output_file
example-5.dat
number_spheres
2
sphere_data
0.d0,0.d0,0.d0,20.d0,(1.54d0,0.d0)
0.d0,0.d0,19.d0,1.d0,(.15d0,3.d0)
end_of_sphere_data
```

```
print_sphere_data
t
incident_alpha_deg
0.d0
length_scale_factor
1.d0
calculate_scattering_matrix
f
loop_variable
incident_beta_deg
0.d0,180.d0,2.d0
end_of_options
```

The option `loop_variable` makes the parameter following it (`incident_beta_deg`) run from a starting value (0.d0) to an ending value (180.d0) in a specified increment (2.d0). Each calculation result would be appended to the output file. These 3–line commands can be stacked (up to 3 levels) to make nested loops: the first and last `loop_variable` specifiers will correspond to the the outermost and innermost loops.

4.7 Using the accelerated solution algorithm for large–scale clusters

The code can implement a FFT–based discrete Fourier convolution procedure for calculation of the exciting field at each external sphere. Under ideal circumstances, this will change the order–N_S^2 operation count for exciting field calculation to $N_S \ln N_S$ scaling, and can accelerate the iterative solution times by a factor of 100 or more. This option will only work for 1) no plane boundaries, and 2) non–periodic. It is best for clusters of spheres that have relatively uniform sphere concentration within the volume in which they are contained. The following input file performs two calculation runs on a configuration of 1000 monodisperse spheres randomly distributed in a cylindrical volume of radius=thickness =20 sphere radii: the first run uses the fft option, and the second turns if off (reverting to the default, conventional solution method). This example uses all default options for the fft convolution algorithm; consult Sec. (5.3) for a listing of parameters for controlling the algorithm.

```
output_file
example-6.dat
number_spheres
1000
sphere_data_input_file
random_in_cylinder_1000.pos
print_sphere_data
f
random_orientation
f
incident_beta_deg
0.d0
incident_alpha_deg
0.d0
length_scale_factor
1.d0
```

```
ref_index_scale_factor
(1.6d0,.0123d0)
calculate_scattering_matrix
f
fft_translation_option
t
new_run
fft_translation_option
f
end_of_options
```

5 Comprehensive input file options

5.1 Options related to system specification

`number_spheres`: N_S, the number of spheres in the cluster.

`sphere_data_input_file`: File name containing the sphere position, radius, and refractive index data. The position file should have at least N_S lines, and the code will only read the first N_S lines of the file. Each line must have either 3, 4, 5, or 6 columns with delimiters of a comma, one or more spaces, or a tab. The first four columns correspond to the X, Y, Z positions and the radius (note that the order has been switched from the previous version) of the i^{th} sphere in the list. Units are arbitrary yet must be consistent for radius and position. If 3 columns are present the radii of all spheres are set equal to `length_scale_factor`. If 4 columns are present, the refractive index of all spheres is equal to `ref_index_scale_factor`. A single complex number in the 5th column specifies the isotropic refractive index of the particular sphere, and complex number in the 5th and 6th columns indicate an optically active material and correspond to the left and right complex refractive indices of the material.

`length_scale_factor`: The radii and positions obtained from the position file are multiplied by this factor, so that the size parameter of the i^{th} sphere is the scale factor times the radius appearing in the position file. If your position file contains dimensional sphere radii and positions (say in μm), then the scale factor will be $2\pi/\lambda$, where λ is the vacuum wavelength in μm. Default is 1. This number will also scale the positions of the plane boundaries and periodic cell widths, if present.

`ref_index_scale_factor`: Complex number, multiplies the sphere refractive index value from the position file. If refractive index values are explicitly given in the position file, then set this parameter to 1. If refractive index values do not appear in the position file, then the scale factor becomes the refractive index value for all spheres (1.,0.).

`x_shift`, `y_shift`, `z_shift`: Shift factors for sphere positions: see below.

`shifted_sphere`: Integer, specifies which sphere will have x position shifted by `x_shift`*`length_scale_factor`, and likewise for y and z positions. If `shifted_sphere=0` all spheres are shifted. These options are typically used with `loop_variable` = `x_shift` to run looped calculations over a changing sphere position.

