Multiple scattering of light in nanoparticle assemblies: user guide for the TERMS program

D. Schebarchov, A. Fazel-Najafabadi, E. C. Le Ru, B. Auguié*

The MacDiarmid Institute for Advanced Materials and Nanotechnology
School of Chemical and Physical Sciences, Victoria University of Wellington,
PO Box 600, Wellington 6140, New Zealand

Abstract

We introduce TERMS, an open-source Fortran program to simulate near-field and far-field optical properties of clusters of particles. The program solves rigorously the Maxwell equations via the superposition T-matrix method, where incident and scattered fields are decomposed into series of vector spherical waves.

TERMS implements several algorithms to solve the coupled system of multiple scattering equations that describes the electromagnetic interaction between neighbouring scatterers. From this formal solution, the program can compute a number of physically-relevant optical properties, such as far-field cross-sections for extinction, absorption, scattering and their corresponding circular dichroism, as well as local field intensities and degree of optical chirality. By describing the incident and scattered fields in a basis of spherical waves the T-matrix framework lends itself to analytical formulas for orientation-averaged quantities, corresponding to systems of particles in random orientation; TERMS offers such computations for both far-field and near-field quantities of interest. This user guide introduces the program, summarises the relevant theory, and is supplemented by a comprehensive suite of stand-alone examples in the website accompanying the code.

Contents

1 Introduction 2
1.1 General features 2
1.2 Relation to other codes 3
1.3 Aims of this manual 4
1.4 Licensing 4
1.5 Disclaimer and request for feedback . 4

2 Getting started 4
2.1 Installation 4
2.2 Initial steps 5
2.3 Minimal example 5
2.4 Range of validity 6

3 Underlying principles of the code 7
3.1 Vector spherical wave functions . . . 7
3.2 The T-matrix ansatz 8
3.3 Rotation and translation properties . 8
3.4 STM for multiple scatterers 11
3.5 Collective T-matrix constructions . . 13
3.6 Far-field cross-sections 14
3.7 Near-field quantities 17
3.8 Solution schemes 19

4 Conclusion and outlook 21

A Appendix 25
A.1 Keywords 25
A.2 Organisation of the code 31
A.3 List of subroutines . . 32

*Corresponding authors
Email addresses: dmitri.schebarchov@gmail.com (D. Schebarchov), atefeh.fazelnajafabadi@vuw.ac.nz (A. Fazel-Najafabadi), eric.leru@vuw.ac.nz (E. C. Le Ru), baptiste.auguie@vuw.ac.nz (B. Auguié)

1. Introduction

TERMS – acronym for T-matrix for Electromagnetic Radiation with Multiple Scatterers – is a suite of Fortran 90 routines to simulate light scattering by rigid clusters of particles immersed in a homogeneous, non-absorbing medium. The calculation is based on the superposition T-matrix (STM) method, an extension of Waterman's T-matrix formalism[1–4] to multiple scatterers[5–7]. The incident and scattered fields are expanded into series of vector spherical wave functions (VSWFs), which can be interpreted as a multipolar decomposition. For linear media, the coefficients describing the scattered field follow a linear relationship with those of the known incident field; this linear relationship is expressed through the so-called T-matrix, which encodes the full information about a scatterer's linear optical properties and its response to an arbitrary incident excitation. Where several particles are present, light scattered by one particle can contribute to the excitation of the others; the self-consistent set of exciting and scattered fields from each particle, and the cluster as a whole, is expressed in the STM framework as a linear system of equations involving the single particle T-matrices, and translation matrices to transform the VSWFs from one particle to another. The solution of this system of equations enables the calculation of near-field quantities as well as far-field cross-sections, for specific directions of incidence and polarisation, or after analytical orientation-averaging.

In principle many types of particle shapes can be used in TERMS, provided an external program can calculate and export their corresponding T-matrix. TERMS provides built-in calculations of single-particle T-matrices for homogeneous and multi-layered spheres, and our Matlab code SMARTIES can export accurate T-matrices for oblate and prolate spheroidal particles in a compatible format[8]. The maximum number of particles that TERMS can consider is typically about a few hundred for standard computers and small maximum multipolar order, although larger systems could be modelled using an iterative linear solver[9–11] or implementing a hierarchical fast multipole method[11,12].

This guide aims to describe the program from a user's perspective, illustrate the types of calculations that it can perform, and highlight its strengths relative to other computational methods. The code is released as open-source, and we welcome contributions from the community. A dedicated website[13] provides stand-alone examples to illustrate the program's capabilities in specific applications.

1.1. *General features*

From a generic description of the scattering problem, consisting in the position and orientation of N particles, dielectric functions or input T-matrix for each particle, and the incident wavelength(s), the program can perform three main types of simulations:

1. *Near-field mode*, to map local fields and derived quantities at fixed incidence, or with orientation-averaging.
2. *Far-field mode*, to calculate cross-sections (extinction, scattering, and absorption, as well as corresponding linear and circular dichroism) at fixed incidence and with orientation-averaging.
3. *Polarimetry mode*, to calculate Mueller matrices, Stokes parameters, and differential scattering cross-sections at specified scattering angles.

At runtime, the program sets up a linear system of equations in the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, where the matrix $\mathbf{A}$ is constructed from a given set of single-particle T-matrices, particle coordinates and orientations, and the vector (or matrix) $\mathbf{b}$ characterises the specified incident plane wave excitation(s). The unknown $\mathbf{x}$ determines the self-consistent field exciting each scatterer, as described in more details in Sec. 3. The linear system is then solved using one of several schemes selected by the user:

0. Application of a (direct) solver to determine $\mathbf{x}$, corresponding to the particle-centred scattering coefficients for one or more specific incident field(s), $\mathbf{b}$.
1. Direct inversion of the matrix $\mathbf{A}$ to determine the particle-centred T-matrices for the cluster of particles.[14]
2. Stout *et al.*'s[14,15] iterative scheme for calculating the particle-centred T-matrices.
3. Mackowski & Mishchenko's[7,9,16–18] scheme for calculating the particle-centred T-matrices.

Implementation of these multiple solution schemes in a modular code-base is a core feature in TERMS; we hope it will prove useful for designing, testing, and benchmarking various methods, and perhaps lead to the implementation of new improved algorithms.

The most notable features of TERMS include:

- Export of the collective T-matrix describing the entire cluster of particles.
- Import of general T-matrices, which can be pre-generated using TERMS or another program, such as SMARTIES (for spheroids)[8].
- Built-in calculation of individual T-matrices for stratified/coated spheres described by Mie theory[19].
- Calculation of partial absorption cross-sections in each layer of coated spheres, following Mackowski[20].
- Calculation of orientation-averaged far-field cross-sections and associated circular dichroism[21].
- Calculation of orientation-averaged near-fields[15] and optical chirality[22].
- Calculation of the Mueller matrix and Stokes parameters for specific incidence and scattering angles[4].
- Possible compilation with all double-precision variables promoted to quad-precision[23].
- Export the output results in plain text or "HDF5" file format[24].

1.2. Relation to other codes

TERMS belongs to the family of codes implementing the superposition T-matrix method for collections of scatterers. Other implementations have been described in the literature[7,14,15,21,25–29] (for a comprehensive review, we refer the reader to Ref. 30); available open-source programs include that of Mishchenko & Mackowski for spherical particles and optically-active media (MSTM)[18], and for non-spherical particles the recent additions of FastMM by Markkanen and Yuffa[11] and QPMS by Nečada and Törmä[31].

Among the many available techniques to solve light scattering problems[32], the STM method holds distinct advantages over purely numerical techniques such as the Finite Elements Method (FEM)[33], the Discrete Dipole Approximation (DDA)[34], or the Finite Differences Time Domain (FDTD) method[35]. Unlike STM, these techniques require discretising the whole cluster geometry and solving the full electromagnetic problem for every direction of incidence. Other notable advantages include:

- Orientation-averaged far-field properties can be obtained at very little computational cost, with analytical formulae[21,36–39]. Orientation-averaged near-field quantities can also be computed[15,22], albeit with some computational overhead, providing analytical benchmark results[40].
- For clusters of several identical particles only one T-matrix needs to be calculated.
- Within its domain of validity the Extended Boundary Condition Method (EBCM), and the T-matrix framework more broadly, is typically faster and more accurate than competing methods, and is therefore often used for benchmark calculations[4].
- The multipolar decomposition of electromagnetic fields can provide physical insight into complex optical responses[41].

It should be noted that the STM method is not without its limitations,

- Closely-spaced scatterers can lead to inaccurate results, or require very large multipolar orders, and the exact domain of applicability of the method in such situations is not fully-understood[23]. Some proposals to overcome this issue have recently been demonstrated[42], and may be implemented in TERMS in the future.
- The calculation of local fields in the vicinity of elongated nanoparticles is limited by the Rayleigh Hypothesis[43].
- Our particular implementation is limited to relatively small numbers of particles (a few tens to hundreds, on a typical workstation, and depending on their size parameter).
- Numerical instabilities arise at high maximum multipolar order (from approximately $n_{\max} \approx 30$ typically), preventing the calculation of accurate T-matrix elements in double-precision, and leading to ill-conditioning of matrices.
- Nonspherical particle shapes require first computing the T-matrix with an external program. TERMS has built-in functions for homogeneous and multi-layered spheres, and for non-spherical particles the T-matrix can be obtained from a variety of methods, from

Mishchenko's EBCM implementation for axisymmetric particles[4] and SMARTIES[8] in particular for spheroids, the surface-integral equation (SIE) method[44], the volume-integral equation method[11], and other algorithms have been proposed for the Discrete Dipole Approximation[45], or general solvers such as the Finite-Element Method[29].

Different superposition T-matrix algorithms have been proposed, with their own strengths and weaknesses depending on the type of particles and their configuration; an important feature of TERMS is that it is possible to compare several algorithms and choose the most suitable for a given problem. For complex geometries, especially compact ones, the invariant-embedding T-matrix method[46], the Surface Integral Equation[47] or Volume Integral Equation methods[44] may provide better alternatives. Our implementation also does not currently consider periodic arrays of scatterers[29,31,48], or the presence of a substrate[49,50].

1.3. Aims of this manual

TERMS is accompanied by a comprehensive set of examples available online[13]; this user guide aims to provide a useful complement introducing i) the necessary background information about the method; ii) the initial steps required to install and run the program; iii) a high-level description of the program and its capabilities.

1.4. Licensing

TERMS is made available under the Mozilla Public License Version 2.0, but note that parts of the code include external Fortran libraries under different licencing, such as LAPACK (BSD), and HDF5 routines (copyright The HDF Group)[24].

1.5. Disclaimer and request for feedback

The TERMS program is provided "as is", without warranty of any kind. While we have tested the program in a large number of configurations we cannot provide any guarantee as to the accuracy or validity of simulation results obtained with the program. The user is strongly encouraged to perform their own reference checks against other methods, but also internal consistency checks by switching the solution method, increasing the multipolar order, and if necessary using quad precision.

We welcome comments, reports of errors, and suggestions of new features, which can be addressed directly to the authors or via the code's hosting website.

2. Getting started

Figure 1 displays a partial overview of TERMS' capabilities, with calculation results taken from the online documentation[13], which includes over 20 self-contained examples illustrating all the different options for using TERMS. We do not repeat these examples in this user guide but instead provide the basic common starting point which can be adapted for any specific use case.

2.1. Installation

The code was developed and tested predominantly on standard personal desktop and laptop computers running Linux (Ubuntu 18.04 LTS) and MacOS, as well as the Rāpoi HPC Cluster at Victoria University of Wellington. We've also successfully installed and run TERMS on Windows via the Windows Subsystem for Linux (WSL 2). Our Linux configuration includes: the `gfortran` compiler in `gcc` version 7.4.0, `HDF5` software with libraries `"libhdf5-dev"`, BLAS `"libblas-dev"` and LAPACK `"liblapack-dev"`. We advise using a fairly recent Fortran 90 compiler (`gcc` versions below 6 have caused problems), and recent HDF5 release `"HDF5-1.12.1"`[24].

There are two ways for producing the executable file:

- (Recommended) using `Cmake`, with parameters defined in `CMakeLists.txt`:

  ```
  > cd build
  > cmake ..
  > make
  ```

  will produce an executable `terms` for your machine, which you can leave in its location or move elsewhere.