`number_plane_boundaries`: The number of plane boundaries N_B (0). The maximum N_B allowed by the code is 10. This number can be increased by changing a parameter statement in the `surface_subroutines` module in `mstm-modules.f90`. The code is primarily designed and tested for $N_B \leq 2 \sim 3$.

`layer_thickness`: A list of $N_B - 1$ numbers, all on the same line, giving the thickness of the 1st, 2nd,... layers. These number are multiplied by `length_scale_factor` to give the actual thicknesses used in the calculations. A single boundary ($N_B = 1$) implies two infinitely-thick half spaces.

`layer_ref_index`: A list of N_B+1 complex numbers giving the refractive index values of layers 0, 1,... N_B. Not scaled by the refractive index scaling factor. Default is (1.,0.).

`periodic_lattice`: Logical: whether or not the sphere system is 2–D periodic in the $x-y$ plane. (`.false.`).

`cell_width`: List of one or two numbers (on the same line), giving the x and y periods for periodic systems, i.e., the unit cell dimensions. When only one number appears the cell is assumed square. It is multiplied by `length_scale_factor` to give the actual cell widths used in the calculations. The dimensions of the unit cell must be sufficient so that overlap of the spheres does not occur at the periodically–repeated cell boundaries.

`gaussian_beam_constant`: Dimensionless inverse width C_B, at the focal point, of an incident Gaussian profile beam. Setting $C_B = 0$ selects plane wave incidence. The localized approximation used to represent the Gaussian beam is accurate for $C_B \leq 0.2$. Default is the plane wave condition (=0.0). This number is not scaled by the length scaled factor. Note that Gaussian beam options apply to both fixed orientation and random orientation calculations. Not compatible with periodic lattice model.

`gaussian_beam_focal_point`: X, Y, and Z coordinates of the focal point of the Gaussian beam, relative to the origin and scaling in the sphere position file (i.e., before `length_scale_factor` has been applied). $(0.0, 0.0, 0.0)$.

Sphere data can be imbedded within the input file by use of the `sphere_data` and `end_of_sphere_data` commands. For example,

```
number_spheres
2
sphere_data
0.d0,0.d0,0.d0,1.d0,(1.5d0,0.d0)
0.d0,0.d0,2.d0,1.d0,(1.5d0,0.d0)
end_of_sphere_data
```

sets up a pair of contacting spheres aligned on the z axis.

5.2 Options related to numerical solution

`mie_epsilon`: Convergence criterion for determining the truncation degree for the wave function expansions for each sphere (10^{-6}). Setting `mie_epsilon` to a negative integer $-L$ forces all sphere expansions to be truncated at the Lth degree.

`translation_epsilon`: Convergence criterion for estimating the maximum order of the T matrix for the cluster (10^{-6}).

`solution_epsilon`: Error criterion for the biconjugate gradient iterative solver (10^{-6}).

`max_iterations`: The maximum number of iterations attempted to reach a solution. The code will send a message if the maximum number of iterations is exceeded (2000).

`translation_epsilon`: Error criterion for determining truncation degree when expanding fields about a common origin; mostly relevant for T matrix analytical orientation averaging (10^{-6}).

`max_t_matrix_order`: The maximum truncation degree of the T matrix. This is needed to mostly to keep the code from crashing your computer due to memory allocation limits. Some of the algorithms can also run into overflow/underflow issues when the truncation degrees exceed 120 or so; it depends on your compiler (120).

`t_matrix_epsilon`: Convergence criterion for T matrix solution. Calculation is terminated when relative change in extinction cross section from one degree to the next is less than `t_matrix_epsilon` (10^{-6}).

One should always check to see if the properties calculated by the code are altered by smaller epsilon values.