  Alternatively,

- A basic script is provided under `build/buildTERMS.sh` to specify the compilation options (double vs quad precision, debug mode, and use of a system's LAPACK).

  ```
  > cd build
  > bash buildTERMS.sh
  ```

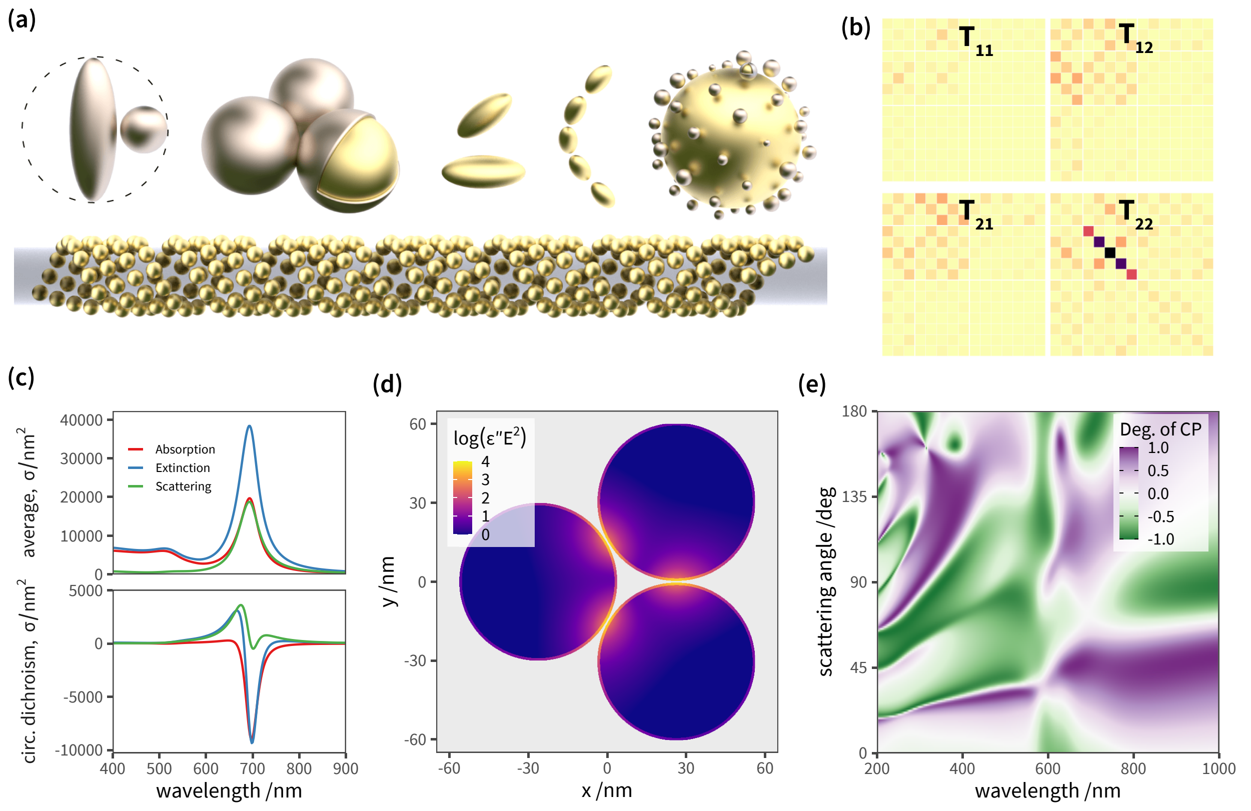

Figure 1: Illustrative overview of TERMS. (a) Pictorial representations of nanoparticle clusters studied with TERMS (Left to right: closely-spaced dimer[23], trimer of Au@Pd core-shell antennas[51], chiral dimer and helix of Au spheroids[40] (also bottom helix), hybrid antenna-satellite photocatalyst[52]). (b) Collective T-matrix of a chiral dimer of prolate spheroids (after online example 13[13]; the colour maps the modulus of the T-matrix elements, here truncated at $n_{\max} = 3$). (c) Far-field spectra of orientation-averaged cross-sections (absorption, scattering, extinction, and their corresponding circular dichroism in the bottom panel); the structure consists of a chiral dimer of prolate Au spheroids in water[40]. (d) Near-field map of $\Im(\varepsilon)|\mathbf{E}|^2$ in a trimer of Au@Pd core-shell antennas[51]. (e) Dispersion map of the degree of circular polarisation displayed by a helix of five prolate Au spheroids (after online example 08[13]).

will produce an executable `terms` for your machine, which you can leave in its location or elsewhere.

The executable reads user-defined instructions describing the scattering problem from an input file, and is called as follows:

```
> ./terms inputfile > messages.log
```

The results of calculations are stored in specific output files in the current directory and displayed in the terminal together with any errors and warnings (it can be convenient to redirect the standard output to a log file, as in the example above).

2.2. Initial steps

The main input parameters are read from a plain text input file (line by line and from left to right; blank lines are ignored). Each line is interpreted as a sentence and split into space-separated words. The first (left-most) word is interpreted as a case-sensitive keyword, and the subsequent words as arguments for that keyword. In each sentence, text from the first word starting with the hash character (#) is interpreted as a human-readable comment and thus ignored by the program. All the supported keywords and corresponding arguments are documented in Appendix A.1. The order of keywords generally doesn't matter, with just two exceptions: `ModeAndScheme` must be the first keyword, and `Scatterers` must be the last.

2.3. Minimal example

We first illustrate the use of TERMS on a simple case, the calculation of far-field spectra for absorption, scattering, and extinction with a structure consisting of four gold spheres immersed in water. Most simulation parameters are kept to their default values.

This simulation uses the following `input` file,

```
ModeAndScheme 2 3
Wavelength 300 900 300
Medium 1.7689  # epsilon of water

Scatterers  4
Au  31.5      0         -50  30
Au -31.5      0         -50  30
Au  22.2738  22.2738  50  30
Au -22.2738 -22.2738  50  30
```

The program is run with the command `./terms input > log`, where the `log` file contains information about the simulation (how detailed depends on the optional `Verbosity` argument). The output for this simulation consists of a number of plain text files, storing the far-field cross-sections:

- Files `cs(Abs|Ext|Sca)OA` contain orientation-averaged cross-sections.
- Files `cd(Abs|Ext|Sca)OA` contain orientation-averaged optical activity.
- Files `cs(Abs|Ext|Sca)(1X|2Y|3R|4L)` contain fixed-orientation cross-sections for the respective polarisation ('`X`', '`Y`': 2 orthogonal linear polarisations; '`R`', '`L`': right and left circular polarisations).
- Files `csAbs(1X|..)_scat(00i)coat(j)` contain partial absorption cross-sections inside multi-layered spheres.

This plain text output can become inconvenient when running many simulations; TERMS provides an option to produce a single Hierarchical Data Format (`HDF5`) output file[24], with the output quantities stored under individual fields instead of separate files. The `HDF5` file format can be read in many other programs, using e.g the built-in `h5read` function in Matlab, or packages `rhdf5` for R, `h5py` for Python, `HDF5.jl` for Julia, to list only a few popular options.

The documentation's website features many minimal examples of use for each option of the program, and with various cluster configurations.

2.4. Range of validity

The T-matrix method is often used as a benchmark for other numerical techniques such as DDA or FEM, as it provides very accurate results. A sufficiently-high value of the maximum multipole order, $n_{\max}$, should be chosen for each simulation, and convergence of the results with increasing $n_{\max}$ is often a good indicator of the accuracy of the results. TERMS performs internal checks of convergence for the far-field cross-sections, by comparing the relative error between successive partial sums over multipole orders 1 to $n_{\max}$. We strongly advise users to monitor the messages and check for issues with convergence. It is also useful to re-run calculations with a higher value of $n_{\max}$ and check that the results do not differ. In near-field calculations a higher $n_{\max}$ value is generally needed, and we find that values above 30 can require switching to quad precision. The challenging case of non-spherical particles with strongly-overlapping circumscribed spheres pushed some calculations to use $n_{\max}$ above 50; even with quad precision arithmetic the accuracy eventually deteriorates (above 60, typically). We emphasise that these are extreme cases; in many standard situations a low value of $n_{\max}$ is sufficient (8 is the default value). The coupled-dipole method, widely used in nano-optics, corresponds roughly to setting $n_{\max} = 1$.

Single-particle T-matrices computed with Mie theory are generally accurate up to $n_{\max} = 60$. Following Wiscombe's criterion, this corresponds to a size parameter of 45, or a sphere radius of 2 microns in vacuum for visible light. For spheroids, SMARTIES enables accurate calculation of T-matrix elements with an aspect ratio of up to 100, and similar size limitations as Mie theory[8].

Multiple-scattering generally introduces a loss of precision compared to single-particle calculations, and requires larger values of $n_{\max}$. The user is advised to consider the different solution schemes implemented in TERMS, as they can offer substantial benefits in specific situations. For instance, Stout and co-workers introduced a balancing scheme[15] that stabilises the numerical calculations and proves very effective for closely-spaced resonant particles. TERMS has extended this improvement to other schemes by default (controlled with the keyword `StoutBalancing`). A dramatic difference between Scheme 2 and 3 is observed when particles are widely-separated: our implementation of Mackowski & Mishchenko's scheme fails where separations are above a few hundred nanometres even at large $n_{\max}$, while Stout's scheme maintains good accuracy without requiring a $n_{\max}$ value much larger than dictated by the single-particle response. The key difference between the two schemes is that Mackowski & Mishchenko's translates all VSWFs to a common origin, while Stout's maintains particle-centred expansions throughout[14–18,53].

The performance of Mackowski & Mishchenko's scheme is usually very good, in both accuracy and speed, and is chosen as the default.

The results of TERMS calculations have been validated against Mackowski & Mishchenko's MSTM code for collections of spheres[18], and against a commercial Finite Element package (Comsol[54]) for dimers of spheroids[23].

3. Underlying principles of the code

In the following presentation, the complex electric field is denoted by $\mathbf{E}(\mathbf{r}, t)$, where $\mathbf{r}$ is a point coordinate and t is time; we assume harmonic time dependence at angular frequency ω, so that $e^{-i\omega t}$ factors out and is omitted from the rest of the discussion.

3.1. Vector spherical wave functions

We define the vector spherical wave functions (VSWFs) as,

$$\mathbf{M}_{nm}^{(\zeta)}(k\mathbf{r}) = \frac{1}{\sqrt{n(n+1)}} \nabla \times (\psi_{nm}^{(\zeta)}(k\mathbf{r})\mathbf{r}), \quad (1)$$

$$\mathbf{N}_{nm}^{(\zeta)}(k\mathbf{r}) = \frac{1}{k} \nabla \times \mathbf{M}_{nm}^{(\zeta)}(k\mathbf{r}). \quad (2)$$

with k the wavenumber and

$$\psi_{nm}^{(\zeta)}(k\mathbf{r}) = z_n^{(\zeta)}(kr) Y_{nm}(\theta, \varphi), \quad (3)$$

where $z_n^{(\zeta)}$ are spherical Bessel functions. For our purposes we only require $\zeta = 1$ ($z_n^{(1)} = j_n$, spherical Bessel functions of the first kind) and $\zeta = 3$ ($z_n^{(3)} = h_n$, spherical Hankel functions of the first kind), referred to as *regular* and the *irregular* functions, respectively, which are linearly independent. Henceforth, for brevity and notational convenience we refer to $\psi_{nm}^{(3)}$ as simply ψ_{nm}, and $\psi_{nm}^{(1)}$ as $\widetilde{\psi}_{nm}$. Furthermore, the tilde will also be placed over the coefficients (e.g. $\widetilde{\mathbf{a}}$) to explicitly indicate a regular basis set.

The spherical harmonics Y_{nm} for $|m| \le n$ we write as,

$$Y_{nm}(\theta, \varphi) = \gamma_{nm} \sqrt{n(n+1)} P_n^m(\cos\theta) \mathrm{e}^{\mathrm{i}m\varphi}, \quad (4)$$

where the associated Legendre functions $P_n^m(\cos\theta)$ are defined using the Condon-Shortley phase and

$$\gamma_{nm} := \sqrt{\frac{(2n+1)}{4\pi n(n+1)} \frac{(n-m)!}{(n+m)!}}. \quad (5)$$

This convention is consistent[1] with our main references.[4,14,19,55]

Formally, n can run from 0 up to ∞, though numerically all series of VSWFs are truncated to some maximum multipole order $n_{\max}$. We also introduce the composite index $p(n, m)$ for convenience, defined as

$$p := n(n+1) + m \quad (6)$$

with,

$$n = \mathrm{Int}(\sqrt{p}) \quad (7)$$
$$m = p - n(n+1). \quad (8)$$

A general regular solution to the Helmholtz equation can be expressed in the VSWF basis as,

$$\begin{aligned}\widetilde{\mathbf{E}}(k\mathbf{r}) &= \sum_{n=1}^{n_{\max}} \sum_{m=-n}^{n} [\widetilde{a}_{1,nm} \widetilde{\mathbf{M}}_{nm}(k\mathbf{r}) + \widetilde{a}_{2,nm} \widetilde{\mathbf{N}}_{nm}(k\mathbf{r})] \\ &= \sum_{s=1}^{2} \sum_{p=1}^{p_{\max}} \widetilde{a}_{s,p} \widetilde{\mathbf{w}}_{s,p}(k\mathbf{r}) \\ &= \sum_{l=1}^{l_{\max}} \widetilde{\mathbf{w}}_l(k\mathbf{r}) \widetilde{a}_l \;=:\; \widetilde{\mathbf{W}}(k\mathbf{r}) \widetilde{\mathbf{a}}, \end{aligned} \quad (9)$$

where $\widetilde{\mathbf{a}} \in \mathbb{C}^{l_{\max}}$ is a column vector of coefficients, $\widetilde{\mathbf{W}} = [\widetilde{\mathbf{w}}_1, \widetilde{\mathbf{w}}_2, \ldots, \widetilde{\mathbf{w}}_{l_{\max}}]$ is a basis-set *pseudo-matrix* of dimension $3 \times l_{\max}$, i.e. a row vector composed of column vectors

$$\widetilde{\mathbf{w}}_{l(s,n,m)} := \begin{cases} \widetilde{\mathbf{M}}_{nm} & \text{for} \quad s = 1 \\ \widetilde{\mathbf{N}}_{nm} & \text{for} \quad s = 2 \end{cases} \quad (10)$$

and $l_{\max}$ is the maximal value of another composite index l, introduced for convenience

$$\begin{aligned} l &:= (s-1) n_{\max}(n_{\max}+2) \quad (11) \\ &\quad + n(n+1) + m \\ &= (s-1) p_{\max} + p \\ &\le l_{\max} \\ l_{\max} &= 2 n_{\max}(n_{\max}+2) = 2 p_{\max}, \quad (12) \end{aligned}$$

[1]However, note that Mishchenko *et al.*[4] define their $\psi_{nm}^{(\zeta)}(k\mathbf{r})$ as $z_n^{(\zeta)}(kr) P_n^m(\cos\theta) \mathrm{e}^{\mathrm{i}m\varphi}$, which must be multiplied by $\gamma_{nm}\sqrt{n(n+1)}$ to match our $\psi_{nm}^{(\zeta)}(k\mathbf{r})$ in (3).

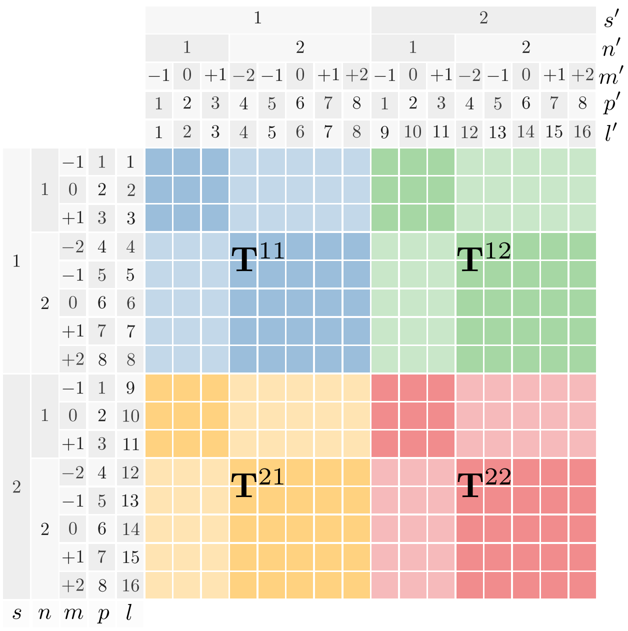

Figure 2: Pictorial representation of a T-matrix and relevant indices for $n_{max} = 2$. The matrix elements coupling $\mathbf{M}_{nm}$ with $\mathbf{M}_{n'm'}$ (magnetic–magnetic) are in blue, $\mathbf{N}_{nm}$ with $\mathbf{N}_{n'm'}$ (electric–electric) in red, the off-diagonal blocks coupling $\mathbf{M}_{nm}$ with $\mathbf{N}_{n'm'}$ (magnetic–electric) and $\mathbf{M}_{n'm'}$ with $\mathbf{N}_{nm}$ (electric–magnetic) are in orange and green, respectively. The elements coupling VSWFs of the same multipole order ($n = n'$) are of darker shade.

where $s \in \{1, 2\}$ is sometimes referred to as the *parity* or *mode* index, corresponding to either $\mathbf{M}$ or $\mathbf{N}$ functions.

The irregular variant of (9) is obtained by simply removing the overhead tildes (e.g. $\widetilde{\mathbf{W}} \to \mathbf{W}$), which corresponds to switching the radial dependence from $j_n(kr)$ to $h_n(kr)$ throughout.

3.2. *The* T*-matrix ansatz*

Outside a given scatterer, the total field $\mathbf{E}_{\text{tot}}(k\mathbf{r}) = \widetilde{\mathbf{E}}_{\text{inc}}(k\mathbf{r}) + \mathbf{E}_{\text{sca}}(k\mathbf{r})$ is partitioned into a known incident contribution $\widetilde{\mathbf{E}}_{\text{inc}}(k\mathbf{r})$ and unknown scattered contribution $\mathbf{E}_{\text{sca}}(k\mathbf{r})$. Both contributions are expanded in terms of VSWFs up to some multipole order n_{max},

$$\widetilde{\mathbf{E}}_{\text{inc}}(k\mathbf{r}) = E\widetilde{\widetilde{\mathbf{W}}}(k\mathbf{r})\widetilde{\mathbf{a}}, \tag{13}$$
$$\mathbf{E}_{\text{sca}}(k\mathbf{r}) = E\mathbf{W}(k\mathbf{r})\mathbf{a}, \tag{14}$$

where E corresponds to the incident field's amplitude (usually taken as unity, $E = |\widetilde{\mathbf{E}}_{\text{inc}}| = 1$), and $\widetilde{\mathbf{a}} \in \mathbb{C}^{l_{\text{max}}}$, $\mathbf{a} \in \mathbb{C}^{l_{\text{max}}}$ are the incident and scattered coefficients, respectively. The association of $\widetilde{\mathbf{E}}_{\text{inc}}$ with regular (or *incoming*) and $\mathbf{E}_{\text{sca}}$ with irregular (or *outgoing*) VSWFs is a choice motivated by physical reasoning: (i) $\widetilde{\mathbf{E}}_{\text{inc}}(k\mathbf{r})$ ought to be well defined everywhere within a finite distance from the origin, which rules out irregular VSWFs due to their singular behaviour at $\mathbf{r} = 0$; and (ii) $\mathbf{E}_{\text{sca}}(k\mathbf{r})$ ought to satisfy the outgoing Sommerfeld radiation condition, requiring that $|\mathbf{E}_{\text{sca}}(\mathbf{r})| \to 0$ as $1/r$ as $|\mathbf{r}| \to \infty$, which rules out regular VSWFs due to their divergence in the far field. Given the linearity of the governing Maxwell equations in linear media, the T-matrix method expresses the linear dependence between $\widetilde{\mathbf{a}}$ and $\mathbf{a}$,

$$\mathbf{a} = \mathbf{T}\,\widetilde{\mathbf{a}}, \quad \text{or} \quad a_l = \sum_{l'} T_{ll'}\widetilde{a}_{l'}, \tag{15}$$

where $\mathbf{T}$ is the so-called "transition" or "transfer"[4,14] matrix (T-matrix for short), which depends on the scatterer's characteristics at a given wavelength but is independent of illumination, encoded in $\widetilde{\mathbf{a}}$. For a spherically symmetric scatterer centred at the origin, $\mathbf{T}$ is a diagonal matrix with the diagonal elements determined analytically by Mie theory. For other particle shapes, TERMS requires that the T-matrix be provided as input, with a format specified in App. A.1 (keyword `TmatrixFiles`). Note that the T-matrix may also represent the response of a composite scatterer comprising multiple particles; TERMS can in fact calculate such a collective T-matrix from individual one-body T-matrices,[14,15] and re-use it as input to simulate the scattering properties of a superstructure of such elements[29].

3.3. *Transformation under rotation/translation of coordinates*

The STM method requires transforming the series expansions of the fields from one origin to another, such as from the centre of one particle to a neighbour's, or to a common origin referred to as the *global frame's*. Typically the T-matrix of a non-spherical particle will have been calculated in a convenient orientation, e.g. for axisymmetric particles with symmetry axis along z, requiring rotations in changing reference frame as illustrated in Fig. 3.3. In the following we summarise useful relations for the translation and rotation of VSWFs. We take the notational convention that expressions in local coordinate frames are specified with a superscript in brackets.

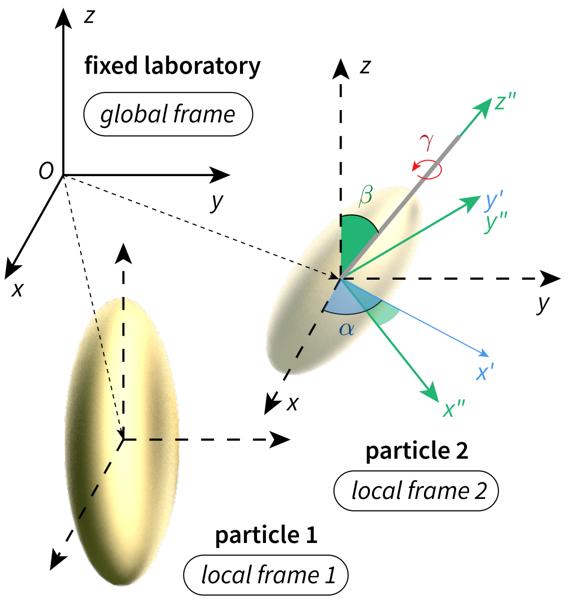

Figure 3: Illustration of local and global reference frames for a cluster of particles and their associated T-matrices.

3.3.1. Rotation

Let $\mathbf{r}^{(1)} = (r, \theta^{(1)}, \varphi^{(1)})$ and $\mathbf{r}^{(2)} = (r, \theta^{(2)}, \varphi^{(2)})$ be the spherical polar coordinates of the same point P in coordinate systems 1 and 2, respectively, sharing the same origin O. If coordinate system 2 is obtained by rotating coordinate system 1 through Euler angles (α, β, γ), here defined in the "zyz" convention[4] with $0 \le \alpha < 2\pi$, $0 \le \beta \le \pi$, and $0 \le \gamma < 2\pi$, then

$$\psi^{(2)}_{nm}(k\mathbf{r}^{(2)}) = \sum_{\mu=-n}^{n} \psi^{(1)}_{n\mu}(k\mathbf{r}^{(1)}) D^n_{\mu m}(\alpha, \beta, \gamma),$$
$$\psi^{(1)}_{nm}(k\mathbf{r}^{(1)}) = \sum_{\mu=-n}^{n} \psi^{(2)}_{n\mu}(k\mathbf{r}^{(2)}) D^n_{\mu m}(-\gamma, -\beta, -\alpha), \tag{16}$$

where $D^n_{\mu m} = e^{-i\mu\alpha} d^n_{\mu m}(\beta) e^{-im\gamma}$ and $d^n_{\mu m}$ are the Wigner D- and d-functions.[4] Conveniently, $\widetilde{\psi}_{nm}$, $\widetilde{\mathbf{M}}_{nm}$, $\widetilde{\mathbf{N}}_{nm}$, $\mathbf{M}_{nm}$ and $\mathbf{N}_{nm}$ transform in exactly the same manner under rotation, so substituting ψ_{nm} by a desired basis function in (16) will give the appropriate expression (see equations (5.23)–(5.24) of Ref. 4 for details). In our notation, $\mathbf{W}^{(2)}(k\mathbf{r}^{(2)})$ in coordinate system 2 is related to $\mathbf{W}^{(1)}(k\mathbf{r}^{(1)})$ in coordinate system 1 via

$$\mathbf{W}^{(2)} = \mathbf{W}^{(1)} \mathbf{R}(\alpha, \beta, \gamma) \tag{17}$$

where $\mathbf{R}(\alpha, \beta, \gamma)$ is a unitary block-diagonal matrix (of size $l_{\max} \times l_{\max}$), satisfying

$$\mathbf{R}^{-1}(\alpha, \beta, \gamma) = \mathbf{R}^\dagger(\alpha, \beta, \gamma) = \mathbf{R}(-\gamma, -\beta, -\alpha) \tag{18}$$

with matrix elements given by

$$R_{ll'}(\alpha, \beta, \gamma) = \delta_{ss'}\delta_{nn'} D^n_{m'm}(\alpha, \beta, \gamma), \tag{19}$$

where the index $l(s, n, m)$ is defined in (11). Note that (17) also applies to regular waves $\widetilde{\mathbf{W}}$. Now, if a (regular or irregular) spherical wave expansion is described by a vector of coefficients $\mathbf{a}^{(1)}$ in coordinate system 1 and by $\mathbf{a}^{(2)}$ in coordinate system 2, then

$$\mathbf{a}^{(2)} = \mathbf{R}^\dagger(\alpha, \beta, \gamma)\mathbf{a}^{(1)}, \tag{20}$$

which follows from equating the field expansions and using (17), i.e.

$$\mathbf{W}^{(1)}\mathbf{a}^{(1)} = \mathbf{W}^{(2)}\mathbf{a}^{(2)} = \mathbf{W}^{(1)}\mathbf{R}(\alpha, \beta, \gamma)\mathbf{a}^{(2)}$$
$$\implies \mathbf{a}^{(1)} = \mathbf{R}(\alpha, \beta, \gamma)\mathbf{a}^{(2)}. \tag{21}$$

Let us re-label coordinate system 1 as G to indicate a global, space-fixed reference frame, and coordinate system 2 as L for local frame, attached to a scatterer. A T-matrix $\mathbf{T}^{(L)}$ expressed in the local frame is transformed into $\mathbf{T}^{(G)} = \mathbf{R}\mathbf{T}^{(L)}\mathbf{R}^\dagger$ in the global frame, where $\mathbf{R}(\alpha, \beta, \gamma)$ depends on the Euler angles (α, β, γ) that rotate frame G onto frame L (as opposed to L onto G). To clarify, consider

$$\begin{aligned}
\mathbf{a}^{(L)} &= \mathbf{T}^{(L)}\widetilde{\mathbf{a}}^{(L)} \\
\mathbf{R}^\dagger\mathbf{a}^{(G)} &= \mathbf{T}^{(L)}\mathbf{R}^\dagger\widetilde{\mathbf{a}}^{(G)} \\
\mathbf{a}^{(G)} &= \underbrace{\mathbf{R}\mathbf{T}^{(L)}\mathbf{R}^\dagger}_{\mathbf{T}^{(G)}}\widetilde{\mathbf{a}}^{(G)} \implies \mathbf{T}^{(G)} = \mathbf{R}\mathbf{T}^{(L)}\mathbf{R}^\dagger.
\end{aligned}$$

If the scatterer is rotationally symmetric about the local z-axis, which is tilted by spherical polar angles (θ, φ) relative to the global z-axis, then $\alpha = \varphi$, $\beta = \theta$, and the value of γ is irrelevant due to axial symmetry, so we can choose $\gamma = 0$ to have $\mathbf{T}^{(G)} = \mathbf{R}(\varphi, \theta, 0)\mathbf{T}^{(L)}\mathbf{R}(0, -\theta, -\varphi)$ (see Sec. 5.2 of Ref. 4 for details).