5.3 Options for the accelerated sphere interaction algorithm

For single-medium ($N_B = 0$), non–periodic systems, the code can use a FFT–based algorithm to compute the secondary exciting field at each sphere [6]. This algorithm can shorten the solution time by a factor of 100 or greater when N_S is large. Options are

`fft_translation_option`: Logical, enables the accelerated algorithm (`.false.`).

`min_fft_nsphere`: Integer, the minimum value of N_S at which the accelerated algorithm will turn on when `fft_translation_option=.true.` (200).

`cell_volume_fraction`: This determines the grid size d of the cubic lattice used in the algorithm. The formula is

$$d = \left(\frac{V_S}{c N_{S,ext}} \right)^{1/3}$$

where V_S is the dimensionless total volume of the external spheres and $c =$`cell_volume_fraction`. The optimum value is usually where the total number of nodes is close to the total number of external spheres $N_{S,ext}$. Setting `cell_volume_fraction=0` will automate the setting of d using an estimate of the sphere volume fraction (0).

`node_order`: Integer, the truncation degree for the node–based expansions of the scattered field. Set `node_order` $= -L$, where L is a positive integer, will set the node order to ceiling(d) + L. (-1).

5.4 Options related to output files

`output_file`: File name for file to which final calculation results are written (`test.dat`).

`append_output_file`: Logical. If `output_file` exists, then if `append_output_file` is false the file is overwritten, and if true the output is appended to the file. If `output_file` does not exist a new file is created for either case (`.false.`).

`run_print_file`: File name for file to which intermediate output results are written. If blank, results are written to standard output (the screen). Default is blank.

`print_sphere_data`: Logical, when true, individual sphere properties (both input and calculated) are written to the output file; false: only the total properties of the target are output. `.true.`.

5.5 Random orientation options

The code offers analytical and numerical methods for calculating the orientation–averaged cross sections and scattering matrix. The analytical method is exact (to the truncation error of the expansions), but can also be time consuming, especially for large systems. The numerical method uses a Monte Carlo integration over incident directions and should be used when only two or three digits of precision are needed.

`random_orientation`: logical: `.true.` for random orientation results via the analytical T matrix scheme. Not compatible with plane boundary or periodic lattice models. Default: `.false.`

`incidence_average`: logical: `.true.` for random orientation results via the Monte Carlo numerical scheme. Be sure to set `random_orientation` to `.false.`.

`number_incident_directions`: Number of incident directions to sample for Monte Carlo integration (100).

5.6 Options related to scattering matrix output

For fixed orientation calculations the 4×4 real–valued Stokes scattering matrix will, in general, exist in the 2–D space representing points on the surface of a sphere. The random orientation scattering matrix, on the other hand, will depend solely on a polar scattering angle defined relative to the incident direction.

`calculate_scattering_matrix`: logical, whether or not to calculate the scattering matrix. (`.true.`).

`incident_frame`: Logical. This selects the reference coordinate frame upon which the scattering angles are based. When true, the frame is based on the incident field so that $\theta = 0$ is the forward scattering direction (i.e., the direction of incident field propagation) and $\theta = \beta$, $\phi = 180$ is the target z axis direction. When false, the frame is based on the target frame, so that $\theta = 0$ corresponds to the target z axis and $\theta = \beta$, $\phi = \alpha$ is the incident field direction `.false.`. Random orientation calculations set this to true.

`scattering_map_model`: Integer, =0 will have the scattering matrix printed at a range of polar angles (θ), with angular increment given in degrees by `scattering_map_increment`. The scattering plane coincides with the incident azimuth plane. The limits on θ depend on the number of plane boundaries N_B. `scattering_map_model`= 1 will print the scattering matrix in a pair ($\cos\theta > 0, < 0$) of 2D arrays, with coordinates $k_x = \sin\theta\cos\phi, k_y = \sin\theta\sin\phi$. The number of points on each axis is given by `scattering_map_dimension`. Default is 0, and random orientation calculations set to 0.

`scattering_map_increment`: Scattering matrix is written at scattering angles from 0 to 180 degrees at increments of `scattering_map_increment`. Relevant for `scattering_map_model=0` and/or random orientation.