3.3.2. Translation

Consider a point P with coordinates $\mathbf{r}^{(1)}$ in coordinate system 1 with the origin at O_1. If we choose another origin O_2 displaced by $\mathbf{d}_{12}$ from O_1, then the coordinates of P relative to O_2 will be $\mathbf{r}^{(2)} = \mathbf{r}^{(1)} - \mathbf{d}_{12}$, as illustrated in Fig. 4. The translation-addition theorem for vector spherical waves states that[6,9], in the limit $n_{\max} \to \infty$,

$$\mathbf{W}^{(1)}(k\mathbf{r}^{(1)}) = \begin{cases} \mathbf{W}^{(2)}(k\mathbf{r}^{(2)})\widetilde{\mathbf{O}}(k\mathbf{d}_{12}), & \text{if } r^{(2)} > d_{12}, \\ \widetilde{\mathbf{W}}^{(2)}(k\mathbf{r}^{(2)})\mathbf{O}(k\mathbf{d}_{12}), & \text{if } r^{(2)} < d_{12}, \end{cases} \tag{22}$$

$$\widetilde{\mathbf{W}}^{(1)}(k\mathbf{r}^{(1)}) = \widetilde{\mathbf{W}}^{(2)}(k\mathbf{r}^{(2)})\widetilde{\mathbf{O}}(k\mathbf{d}_{12}), \tag{23}$$

where $\widetilde{\mathbf{O}}(k\mathbf{d}_{12})$ and $\mathbf{O}(k\mathbf{d}_{12})$ are $(l_{\max} \times l_{\max})$ matrices of regular and irregular translation-addition coefficients (TACs), respectively[14]. Note the conditional statement for irregular waves: the transformation depends on the relative length of $\mathbf{r}^{(2)}$ and $\mathbf{d}_{12}$. In Fig. 4, an irregular basis centred at O_1 is mapped onto a regular basis centred at O_2 via the irregular TACs. However, if O_2 were to the left of the bisector, so that $r^{(2)} > d_{12}$, then the irregular basis centred at O_1 would be mapped onto an irregular basis centred at O_2 via the regular TACs.

Note that O_1 and O_2 are themselves points with coordinates $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively, in a common "global" frame with a fixed origin O. If we denote the global frame coordinates of P by $\mathbf{r}$, then $\mathbf{r}^{(1)} = \mathbf{r} - \mathbf{r}_1$, $\mathbf{r}^{(2)} = \mathbf{r} - \mathbf{r}_2$, and $\mathbf{d}_{12} = \mathbf{r}^{(1)} - \mathbf{r}^{(2)} = \mathbf{r}_2 - \mathbf{r}_1 =: \mathbf{r}_{21}$. Henceforth we follow Stout and coworkers[14,15] and adopt the shorthand notation $\widetilde{\mathbf{O}}^{(i,j)} := \widetilde{\mathbf{O}}(k\mathbf{r}_{ij}) = \widetilde{\mathbf{O}}(k\mathbf{d}_{ji})$, and likewise for $\mathbf{O}^{(i,j)}$, yielding

$$\mathbf{W}^{(i)} = \begin{cases} \mathbf{W}^{(j)}\widetilde{\mathbf{O}}^{(j,i)}, & \text{if } r^{(j)} > r_{ij}, \\ \widetilde{\mathbf{W}}^{(j)}\mathbf{O}^{(j,i)}, & \text{if } r^{(j)} < r_{ij}, \end{cases} \tag{24}$$

$$\widetilde{\mathbf{W}}^{(i)} = \widetilde{\mathbf{W}}^{(j)}\widetilde{\mathbf{O}}^{(j,i)}. \tag{25}$$

Note the reversal of indices in $\mathbf{r}_{ij} = \mathbf{d}_{ji}$ and the minus sign in $\mathbf{r}_{ij} = -\mathbf{d}_{ij}$; note that $d_{ij} = d_{ji} = r_{ij} = r_{ji} \geq 0$ in our notations.

To express the translation-addition theorem in terms of the coefficients (the $\mathbf{a}$'s) of a VSWF expansion, multiply (from the right) both sides of equations (24) and (25) by column vector $\mathbf{a}^{(i)}$, where the superscript (i) indicates where the VSWF expansion is centred. From inspection of the right-hand side we find that

$$\mathbf{a}^{(j)} = \widetilde{\mathbf{O}}^{(j,i)}\mathbf{a}^{(i)}, \quad \text{if } r^{(j)} > r_{ij}, \tag{26}$$

$$\widetilde{\mathbf{a}}^{(j)} = \mathbf{O}^{(j,i)}\mathbf{a}^{(i)}, \quad \text{if } r^{(j)} < r_{ij}, \tag{27}$$

$$\widetilde{\mathbf{a}}^{(j)} = \widetilde{\mathbf{O}}^{(j,i)}\widetilde{\mathbf{a}}^{(i)}. \tag{28}$$

Note that (26), (27) and (28) are in a similar matrix-vector form to the rotation equation (20).

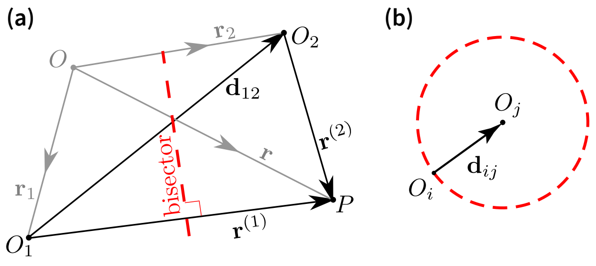

Figure 4: (a) Illustration of how the coordinate vector of point P is transformed from $\mathbf{r}^{(1)}$ to $\mathbf{r}^{(2)}$ when the origin is switched from O_1 to O_2. Dashed red line bisects the O_1P edge of the O_1PO_2 triangle. (b) Illustration of how the singularity at O_i exhibited by $\mathbf{W}(k\mathbf{r}^{(i)})$ is spread over the surface of a ball with radius $d_{ij} = r_{ij}$ after translation to a target origin O_j by displacement vector $\mathbf{d}_{ij}$. The irregular basis remains irregular outside the ball, i.e. $\mathbf{W}(k\mathbf{r}^{(i)}) = \mathbf{W}(k\mathbf{r}^{(j)})\widetilde{\mathbf{O}}^{(j,i)}$ for $r^{(j)} > r_{ij}$, but is transformed into a *regular* basis *inside* the ball, i.e. $\mathbf{W}(k\mathbf{r}^{(i)}) = \widetilde{\mathbf{W}}(k\mathbf{r}^{(j)})\mathbf{O}^{(j,i)}$ for $r^{(j)} < r_{ij}$.

3.3.3. Factorized translation (involving rotation)

A general translation from centre $\mathbf{r}_i$ to another centre $\mathbf{r}_j$ by displacement vector $\mathbf{d}_{ij} = (d_{ij}, \theta_{ij}, \varphi_{ij})$ can be separated into three steps:

1. Rotation of the local frame to align the z-axis with the $\mathbf{d}_{ij}$ vector. In the zyz convention, the appropriate Euler angles are $\alpha = \varphi_{ij}$, $\beta = \theta_{ij}$, and $\gamma = 0$.
2. Axial translation along the rotated local z-axis by d_{ij}.
3. Rotation of the local frame to realign the z-axis with the original orientation. The appropriate Euler angles are $\alpha' = -\gamma = 0$, $\beta' = -\beta = -\theta_{ij}$, and $\gamma' = -\alpha = -\varphi_{ij}$.

This factorisation can be expressed in matrix form as

$$\mathbf{O}^{(j,i)} = \mathbf{R}(\varphi_{ij}, \theta_{ij}, 0)\mathbf{O}_z(d_{ij})\mathbf{R}(0, -\theta_{ij}, -\varphi_{ij}) \tag{29}$$

for the irregular case, where $\mathbf{O}_z(d_{ij})$ represents the matrix of z-axial translation coefficients, many of which are zero due to the special case of axial translation along z. Note that the three aforementioned steps correspond to stepwise movement of the local axis, from the perspective of the initial point i, and the transformation corresponds to reading the matrix multiplication in (29) from left to right; but the sequence of steps and the direction of movement is actually reversed from the perspective of the scatterer at the destination point. More importantly, since all three matrix-factors on the right-hand side of (29) will contain many zeroes, operating on a vector of VSWF coefficients in a sequence of three steps can actually reduce the scaling of the net computational cost from $\sim n_{\max}^4$ to $\sim n_{\max}^3$, when the naïve matrix multiplication on each step is replaced by a custom operation that sums just over the relevant (non-zero) components.

Another potential advantage of using (29) is that, after obtaining the three factors for $\mathbf{O}^{(j,i)}$, they can be recycled when computing the reverse translation $\mathbf{O}^{(i,j)}$ to reduce the overall computational cost. First, beware that $\mathbf{O}_z(-d_{ij})$ is *not* the inverse of $\mathbf{O}_z(d_{ij})$, and note that $\mathbf{O}_z(d_{ij})$ is invariant to interchanging i and j ($d_{ij} = d_{ji} \geq 0$). Actually, $[\mathbf{O}_z(d_{ij})]^{-1} = \mathbf{R}(0,\pi,0)\mathbf{O}_z(d_{ij})\mathbf{R}(0,-\pi,0)$, where $\mathbf{R}(0,\pi,0)$ is block-diagonal and each block is *anti*-diagonal. Second, since $\varphi_{ij} = \pi + \varphi_{ji}$ and $\theta_{ij} = \pi - \theta_{ji}$, $\mathbf{R}(\varphi_{ji},\theta_{ji},0)$ can be calculated from $\mathbf{R}(\varphi_{ij},\theta_{ij},0)$ using the symmetry relation $d^n_{\mu m}(\pi-\theta) = (-1)^{n-m} d^n_{-\mu m}(\theta)$ (see Eq. B.7 in Ref. 4), so that $D^n_{\mu m}(\varphi_{ji},\theta_{ji},0) = (-1)^{n-m+1} D^n_{-\mu m}(\varphi_{ij},\theta_{ij},0)$, where the extra prefactor of -1 comes from multiplying by $e^{-i\pi} = -1$.

Computing $\mathbf{O}^{(i,j)}$ in general is more costly than the combined effort of computing $\mathbf{O}_z$, $\mathbf{R}$, *and* $\mathbf{R}^{-1}$. However, in our experience, performing matrix multiplication of the three factors in the sequence from the right ($\mathbf{R}\mathbf{O}_z\mathbf{R}^{-1}$) does not seem to yield the expected result (matrix $\mathbf{O}^{(i,j)}$), suggesting a numerical instability in this approach.

3.4. *Superposition* T-*matrix for multiple scatterers*

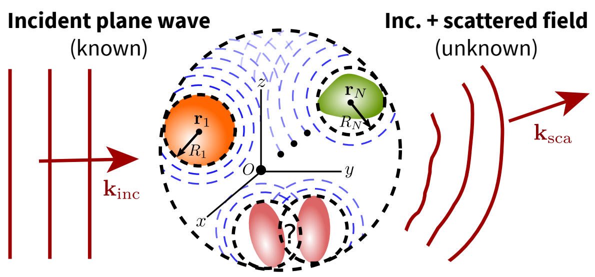

Figure 5: Pictorial representation of light scatttering by a nanoparticle cluster. An incident plane wave with known wavevector ($\mathbf{k}_{\rm inc}$) and incident field ($\mathbf{E}_{\rm inc}$) is scattered by a cluster of N particles centred at $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$. Each particle scatters in response to the net incident field exciting it (partial waves illustrated in dashed blue). The self-consistent total field everywhere in space is the superposition of the incident field ($\mathbf{E}_{\rm inc}$), and of the collectively scattered field ($\mathbf{E}_{\rm sca}$). The scattered field is illustrated by distorted wavefronts outgoing from the cluster.

This section follows the treatments by Stout *et al.*[14,15] as well as similar discussions for multi-sphere clusters by Mackowski & Mishchenko[7,9,16,17].

For a cluster of N scatterers (each of arbitrary shape), the collectively scattered field $\mathbf{E}_{\rm sca}$ may be formally separated into additive contributions from the individuals, namely:

$$\begin{aligned}\mathbf{E}_{\rm sca}(\mathbf{r};k) &= \sum_{j=1}^{N} \mathbf{E}_{{\rm sca},j}(\mathbf{r};k) \\ &= E\sum_{j=1}^{N} \mathbf{W}^{(j)}(k\mathbf{r}^{(j)})\mathbf{c}_j^{(j)} \qquad (30)\end{aligned}$$

where $\mathbf{r}^{(j)} = \mathbf{r} - \mathbf{r}_j$, $\mathbf{r}_j$ is the position of the jth scatterer in the global frame. Note that each partial field contribution $\mathbf{E}_{{\rm sca},j}(\mathbf{r})$ in (30) is developed in terms of irregular waves centred at $\mathbf{r}_j$, as indicated in the superscript of $\mathbf{r}^{(j)}$ and the corresponding coefficients $\mathbf{c}_j^{(j)}$. The centre of expansion need not necessarily be the centre of the particle associated with $\mathbf{E}_{{\rm sca},j}(\mathbf{r})$, so we still keep the subscript j in $\mathbf{c}_j^{(j)}$ as a label specifying the particle centre, which may seem redundant, but keep in mind that $\mathbf{c}_j^{(i)}$ is well defined for $i \neq j$. It is also important to note that (30) does not actually prescribe how exactly the collectively scattered field is partitioned among the individuals. The partitioning is to be determined self-consistently. To set up a self-consistent system of linear equations, it is useful to define the excitation field $\widetilde{\mathbf{E}}_{{\rm exc},j}(\mathbf{r})$ for each scatterer j, and then develop it in terms of regular VSWFs centred at $\mathbf{r}_j$, i.e.

$$\begin{aligned}\widetilde{\mathbf{E}}_{{\rm exc},j}(\mathbf{r}) &:= \widetilde{\mathbf{E}}_{\rm inc}(\mathbf{r}) + \sum_{\substack{l=1\\ l\neq j}}^{N} \mathbf{E}_{{\rm sca},l}(\mathbf{r}) \\ &= E\widetilde{\mathbf{W}}^{(j)}(k\mathbf{r}^{(j)})\widetilde{\mathbf{a}}^{(j)} + E\sum_{\substack{l=1\\ l\neq j}}^{N} \mathbf{W}^{(l)}(k\mathbf{r}^{(l)})\mathbf{c}_l^{(l)} \\ &= E\widetilde{\mathbf{W}}^{(j)}(k\mathbf{r}_j)\left(\widetilde{\mathbf{a}}^{(j)} + \sum_{\substack{l=1\\ l\neq j}}^{N} \underbrace{\mathbf{O}^{(j,l)}\mathbf{c}_l^{(l)}}_{\widetilde{\mathbf{c}}^{(j)}_{j\leftarrow l}}\right) \qquad (31) \\ &\text{for } r^{(j)} < \min\|\mathbf{r}_j - \mathbf{r}_l\| \qquad (32) \\ &=: E\widetilde{\mathbf{W}}^{(j)}(k\mathbf{r}_j)\widetilde{\mathbf{e}}_j^{(j)} \qquad (33)\end{aligned}$$

where $\widetilde{\mathbf{a}}^{(j)} = \widetilde{\mathbf{O}}^{(j,0)}\widetilde{\mathbf{a}}$ contains the incident field coefficients and $\widetilde{\mathbf{O}}^{(j,0)} := \widetilde{\mathbf{O}}(k\mathbf{r}_j)$. In the last equality of (31), $\mathbf{W}^{(l)}$ is transformed into $\widetilde{\mathbf{W}}^{(j)}$, with the corresponding coefficients given by $\widetilde{\mathbf{c}}^{(j)}_{j\leftarrow l} = \mathbf{O}^{(j,l)}\mathbf{c}^{(l)}_l$, where the subscript $j \leftarrow l$ specifies the scattered field partition of scatterer l developed (as a regular VSWF expansion) about particle j. Note the application of the translation-addition theorem clause that applies only inside the ball of radius r_{jl} (for each $l \neq j$) centred at $\mathbf{r}_j$). This approach is strictly valid only if the translation ball fully contains the target scatterer's surface, where the boundary conditions are to be matched. While non-overlapping spherical scatterers are always guaranteed to satisfy this condition, elongated particles such as spheroids can be problematic, because the singularity sphere can cross the target scatterer's surface if it is sufficiently close (yet still not overlapping). These aspects are discussed in more details in Ref. 23 (and references therein).

A self-consistent system of linear equations can now be obtained by requiring that

$$\mathbf{c}^{(j)}_j = \mathbf{T}_j\widetilde{\mathbf{e}}^{(j)}_j, \tag{34}$$

where $\mathbf{T}_j$ is the "one-body" T-matrix characterising scatterer j, as defined above in Sec. 3.2. Note that (34) reduces to (15) for a single scatterer ($j = N = 1$) at the origin, because then $\widetilde{\mathbf{e}}^{(j)}_j \mapsto \widetilde{\mathbf{a}}$, $\mathbf{c}^{(j)}_j \mapsto \mathbf{a}$, and $\mathbf{T}_j \mapsto \mathbf{T}$. From (31), (33), and (34) we obtain an equation expressed just in terms of the field coefficients:

$$\widetilde{\mathbf{e}}^{(j)}_j = \widetilde{\mathbf{a}}^{(j)} + \sum_{\substack{i=1\\ i\neq j}}^{N} \mathbf{O}^{(j,i)}\mathbf{T}_i\widetilde{\mathbf{e}}^{(i)}_i, \tag{35}$$

which Stout *et al.* label as "the fundamental multiple scattering equation" (Eq. 9 in Ref. 14). It is helpful to rewrite the linear system (35) in block-matrix form:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{O}^{(1,2)}\mathbf{T}_2 & \cdots & -\mathbf{O}^{(1,N)}\mathbf{T}_N \\ -\mathbf{O}^{(2,1)}\mathbf{T}_1 & \mathbf{I} & \cdots & -\mathbf{O}^{(2,N)}\mathbf{T}_N \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{O}^{(N,1)}\mathbf{T}_1 & -\mathbf{O}^{(N,2)}\mathbf{T}_2 & \cdots & \mathbf{I} \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{e}}^{(1)}_1 \\ \widetilde{\mathbf{e}}^{(2)}_2 \\ \vdots \\ \widetilde{\mathbf{e}}^{(N)}_N \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{a}}^{(1)} \\ \widetilde{\mathbf{a}}^{(2)} \\ \vdots \\ \widetilde{\mathbf{a}}^{(N)} \end{pmatrix}, \tag{36}$$

which is in the standard form $\mathbf{A}\mathbf{x} = \mathbf{b}$, with $\mathbf{x}$ the unknown, and $\mathbf{b}$ a known input source. The solution $\mathbf{x}$ gives all the $\widetilde{\mathbf{e}}^{(i)}_i$'s for a given $\widetilde{\mathbf{a}}$ and $\mathbf{T}_i$ ($i = 1, \ldots, N$), from which we can determine all the $\widetilde{\mathbf{c}}^{(i)}_i$'s using (34). Alternatively, we can use (34) to substitute the excitation field coefficients for the scattered field coefficients and, assuming the one-body T-matrices are invertible, obtain

$$\mathbf{T}^{-1}_j\mathbf{c}^{(j)}_j - \sum_{\substack{i=1\\ i\neq j}}^{N} \mathbf{O}^{(j,i)}\mathbf{c}^{(i)}_i = \widetilde{\mathbf{a}}^{(j)}, \tag{37}$$

$$\begin{bmatrix} \mathbf{T}^{-1}_1 & -\mathbf{O}^{(1,2)} & \cdots & -\mathbf{O}^{(1,N)} \\ -\mathbf{O}^{(2,1)} & \mathbf{T}^{-1}_2 & \cdots & -\mathbf{O}^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{O}^{(N,1)} & -\mathbf{O}^{(N,2)} & \cdots & \mathbf{T}^{-1}_N \end{bmatrix} \begin{pmatrix} \mathbf{c}^{(1)}_1 \\ \mathbf{c}^{(2)}_2 \\ \vdots \\ \mathbf{c}^{(N)}_N \end{pmatrix} = \begin{pmatrix} \widetilde{\mathbf{a}}^{(1)} \\ \widetilde{\mathbf{a}}^{(2)} \\ \vdots \\ \widetilde{\mathbf{a}}^{(N)} \end{pmatrix}. \tag{38}$$

To avoid involving the matrix inverses $\mathbf{T}^{-1}_j$, we can also rearrange the linear system into

$$\mathbf{c}^{(j)}_j - \mathbf{T}_j\sum_{\substack{i=1\\ i\neq j}}^{N} \mathbf{O}^{(j,i)}\mathbf{c}^{(i)}_i = \mathbf{T}_j\widetilde{\mathbf{a}}^{(j)}, \tag{39}$$

$$\begin{bmatrix} \mathbf{I} & -\mathbf{T}_1\mathbf{O}^{(1,2)} & \cdots & -\mathbf{T}_1\mathbf{O}^{(1,N)} \\ -\mathbf{T}_2\mathbf{O}^{(2,1)} & \mathbf{I} & \cdots & -\mathbf{T}_2\mathbf{O}^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{T}_N\mathbf{O}^{(N,1)} & -\mathbf{T}_N\mathbf{O}^{(N,2)} & \cdots & \mathbf{I} \end{bmatrix} \begin{pmatrix} \mathbf{c}_1^{(1)} \\ \mathbf{c}_2^{(2)} \\ \vdots \\ \mathbf{c}_N^{(N)} \end{pmatrix} = \begin{pmatrix} \mathbf{a}_1^{(1)} \\ \mathbf{a}_2^{(2)} \\ \vdots \\ \mathbf{a}_N^{(N)} \end{pmatrix}, \tag{40}$$

where $\mathbf{a}_j^{(j)} = \mathbf{T}_j\widetilde{\mathbf{a}}^{(j)}$ corresponds to irregular series coefficients for the scattered field of particle j in isolation (from all the other $N-1$ particles).[2]

Note that (36), (38), and (40) are all in the form of a general matrix equation $\mathbf{Ax} = \mathbf{b}$, which can be solved for the column vector(s) $\mathbf{x}$ without inverting the matrix $\mathbf{A}$. However, formal inversion is necessary when seeking *collective* T-matrix constructions, which describe the entire cluster's response to arbitrary incident fields, and can notably provide analytical formulas for orientation-averaged quantities. A number of collective T-matrices are defined in the next section, following the different treatments of Stout and Mackowski & Mishchenko.

3.5. Collective T*-matrix constructions*

The system of coupled matrix equations in (38) can be solved for $\mathbf{c}_j^{(j)}$ by inverting the matrix to obtain

$$\mathbf{c}_j^{(j)} = \sum_{i=1}^{N} \mathbf{T}^{(j,i)}\widetilde{\mathbf{a}}^{(i)} = \underbrace{\left(\sum_{i=1}^{N} \mathbf{T}^{(j,i)}\widetilde{\mathbf{O}}^{(i,0)}\right)}_{:=\text{Mackowski's } \mathbf{T}_M^{(j)}} \widetilde{\mathbf{a}} \tag{41}$$

$$= \underbrace{\left(\sum_{i=1}^{N} \mathbf{T}^{(j,i)}\widetilde{\mathbf{O}}^{(i,j)}\right)}_{:=\text{Stout's } \mathbf{T}_S^{(j)}} \widetilde{\mathbf{a}}^{(j)}, \tag{42}$$

where $\mathbf{T}^{(j,i)}$ represent what we may call "pairwise T-matrices", expressing the portion of the scattered field from particle j in response to its excitation by particle i; the $\mathbf{T}^{(j,i)}$ matrices are arranged and defined as follows

$$\left[\mathbf{T}^{(j,i)}\right] = \begin{bmatrix} \mathbf{T}^{(1,1)} & \mathbf{T}^{(1,2)} & \cdots & \mathbf{T}^{(1,N)} \\ \mathbf{T}^{(2,1)} & \mathbf{T}^{(2,2)} & \cdots & \mathbf{T}^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}^{(N,1)} & \mathbf{T}^{(N,2)} & \cdots & \mathbf{T}^{(N,N)} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_1^{-1} & -\mathbf{O}^{(1,2)} & \cdots & -\mathbf{O}^{(1,N)} \\ -\mathbf{O}^{(2,1)} & \mathbf{T}_2^{-1} & \cdots & -\mathbf{O}^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{O}^{(N,1)} & -\mathbf{O}^{(N,2)} & \cdots & \mathbf{T}_N^{-1} \end{bmatrix}^{-1} \tag{43}$$

$$= \begin{bmatrix} \mathbf{T}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{T}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{T}_N \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{O}^{(1,2)}\mathbf{T}_2 & \cdots & -\mathbf{O}^{(1,N)}\mathbf{T}_N \\ -\mathbf{O}^{(2,1)}\mathbf{T}_1 & \mathbf{I} & \cdots & -\mathbf{O}^{(2,N)}\mathbf{T}_N \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{O}^{(N,1)}\mathbf{T}_1 & -\mathbf{O}^{(N,2)}\mathbf{T}_2 & \cdots & \mathbf{I} \end{bmatrix}^{-1} \tag{44}$$

$$= \begin{bmatrix} \mathbf{I} & -\mathbf{T}_1\mathbf{O}^{(1,2)} & \cdots & -\mathbf{T}_1\mathbf{O}^{(1,N)} \\ -\mathbf{T}_2\mathbf{O}^{(2,1)} & \mathbf{I} & \cdots & -\mathbf{T}_2\mathbf{O}^{(2,N)} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{T}_N\mathbf{O}^{(N,1)} & -\mathbf{T}_N\mathbf{O}^{(N,2)} & \cdots & \mathbf{I} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{T}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{T}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{T}_N \end{bmatrix} \tag{45}$$

[2]Note that (39) is equivalent to Mackowski's Eq. 4 of Ref. 18, though in Mackowski & Mishchenko's earlier papers the same equation (Eq. 13 of Ref. 16 and Eq. 3 of Ref. 17) has a plus sign instead of the minus, which may be entirely due to a minus sign featuring in the incident field expansion (see Eq. 4 in Ref. 16). This minus sign is absent in Eq. 2 of the more recent Ref. 18, and the subsequent Eq. 4 matches our equation (39).

with the last two lines merely showing how the one-body T-matrices can be factored out in two different ways (left or right).