5.7 Options for fixed orientation calculations

`incident_alpha_deg`: The azimuth angle α of the incident field propagation direction, relative to the sphere cluster coordinate system, in degrees (0.0).

`incident_beta_deg`: Polar angle β for propagation direction, degrees (0.0).

`incident_sin_beta`: A more generalized specification of the incident field state. Can be > 1, corresponding to evanescent field incident excitation. This is used to specify an incident field corresponding to a slab waveguide mode for systems containing two or more plane boundaries.

`azimuthal_average`: Logical, when = true the scattering matrix values are analytically averaged over azimuthal (i.e., scattering plane) angle. When this is in effect the output scattering matrix values will be listed as a function of θ only, similar to that for random orientation. I developed this option for a very specialized purpose, involving the direct simulation of reflection from a modeled particle deposit when incidence is normal. Default is false.

5.8 Near field calculations

The code can calculate electric and magnetic vector field values at points internal or external to the spheres, and in systems having periodicity and plane boundaries. The algorithm is optimized to make efficient the calculation of values in a regular 1-D, 2-D, or 3-D grid. This option requires `random_orientation=.false.`.

`calculate_near_field`: Logical. Enables calculation of electric and magnetic vector amplitude values in a specified rectangular solid region of space.

`near_field_minimum_border`: $x_{min}, y_{min}, z_{min}$ of the rectangular region.

`near_field_maximum_border`: $x_{max}, y_{max}, z_{max}$ of the rectangular region. Setting $x_{min} = x_{max}$ will make the region 2D in the $y - z$ plane, and likewise for the other directions. These numbers are not modified by `length_scale_factor`.

`near_field_step_size`: Spatial increment between near field calculation points.

`near_field_calculation_model`: Integer, controls where the incident field appears in the calculation results; =1 is the standard model, where the external field is = scattered + incident and the field inside the spheres is calculated from the internal field expansions; $\neq 1$ has the external field due solely to the scattered field, and the field inside the particles is now internal-incident (1).

`store_surface_vector`: logical, = `.true.` enables an accelerated algorithm for calculation of field values. Should be `.true.` unless you are doubtful about the field values being calculated using this option, and want to compare them to those calculated using the direct, non–accelerated algorithm (which is slow!). The next two options are relevant when `store_surface_vector`= .true. .

`near_field_expansion_spacing`: the size of the grid used for re–expansion of fields resulting from plane boundary and periodic lattice interactions. Default is 5.0. Experiment with this if your calculations are taking too long.

`near_field_expansion_order`: the expansion truncation order used for the grid re–expansion of the secondary field, per the above option. Should scale with `near_field_expansion_spacing` following a Mie–type criteria. The bigger the value, the more accurate the results, but the slower the calculations and the greater the memory demands (10).

`near_field_output_file`: Character. Near field output file name. The first line of the file contains the text "run number", the second line is an integer corresponding to the run number. The 3rd line is an integer N_{is}: the number of spheres intersecting the rectangular plot region, and the next N_{is} lines contain the x, y, z and radius a of the intersecting spheres. The next line is N_{ib}: the number of plane boundaries intersecting with the plot region, followed by N_{is} lines containing the z position of the boundary (the first z will always be zero, per the convention used in the code). The next two lines have `near_field_minimum_border` and `near_field_maximum_border`, followed by a line with $N_x,\ N_y,\ N_z$: the number of grid points in the x, y, z directions. The total points are $N = N_x N_y N_z$. The next N lines have 27 columns associated with each calculation point: $x, y, z,\ \mathbf{E}_\parallel, \mathbf{H}_\parallel,\ \mathbf{E}_\perp, \mathbf{H}_\perp$; each vector field has 6 columns: Re E_x, Im E_x, Re E_y, and so on, and $\parallel, \perp$ correspond to the parallel and perpendicular incident polarization states. The whole slew of numbers is repeated for each new run.

5.9 Options for multiple runs, looping over parameters, and input termination

`end_of_options`: Stops reading input data, performs calculations, and terminates program. The last read line in every input file needs to be this.