The $\mathbf{T}^{(j,i)}$ matrices provide a complete and exact solution to the multiple scattering problem. Crucially, they retain all the information required to calculate fields at any point within or outside the cluster (except within the Rayleigh Hypothesis region for nonspherical scatterers). Stout *et al.* denotes these matrices "scatterer-centred transfer matrices" (see Eq. 12 in Ref. 14), while Mackowski & Mishchenko refer to them as "sphere-centred" (see Eq. 16 in Ref. 16 and Eq. 4 in Ref. 17). Arguably, both appellations are equally applicable to $\mathbf{T}^{(j)}$, which Mackowski & Mishchenko define one way (see Eq. 61 in Ref. 17) without giving a particular name, while Stout *et al.* define $\mathbf{T}^{(j)}$ differently and call it "individual N-body transfer matrices" (see equations. 14 and 27 in Ref. 14). The difference between both is explicitly stated in Eq. (41), with Stout's $\mathbf{T}_S^{(j)}$ retaining expansions from each scatterer's origin, and Mackowski & Mishchenko's $\mathbf{T}_M^{(j)}$ collapsing all expansions to a common origin O. Note that neither definition should be confused with the one-body T-matrices $\mathbf{T}_j$ of equation (34).

Mackowski & Mishchenko additionally consider the collective scattering coefficients $\mathbf{a}$ for the irregular VSWF expansion about the common origin of the whole cluster, i.e.

$$\begin{aligned}\mathbf{E}_{\text{sca}}(\mathbf{r};k) =& E\mathbf{W}(k\mathbf{r})\mathbf{a} \\ =& E\mathbf{W}(k\mathbf{r})\left(\sum_{j=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\mathbf{c}^{(j)}\right), \\ & \text{for}\|\mathbf{r}\| > \max\|\mathbf{r}_j\|,\end{aligned} \tag{46}$$

where the second equality relies on a particular clause of the translation-addition theorem, which is valid only outside of the smallest circumscribed sphere (encompassing all N scatterers) centred at the global frame's origin. From (46) and (41) we have

$$\begin{aligned}\mathbf{a} &= \sum_{j=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\mathbf{c}_j^{(j)} \\ &= \sum_{j=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\sum_{i=1}^{N}\mathbf{T}^{(j,i)}\widetilde{\mathbf{a}}^{(i)} \\ &= \sum_{j=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\mathbf{T}_M^{(j)}\widetilde{\mathbf{a}},\end{aligned} \tag{47}$$

where $\mathbf{a}$ and $\widetilde{\mathbf{a}}$ are now related as in (15), providing expressions for the collective T-matrix of the entire cluster

$$\begin{aligned}\mathbf{T} &= \sum_{j=1}^{N}\sum_{i=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\mathbf{T}^{(j,i)}\widetilde{\mathbf{O}}^{(i,0)} \\ &= \sum_{j=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\mathbf{T}_M^{(j)} \\ &= \sum_{j=1}^{N}\widetilde{\mathbf{O}}^{(0,j)}\mathbf{T}_S^{(j)}\widetilde{\mathbf{O}}^{(j,0)},\end{aligned} \tag{48}$$

where in the last equality we used the fact that $\mathbf{T}_M^{(j)} = \mathbf{T}_S^{(j)}\widetilde{\mathbf{O}}^{(j,0)}$. Mackowski & Mishchenko refer to the collective $\mathbf{T}$ in (48) as the "cluster-centred" T-matrix (see Eq. 19 in Ref. 16, Eq. 64 in Ref. 17, and Eq. 29 in Ref. 18). Note that (47) is valid only outside of the cluster's smallest circumscribed sphere centred at the common origin; this collective T-matrix does not allow the calculation of near-fields between particles.[17,18]

In the TERMS program, when `Scheme` $\neq 0$ the collective T-matrix is calculated in the subroutine `contractTmat` of the `multiscat` module; it is used to calculate orientation-averaged far-field cross-sections. However, if the keyword `ScattererCentredCrossSections` is included in the input file, the collective $\mathbf{T}$ will not be calculated and the program will be using alternative orientation-averaging formulas based on particle-centred T-matrices $\mathbf{T}^{(i,j)}$ instead.

3.6. Far-field cross-sections

In `Mode` $= 2$ the program calculates far-field cross-sections for the given incident field direction(s) and four polarisations (two linear, two circular), as well as their average over the full solid angle using analytical formulas.

3.6.1. Fixed orientation cross-sections

After solving for the particle-centred coefficients $\mathbf{c}_j^{(j)}$ for a given $\widetilde{\mathbf{a}}$ and $\mathbf{T}_j$'s (where $j = 1, \ldots, N$), the corresponding fixed-orientation extinction (σ_{ext}), scattering (σ_{sca}), and absorption (σ_{abs}) cross-sections can be calculated. Here we state formulae for σ_{ext} and σ_{sca} expressed in terms of origin- and particle-centred coefficients, without derivation but with references to the previously-cited literature. For a non-absorbing incident medium (real-valued wavenumber k, as must be the case throughout TERMS), σ_{abs} can be inferred using energy conservation: $\sigma_{\text{abs}} = \sigma_{\text{ext}} - \sigma_{\text{sca}}$.

Fixed-orientation extinction cross-sections are calculated using

$$\sigma_{\text{ext}} = -\frac{\Re\{\widetilde{\mathbf{a}}^\dagger \mathbf{a}\}}{k^2} = -\frac{1}{k^2}\sum_{l=1}^{l_{\max}} \Re\left\{\widetilde{a}_l^* a_l\right\} \tag{49}$$

$$= -\frac{1}{k^2}\Re\left\{\sum_{j=1}^{N} \widetilde{\mathbf{a}}^{(j)\dagger}\mathbf{c}^{(j)}\right\} \tag{50}$$

$$= -\frac{1}{k^2}\sum_{j=1}^{N}\sum_{l=1}^{l_{\max}} \Re\left\{\widetilde{a}_l^{(j)*} c_l^{(j)}\right\} \tag{51}$$

$$=: \sum_{j=1}^{N} \sigma_{\text{ext}}^{(j)}, \tag{52}$$

where $\Re\{\dots\}$ indicates taking the real part of the quantity inside the braces, $\widetilde{\mathbf{a}}^{(j)\dagger}$ is conjugate transpose of the column vector $\widetilde{\mathbf{a}}^{(j)}$, $\widetilde{a}_l^{(j)*}$ is the complex conjugate of the vector component $a_l^{(j)}$, and k is the wavenumber in the incident medium. Note that, since $\Re\{ab^*\} = \Re\{a^*b\}$ for any complex numbers a and b, (49) is equivalent to Eq. H.65 of Ref. 19 and Eq. 5.18a of Ref. 4 (provided $|\mathbf{E}_0^{\text{inc}}|^2 = E = 1$). Equation (51) is taken from Eq. 29 of Ref. 15, which follows from simplifying Eq. 43 of Ref. 14 and substituting into Eq. 42 of the same reference. Also note that σ_{ext} is (formally) separable into additive contributions from individual scatterers, i.e. $\sigma_{\text{ext}} = \sum_j \sigma_{\text{ext}}^{(j)}$.

Fixed-orientation scattering cross-sections can be calculated using

$$\sigma_{\text{sca}} = \frac{|\mathbf{a}|^2}{k^2} = \frac{1}{k^2}\sum_{l=1}^{l_{\max}} a_l^* a_l \tag{53}$$

$$= \frac{1}{k^2}\sum_{j=1}^{N}\sum_{i=1}^{N} \mathbf{a}^{(j)\dagger}\widetilde{\mathbf{O}}^{(j,i)}\mathbf{a}^{(i)} \tag{54}$$

$$= \frac{1}{k^2}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{l=1}^{l_{\max}}\sum_{l'=1}^{l_{\max}} a_l^{(j)*}\widetilde{O}_{ll'}^{(j,i)} a_{l'}^{(i)} \tag{55}$$

$$=: \sum_{j=1}^{N} \sigma_{\text{sca}}^{(j)}, \tag{56}$$

where $|\mathbf{a}|^2 = \mathbf{a}^\dagger\mathbf{a}$. Note that (53) is equivalent to Eq. H.64 of Ref. 19 and Eq. 5.18b of Ref. 4 (provided $|\mathbf{E}_0^{\text{inc}}|^2 = E = 1$); while (55) is from Eq. 29 of Ref. 15, which follows from substituting Eq. 45 of Ref. 14 into Eq. 42 of the same reference.

In TERMS, with `Mode` = 2 all the far-field cross-sections are calculated in the main subroutine `spectrumFF`. If the keyword `ScattererCentredCrossSections` is included in the input file, fixed orientation cross-sections, are calculated using particle-centred coefficients via the subroutine `calcCsStout` (equations 51 and 55 are implemented in this subroutine). Otherwise, they are calculated using origin-centred coefficients via the subroutine `calcCs`.

3.6.2. Orientation averaged cross-sections

One attractive feature of the T-matrix method is that it provides relatively simple means of calculating orientation-averaged cross-sections, herein denoted by $\langle\sigma_{\text{ext}}\rangle$, $\langle\sigma_{\text{sca}}\rangle$ and $\langle\sigma_{\text{abs}}\rangle$; these are often used to describe a randomly oriented scatterer, or, equivalently, light incident from a random direction[56]. Note that in TERMS the orientation averaging applies to the cluster as a whole, not to individual particles within the cluster: they are considered rigidly held together (attached on a template, in practice). As with the fixed-orientation cross-sections, orientation averages can be calculated either from the origin-centred collective T-matrix $\mathbf{T}$, or from the particle-centred T-matrices $\mathbf{T}^{(i,j)}$.

$$\langle\sigma_{\text{ext}}\rangle = -\frac{2\pi}{k^2}\Re\left\{\text{Tr}(\mathbf{T})\right\} \tag{57}$$

$$= -\frac{2\pi}{k^2}\sum_j\sum_k \Re\left\{\text{Tr}\left(\mathbf{T}^{(j,k)}\widetilde{\mathbf{O}}^{(k,j)}\right)\right\} \tag{58}$$

$$=: \sum_j \langle\sigma_{\text{ext},j}\rangle \tag{59}$$

$$\langle\sigma_{\text{sca}}\rangle = \frac{2\pi}{k^2}\,\text{Tr}\left(\mathbf{T}^\dagger\mathbf{T}\right) \tag{60}$$

$$= \frac{2\pi}{k^2}\sum_j\sum_k \text{Tr}\left(\left[\sum_l \widetilde{\mathbf{O}}^{(k,l)}\mathbf{T}^{(l,j)}\right]^\dagger \left[\sum_i \mathbf{T}^{(k,i)}\widetilde{\mathbf{O}}^{(i,j)}\right]\right) \tag{61}$$

$$=: \sum_j \langle\sigma_{\text{sca},j}\rangle \tag{62}$$

The cluster's absorption cross-section can be calculated from energy conservation,

$$\langle\sigma_{\text{abs}}\rangle = \langle\sigma_{\text{ext}}\rangle - \langle\sigma_{\text{sca}}\rangle. \tag{63}$$

Alternatively, it may also be calculated directly from the flux of the Poynting vector of the total field entering the surface of each individual particle. This provides the physically-meaningful portion of energy absorbed within each scatterer j, and their sum adds up to the total absorption cross-section for the cluster. Following Stout[14] and restricting ourselves to non-magnetic, homogeneous spheres,

$$\langle\sigma_{\text{abs}}\rangle = \frac{2\pi}{k^2}\sum_j\sum_k\sum_l \operatorname{Tr}\left(\left[\mathbf{T}^{(j,k)}\right]^\dagger \Gamma^j \mathbf{T}^{(j,l)}\widetilde{\mathbf{O}}^{(l,k)}\right) \tag{64}$$

$$=: \sum_j \langle\sigma_{\text{abs},j}\rangle \tag{65}$$

where the absorption matrix Γ^j is of the form

$$\Gamma^j = \begin{bmatrix} C^j & 0 \\ 0 & D^j \end{bmatrix}. \tag{66}$$

C^j and D^j are diagonal matrices with matrix elements

$$C^j_n = \frac{\Re\left[i\rho_j\psi_n^*(\rho_j\chi_j)\psi_n'(\rho_j\chi_j)\right]}{|\psi_n(\rho_j\chi_j)\psi_n'(\chi_j) - \rho_j\psi_n'(\rho_j\chi_j)\psi_n(\chi_j)|^2} \tag{67}$$

$$D^j_n = \frac{\Re\left[i\rho_j^*\psi_n^*(\rho_j\chi_j)\psi_n'(\rho_j\chi_j)\right]}{|\rho_j\psi_n(\rho_j\chi_j)\psi_n'(\chi_j) - \psi_n'(\rho_j\chi_j)\psi_n(\chi_j)|^2} \tag{68}$$

where $\psi_n(x)$ are Ricatti-Bessel functions: $\psi_n(x) = xj_n(x)$, $\chi_j = kR_j$, and $\rho_j = k_j/k$. k is the wavenumber in the incident medium, R_j the radius of sphere j, and k_j the wavenumber inside (homogeneous) sphere j.