`new_run`: signals the code to run a new job once the job, specified by the options appearing before `new_run`, is completed. Options that are to be changed from the previous run should appear after `new_run`, and all other options will be held at the same values from the previous run. The output from the new job will be appended to the output and near field files unless new file names are given as part of the new run options. There is no limit on the number of `new_run` statements, but the last job needs to be followed with `end_of_options`.

`loop_variable`: Performs looped calculations. The format is

`loop_variable`
looped_parameter_ID
parameter_start parameter_end parameter_increment

The option *looped_parameter_ID* corresponds to a parameter ID for a numerical characteristic of the run, such as `incident_beta_deg` or `length_scale_factor`. The next line contains the numerical starting values, ending values, and increment for the parameter. These loops can be nested up to three deep. They can be followed with a `new_run` or `end_of_options`.

6 Output file

Quantities appearing in the output file that need some explanation are

`volume cluster radius` $= R_V = \left(\sum_{ext} a_i^3 \right)^{1/3}$

`area mean sphere radius` $= R_A = \left(\frac{1}{N_{ext}} \sum_{ext} a_i^2 \right)^{1/2}$

`cross section radius` : the dimensionless radius used to define aggregate efficiency factors; dimensionless cross sections would be $C = \pi a_{cs}^2 Q$, dimensional cross sections = dimensionless/$(2\pi/\lambda_0)^2$. For plane wave incidence or random orientation calculations, the cross section radius is the volume mean radius of the cluster, defined by $4\pi a_V^3/3 = V_{clus}$. For Gaussian beam incidence a_{cs} =`1/gaussian_beam_constant`/$\sqrt{2}$, and for periodic lattices $\pi a_{cs}^2 = W_x W_y$.

`absorption, volume absorption efficiencies` : These appear when `write_sphere_data=.true.`. The first is the standard absorption efficiency for the sphere surface and measures all absorption within the surface. The second subtracts out absorption due to spheres internal to the surface, and includes only the absorption associated with the inside material of the surface.

For simple systems (no plane boundaries, not periodic) the efficiency factors for parallel, perpendicular, and unpolarized incident waves are listed. Systems with plane boundaries will include the extinction and scattering efficiencies in the forward and backwards directions. Periodic systems (with or without plane boundaries) will report the reflection, absorption, and transmission of the lattice, as opposed to the efficiency factors.

For random orientation calculations, and fixed–orientation calculations using `azimuthal_average=.true`, the $S_{11}(\theta)$ scattering matrix element is normalized so that

$$2 \int_0^\pi S_{11}(\theta) \sin\theta \, d\theta = 1$$

The other matrix elements are scaled with S_{11}.

References

[1] D. W. Mackowski, M. I. Mishchenko, Calculation of the T matrix and the scattering matrix for ensembles of spheres, J. Opt. Soc. Amer. A 13 (1996) 2266–2278.

[2] D. W. Mackowski, Exact solution for the scattering and absorption properties of sphere clusters on a plane surface, J. Quant. Spectrosc. Radiat. Transfer 109 (2007) 770–788.

[3] D. W. Mackowski, M. I. Mishchenko, Direct simulation of multiple scattering by discrete random media illuminated by Gaussian beams, Phys. Rev. A 83 (1) (2011) 013804–+. doi:10.1103/PhysRevA.83.013804.

[4] D. Mackowski, The extension of Mie theory to multiple spheres, in: W. Hergert, T. Wriedt (Eds.), The Mie Theory, Springer Berlin/Heidelberg, 2012, pp. 223–256.

[5] D. Mackowski, A general superposition solution for electromagnetic scattering by multiple spherical domains of optically active media, "J. Quant. Spectrosc. Radiat. Transfer" 133 (2014) 264 – 270.

[6] D. W. Mackowski, L. Kolokolova, Application of the multiple sphere superposition solution to large–scale systems of spheres via an accelerated algorithm, Journal of Quantitative Spectroscopy and Radiative TransferSubmitted (2022).