When `Scheme` $=$ 1 or 2 and the input file requests `ScattererCentredCrossSections`, the orientation-averaged cross-sections are calculated in the subroutine `calcOaStout` which uses particle-centred T-matrices $\mathbf{T}^{(i,j)}$ (Eqs. 58, 61, 64). Per-particle orientation-averaged absorption is currently only returned for homogeneous spheres, as the generalisation of Eq. 64 to arbitrary scatterers is not yet available.

In other cases, orientation-averaged cross-sections are calculated in the subroutine `calcOAprops`, which uses the common-origin collective T-matrix $\mathbf{T}$ (Eqs. 57, 60, 63). Note that these calculations based on collective $\mathbf{T}$ are much faster than those based on particle-centred T-matrices $\mathbf{T}^{(i,j)}$.

3.6.3. Circular dichroism

Circular dichroism (CD) is defined as the difference between the optical properties of the structure under left and right circularly polarised incident fields. Its calculation is more natural when VSWFs are expressed in the "helicity" basis, related to the standard "parity" (TE, TM) basis via the helicity operator ($\Lambda = \frac{\nabla\times}{k}$) leading to[21],

$$\mathbf{Z}_{R,nm} = \frac{1}{\sqrt{2}}(\mathbf{M}_{nm} - \mathbf{N}_{nm}), \quad \Lambda\mathbf{Z}_{R,nm} = -\mathbf{Z}_{R,nm} \tag{69}$$

$$\mathbf{Z}_{L,nm} = \frac{1}{\sqrt{2}}(\mathbf{M}_{nm} + \mathbf{N}_{nm}), \quad \Lambda\mathbf{Z}_{L,nm} = \mathbf{Z}_{L,nm}, \tag{70}$$

where the subscripts (R) and (L) refer to right and left circularly polarised light. The corresponding T-matrix describes the scattering of circularly-polarised incident fields in the helicity basis.

Using these definitions the relation between the T-matrix blocks in parity and helicity bases reads

$$\begin{bmatrix} T_{LL} & T_{LR} \\ T_{RL} & T_{RR} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} I & I \\ I & -I \end{bmatrix}\begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}\begin{bmatrix} I & I \\ I & -I \end{bmatrix}, \tag{71}$$

where I is the identity matrix with the same size as the 4 matrix blocks (T_{11}, etc.). The orientation averaged cross-sections for a specific (L) or (R) polarisation can be obtained from Eqs. (59), (62), by restricting the coefficients of the incident field to one helicity,[21]

$$\langle\sigma_{\text{sca}}\rangle_L = \frac{4\pi}{k^2}\operatorname{Tr}\left(\mathbf{T}_{LL}^\dagger\mathbf{T}_{LL} + \mathbf{T}_{RL}^\dagger\mathbf{T}_{RL}\right) \tag{72}$$

$$\langle\sigma_{\text{ext}}\rangle_L = \frac{4\pi}{k^2}\Re\left[\operatorname{Tr}\left(\mathbf{T}_{LL}\right)\right] \tag{73}$$

$$\langle\sigma_{\text{abs}}\rangle_L = \frac{4\pi}{k^2}\Re\left[\operatorname{Tr}\left(\mathbf{T}_{LL}(\mathbf{I} - \mathbf{T}_{LL}^\dagger) - \mathbf{T}_{RL}^\dagger\mathbf{T}_{RL}\right)\right] \tag{74}$$

with simple changes $L \leftrightarrow R$ for R polarisation. Circular dichroism is then obtained as the difference between (L) and (R) cross-sections.

The subroutine `calcOAprops` implements these formulas, calculated for `Scheme` $\neq 0$.

3.6.4. Stokes scattering vector and phase matrix

Some light scattering applications require characterising the angular and polarisation characteristics of the scattered field for a specified incident plane wave. TERMS uses the Stokes vector formalism to describe such situations in `Mode` = 3, following Ref. 4. From the incident electric field $\mathbf{E}_0$, the components of the incident Stokes vector read

$$\mathbf{I} = \begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix} = \frac{1}{2}\sqrt{\frac{\varepsilon}{\mu}} \begin{bmatrix} E_{0\theta}E_{0\theta}^* + E_{0\varphi}E_{0\varphi}^* \\ E_{0\theta}E_{0\theta}^* - E_{0\varphi}E_{0\varphi}^* \\ -2\Re(E_{0\theta}E_{0\varphi}^*) \\ 2\Im(E_{0\theta}E_{0\varphi}^*) \end{bmatrix} \tag{75}$$

The 4×4 phase matrix $\mathbf{Z}$ relates incident and scattered field Stokes vectors, with the following expressions,

$$Z_{11} = \tfrac{1}{2}(|S_{11}|^2 + |S_{12}|^2 + |S_{21}|^2 + |S_{22}|^2) \tag{76}$$
$$Z_{12} = \tfrac{1}{2}(|S_{11}|^2 - |S_{12}|^2 + |S_{21}|^2 - |S_{22}|^2) \tag{77}$$
$$Z_{13} = -\Re(S_{11}S_{12}^* + S_{22}S_{21}^*) \tag{78}$$
$$Z_{14} = -\Im(S_{11}S_{12}^* - S_{22}S_{21}^*) \tag{79}$$
$$Z_{21} = \tfrac{1}{2}(|S_{11}|^2 + |S_{12}|^2 - |S_{21}|^2 - |S_{22}|^2) \tag{80}$$
$$Z_{22} = \tfrac{1}{2}(|S_{11}|^2 - |S_{12}|^2 - |S_{21}|^2 + |S_{22}|^2) \tag{81}$$
$$Z_{23} = -\Re(S_{11}S_{12}^* - S_{22}S_{21}^*) \tag{82}$$
$$Z_{24} = -\Im(S_{11}S_{12}^* + S_{22}S_{21}^*) \tag{83}$$
$$Z_{31} = -\Re(S_{11}S_{21}^* + S_{22}S_{12}^*) \tag{84}$$
$$Z_{32} = -\Re(S_{11}S_{21}^* - S_{22}S_{12}^*) \tag{85}$$
$$Z_{33} = \Re(S_{11}S_{22}^* + S_{12}S_{21}^*) \tag{86}$$
$$Z_{34} = \Im(S_{11}S_{22}^* + S_{21}S_{12}^*) \tag{87}$$
$$Z_{41} = -\Im(S_{21}S_{11}^* + S_{22}S_{12}^*) \tag{88}$$
$$Z_{42} = -\Im(S_{21}S_{11}^* - S_{22}S_{12}^*) \tag{89}$$
$$Z_{43} = \Im(S_{22}S_{11}^* - S_{12}S_{21}^*) \tag{90}$$
$$Z_{44} = \Re(S_{22}S_{11}^* - S_{12}S_{21}^*) \tag{91}$$

where $\mathbf{S}$ is the standard 2×2 amplitude scattering matrix linking incident and scattered transverse field vectors in the respective directions $(\hat{\mathbf{r}}^{\text{inc}})$ and $(\hat{\mathbf{r}}^{\text{sca}})$,[4]

$$\begin{bmatrix} E_\theta^{\text{sca}}\ (\hat{\mathbf{r}}^{\text{sca}}) \\ E_\varphi^{\text{sca}}\ (\hat{\mathbf{r}}^{\text{sca}}) \end{bmatrix} = \frac{\exp(\mathrm{i}kr)}{r}\mathbf{S}\left(\hat{\mathbf{r}}^{\text{sca}}, \hat{\mathbf{r}}^{\text{inc}}\right) \begin{bmatrix} E_{0\theta}^{\text{inc}} \\ E_{0\varphi}^{\text{inc}} \end{bmatrix}. \tag{92}$$

The amplitude scattering matrix $\mathbf{S}$ is derived from the collective T-matrix following Ref. 4 (Eqs. 5.277–5.280).

3.6.5. Differential scattering cross-section

The differential scattering cross-section describes the angular distribution of the scattered light. It depends on the polarisation of the incident wave as well as the incidence and scattering directions, and is readily calculated from the Stokes phase matrix and incident Stokes vector[4]

$$\begin{aligned}\frac{dC_{\text{sca}}}{d\Omega} = \frac{1}{I_{\text{inc}}}[&Z_{11}(\hat{\mathbf{r}}, \hat{\mathbf{n}}_{\text{inc}})I_{\text{inc}} + Z_{12}(\hat{\mathbf{r}}, \hat{\mathbf{n}}_{\text{inc}})Q_{\text{inc}} \\ &+ Z_{13}(\hat{\mathbf{r}}, \hat{\mathbf{n}}_{\text{inc}})U_{\text{inc}} + Z_{14}(\hat{\mathbf{r}}, \hat{\mathbf{n}}_{\text{inc}})V_{\text{inc}}].\end{aligned} \tag{93}$$

3.7. Near-field quantities

Solving the linear system of multiple-scattering equations provides the scattered field coefficients, from which we can compute the complex vector fields $\mathbf{E}, \mathbf{B}$ everywhere in space, as well as derived quantities such as $|\mathbf{E}|^2, |\mathbf{E}|^4$, or the local degree of optical chirality $\mathscr{C} \propto \Im(\mathbf{E}^* \cdot \mathbf{B})$. If only specific directions of incidence are needed, the system may be solved directly, without inversion (`Scheme` = 0, fastest). However, in some circumstances, such as the description of randomly-oriented clusters, we also seek orientation-averaged near-field quantities, requiring `Scheme` > 0.

Near-field values are calculated in `Mode` = 1 in TERMS, with `mapNF` the main subroutine which receives input values and dispatches to other subroutines for the calculation of specific near-field quantities.

3.7.1. Orientation averaged local field intensity

Following Ref. 53, the local field intensity expressed in terms of the scatterer-centred T-matrices $\mathbf{T}^{(i,j)}$ can be averaged over all possible directions of light incidence, yielding

$$\langle|\mathbf{E}_{\text{tot}}(k\mathbf{r})|^2\rangle = 2\pi E^2\,(A_0 + B_0 + C_0) \tag{94}$$

where

$$A_0 = 1/2\pi, \tag{95}$$

$$B_0 = 2\Re\sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\widetilde{\mathbf{W}}^\dagger(\mathbf{r}_l)\mathbf{P}(\hat{\mathbf{r}}_l, \hat{\mathbf{r}}_j)\mathbf{W}(\mathbf{r}_j)\mathbf{T}_N^{(j,l)}\right), \tag{96}$$

$$C_0 = \mathrm{Tr}\left(\sum_{j=1}^{N}\sum_{l=1}^{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\mathbf{O}^{(l,k)}\mathbf{T}_N^{\dagger(i,k)}\mathbf{W}^\dagger(\mathbf{r}_i)\right. \tag{97}$$
$$\left.\mathbf{P}(\hat{\mathbf{r}}_i, \hat{\mathbf{r}}_j)\mathbf{W}(\mathbf{r}_j)\mathbf{T}_N^{(j,l)}\right)$$

where $\mathbf{P}(\hat{\mathbf{r}}_i, \hat{\mathbf{r}}_j) = C^t(\hat{\mathbf{r}}_i)C(\hat{\mathbf{r}}_j)$ and $C(\hat{\mathbf{r}}_j)$ transforms vector in the jth particle spherical coordinate basis to the cartesian coordinate system:

$$C(\hat{\mathbf{r}}_j) = \begin{bmatrix} \sin\theta_j\cos\varphi_j & \cos\theta_j\cos\varphi_j & -\sin\varphi_j \\ \sin\theta_j\sin\varphi_j & \cos\theta_j\sin\varphi_j & \cos\varphi_j \\ \cos\theta_j & -\sin\theta_j & 0 \end{bmatrix}. \tag{98}$$

The terms A_0, B_0, C_0 correspond to the incident electric field, the interference between incident and scattered electric field, and the scattered electric field, respectively.

The orientation average of the local field intensity is mainly calculated in the subroutine `calcOaExtField` of the `multiscat` module.

3.7.2. Optical chirality ($\mathscr{C}$)

In order to evaluate the local degree of optical chirality ($\mathscr{C}$), the total electric $\mathbf{E} = \mathbf{E}_{\text{sca}} + \mathbf{E}_{\text{inc}}$ and magnetic $\mathbf{B} = \mathbf{B}_{\text{sca}} + \mathbf{B}_{\text{inc}}$ vector fields are first evaluated at the requested position, from which $\mathscr{C}$ is obtained as[57]

$$\mathscr{C} = \frac{-\omega\varepsilon_0}{2}\Im(\mathbf{E}^*.\mathbf{B}) \tag{99}$$

From the Maxwell equation $\mathbf{B} = -i\omega^{-1}\nabla\times\mathbf{E}$, the magnetic field is expressed in the VSWF basis with the same coefficients as the electric field (with a simple swap and a prefactor),

$$\mathbf{B}(\mathbf{r};k) = \frac{-ik}{\omega}\sum_{n=1}^{n_{\max}}\sum_{m=-n}^{n}[\mathbf{a}_{1,nm}\mathbf{N}_{nm}(k\mathbf{r}) + \mathbf{a}_{2,nm}\mathbf{M}_{nm}(k\mathbf{r})] \tag{100}$$

Thus, the field coefficients $(a_{1,nm}, a_{2,nm})$ provide us with both the electric and magnetic field, from which we derive the local degree of optical chirality $\mathscr{C}$. The subroutine `calcLDOC` of the `multiscat` module calculates $\mathscr{C}$. The value of $\mathscr{C}$ is often normalised with respect to the degree of chirality of circularly-polarised plane waves $\mathscr{C} = \pm kE^2\varepsilon_0/2$ (+ for RCP and − for LCP, respectively), with incident electric field $E \equiv |\mathbf{E}_{\text{inc}}|$, defining

$$\overline{\mathscr{C}} = \frac{2}{k\varepsilon_0 E^2}\mathscr{C}. \tag{101}$$

3.7.3. Orientation-averaged optical chirality $\langle\overline{\mathscr{C}}\rangle$

For the calculation of orientation-averaged local degree of optical chirality $\langle\overline{\mathscr{C}}\rangle$, we combine the near-field averaging procedure of Sec. 3.7.1 with the treatment of optical activity in Sec. 3.6.3, expressing electric and magnetic fields as VSWFs in the helicity basis. We refer the reader to Ref. 22 for details of the derivation, and summarise the result:

$$\langle\overline{\mathscr{C}}\rangle = 2\pi k\varepsilon_0 E^2\,\Re\left(A_0 + B_0 + C_0 + D_0\right) \tag{102}$$

where, for right-handed circular polarisation

$$\begin{aligned}
A_0^{(\mathrm{R})} &= -1/4\pi \\
B_0^{(\mathrm{R})} &= \sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\widetilde{\mathbf{Z}}_R^\dagger(k\mathbf{r}_l)\left[-\mathbf{U}_{j,l}^{(R)} + \mathbf{V}_{j,l}^{(R)}\right]\right) \\
C_0^{(\mathrm{R})} &= \sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\left[-\mathbf{U}_{j,l}^{\dagger(R)} - \mathbf{V}_{j,l}^{\dagger(R)}\right]\widetilde{\mathbf{Z}}_R(k\mathbf{r}_l)\right) \\
D_0^{(\mathrm{R})} &= \sum_{j=1}^{N}\sum_{l=1}^{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\mathrm{Tr}\left(\mathbf{O}_{RR}^{(k,l)}\left[\mathbf{U}_{j,l}^{\dagger(R)} + \mathbf{V}_{j,l}^{\dagger(R)}\right]\right. \\
&\qquad\left.\left[-\mathbf{U}_{i,k}^{(R)} + \mathbf{V}_{i,k}^{(R)}\right]\right).
\end{aligned} \tag{103}$$

where $\widetilde{\mathbf{Z}}_R$ and $\widetilde{\mathbf{Z}}_L$ are the left and right regular VSWFs in helicity basis and $\mathbf{Z}_R$, $\mathbf{Z}_L$ the corresponding irregular VSWFs (cf Eqs. 69, 70). The terms $\mathbf{U}_{j,l}^{(R)}$ and $\mathbf{V}_{j,l}^{(R)}$ (and their Hermitian transpose) are introduced for conciseness and defined as,

$$\begin{aligned}
\mathbf{U}_{j,l}^{(R)} &:= \mathbf{Z}_R(k\mathbf{r}_j)\mathbf{T}_{RR}^{(j,l)}; & \mathbf{U}_{j,l}^{\dagger(R)} &= \mathbf{T}_{RR}^{\dagger(j,l)}\mathbf{Z}_R^\dagger(k\mathbf{r}_j) \\
\mathbf{V}_{j,l}^{(R)} &:= \mathbf{Z}_L(k\mathbf{r}_j)\mathbf{T}_{LR}^{(j,l)}; & \mathbf{V}_{j,l}^{\dagger(R)} &= \mathbf{T}_{LR}^{\dagger(j,l)}\mathbf{Z}_L^\dagger(k\mathbf{r}_j)
\end{aligned} \tag{104}$$

The corresponding formulas for left-handed circular polarisation read

$$\begin{aligned}
A_0^{(\mathrm{L})} &= +1/4\pi \\
B_0^{(\mathrm{L})} &= \sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\widetilde{\mathbf{Z}}_L^\dagger(k\mathbf{r}_l)\left[\mathbf{U}_{j,l}^{(L)} - \mathbf{V}_{j,l}^{(L)}\right]\right) \\
C_0^{(\mathrm{L})} &= \sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\left[\mathbf{U}_{j,l}^{\dagger(L)} + \mathbf{V}_{j,l}^{\dagger(L)}\right]\widetilde{\mathbf{Z}}_L(k\mathbf{r}_l)\right) \\
D_0^{(\mathrm{L})} &= \sum_{j=1}^{N}\sum_{l=1}^{N}\sum_{i=1}^{N}\sum_{k=1}^{N}\mathrm{Tr}\left(\mathbf{O}_{LL}^{(k,l)}\left[\mathbf{U}_{j,l}^{\dagger(L)} + \mathbf{V}_{j,l}^{\dagger(L)}\right]\right. \\
&\qquad\left.\left[\mathbf{U}_{i,k}^{(L)} - \mathbf{V}_{i,k}^{(L)}\right]\right).
\end{aligned} \tag{105}$$

with

$$\begin{aligned}
\mathbf{U}_{j,l}^{(L)} &:= \mathbf{Z}_L(k\mathbf{r}_j)\mathbf{T}_{LL}^{(j,l)}; & \mathbf{U}_{j,l}^{\dagger(L)} &= \mathbf{T}_{LL}^{\dagger(j,l)}\mathbf{Z}_L^\dagger(k\mathbf{r}_j) \\
\mathbf{V}_{j,l}^{(L)} &:= \mathbf{Z}_R(k\mathbf{r}_j)\mathbf{T}_{RL}^{(j,l)}; & \mathbf{V}_{j,l}^{\dagger(L)} &= \mathbf{T}_{RL}^{\dagger(j,l)}\mathbf{Z}_R^\dagger(k\mathbf{r}_j)
\end{aligned} \tag{106}$$

Note that the sum of B_0 and C_0 simplifies to,

$$\Re\left(B_0^{(\mathrm{R})}+C_0^{(\mathrm{R})}\right)=-2\Re\left(\sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\widetilde{\mathbf{Z}}_{R}^{\dagger}(k\mathbf{r}_l)\mathbf{U}_{j,l}^{(R)}\right)\right) \tag{107}$$

and

$$\Re\left(B_0^{(\mathrm{L})}+C_0^{(\mathrm{L})}\right)=2\Re\left(\sum_{j=1}^{N}\sum_{l=1}^{N}\mathrm{Tr}\left(\widetilde{\mathbf{Z}}_{L}^{\dagger}(k\mathbf{r}_l)\mathbf{U}_{j,l}^{(L)}\right)\right). \tag{108}$$

These formulas (102–108) are implemented in the subroutine `calcOaLDOC` of the `multiscat` module.

3.8. Solution schemes

TERMS offers several options, selected by `Scheme`, to solve the multiple scattering problem described in Section 3.4. It generally requires determining the particle-centred coefficients $\mathbf{c}_j^{(j)}$ for given individual scatterer properties (described by $\mathbf{T}_1^{(j)}$) and an excitation field (described by $\widetilde{\mathbf{a}}^{(j)}$) impinging from a particular direction. This can be achieved by solving the linear system of equations in (36) for $\mathbf{c}_j^{(j)}$ *without* performing matrix inversion, thus producing a complete description of the scattered field for the given excitation. The linear system can be solved with multiple right-hand sides, representing different excitations, with standard linear algebra routines. Performing matrix inversion to determine the collective T-matrix becomes worthwhile only when many impinging directions are to be considered, or when orientation-averaged quantities are of primary interest.

A brute force approach to solving the multiple scattering problem would be to construct the $Nl_{\max}\times Nl_{\max}$ matrix in equation (36) and then invert it to obtain the pairwise scatterer-centred T-matrices $\mathbf{T}^{(i,j)}$. However, this approach is computationally demanding. To help alleviate the cost of one large matrix inversion, Stout *et al.*[14,15] proposed a recursive scheme where a smaller ($l_{\max}\times l_{\max}$) matrix is inverted $N-1$ times.

3.8.1. Recursive scheme with matrix balancing

In the recursive algorithm described by Stout *et al.*[14], the elements of $\mathbf{T}_N^{(j,i)}$ are accumulated recursively from auxiliary subsystems, incrementally built up from one to N particles. The recursive system is prescribed by the following four equations:

$$\begin{aligned}\mathbf{T}_N^{(N,N)}=&\left[\left[\mathbf{T}_1^{(N)}\right]^{-1}\right. &(109)\\ &\left.-\sum_{j=1}^{N-1}\mathbf{O}^{(N,j)}\left(\underline{\sum_{i=1}^{N-1}\mathbf{T}_{N-1}^{(j,i)}\mathbf{O}^{(i,N)}}\right)\right]^{-1}\\ =:&\,\mathbf{S}^{-1}, &(110)\end{aligned}$$

$$\mathbf{T}_N^{(N,i)}=\mathbf{T}_N^{(N,N)}\left(\sum_{j=1}^{N-1}\mathbf{O}^{(N,j)}\mathbf{T}_{N-1}^{(j,i)}\right),\quad i\neq N, \tag{111}$$

$$\mathbf{T}_N^{(j,N)}=\left(\underline{\sum_{i=1}^{N-1}\mathbf{T}_{N-1}^{(j,i)}\mathbf{O}^{(i,N)}}\right)\mathbf{T}_N^{(N,N)},\quad j\neq N, \tag{112}$$

$$\begin{aligned}\mathbf{T}_N^{(j,i)}=&\,\mathbf{T}_{N-1}^{(j,i)} &(113)\\ &+\left(\underline{\sum_{l=1}^{N-1}\mathbf{T}_{N-1}^{(j,l)}\mathbf{O}^{(l,N)}}\right)\mathbf{T}_N^{(N,i)},\quad j,i\neq N, &(114)\end{aligned}$$

where a common matrix sum has been underlined. Note that only one $l_{\max}\times l_{\max}$ matrix is inverted on each of the $N-1$ iterations. The inverted matrix becomes ill-conditioned for large $n_{\max}$, but the associated problems can be (at least partly) circumvented by applying the recursive scheme to appropriately "balanced" matrices and coefficients:[15]

$$\begin{aligned}\left[\widehat{\mathbf{T}}_N^{(j,i)}\right]_{sp,s'p'}:=&\left[\mathbf{D}^{(j)}\mathbf{T}_N^{(j,i)}[\widetilde{\mathbf{D}}^{(i)}]^{-1}\right]_{sp,s'p'} &(115)\\ =&\frac{\xi_{n(p)}(k_{\mathrm{M}}R_j)}{\psi_{n'(p')}(k_{\mathrm{M}}R_i)}\left[\mathbf{T}_N^{(j,i)}\right]_{sp,s'p'}\\ \left[\widehat{\mathbf{T}}_1^{(j)}\right]_{sp,s'p'}:=&\left[\mathbf{D}^{(j)}\mathbf{T}_1^{(j)}[\widetilde{\mathbf{D}}^{(j)}]^{-1}\right]_{sp,s'p'} &(116)\\ =&\frac{\xi_{n(p)}(k_{\mathrm{M}}R_j)}{\psi_{n'(p')}(k_{\mathrm{M}}R_j)}\left[\mathbf{T}_1^{(j)}\right]_{sp,s'p'}\\ \left[\widehat{\mathbf{O}}^{(j,i)}\right]_{sp,s'p'}:=&\left[\widetilde{\mathbf{D}}^{(j)}\mathbf{O}^{(j,i)}[\mathbf{D}^{(i)}]^{-1}\right]_{sp,s'p'} &(117)\\ =&\frac{\psi_{n(p)}(k_{\mathrm{M}}R_j)}{\xi_{n'(p')}(k_{\mathrm{M}}R_i)}\left[\mathbf{O}^{(j,i)}\right]_{sp,s'p'},\end{aligned}$$

where $\widetilde{\mathbf{D}}^{(j)}$ and $\mathbf{D}^{(j)}$ are regular and irregular diagonal matrices with the Riccati-Bessel functions

$\psi_n(x) = xj_n(x)$ or $\xi_n(x) = xh_n(x)$ on the diagonal. (Here, $j_n(x)$ and $h_n(x)$ are, respectively, the spherical Bessel and Hankel functions of the first kind)[19]. Note that Stout *et al.*[15]'s "matrix balancing" may also be regarded as "left and right preconditioning", as the matrix to be inverted is essentially left- and right-multiplied by two different matrices to improve its condition number, which aims to make numerical inversion more robust and accurate. Instead of balancing throughout, as Stout *et al.*[15] propose to do, we prefer to localise the balancing act just at the inversion step in (110), i.e.

$$\begin{aligned}\mathbf{T}_N^{(N,N)} =& \mathbf{S}^{-1}\\ =& \left[\mathbf{D}^{(N)}\right]^{-1}\mathbf{D}^{(N)}\mathbf{S}^{-1}\left[\widetilde{\mathbf{D}}^{(N)}\right]^{-1}\widetilde{\mathbf{D}}^{(N)} \qquad (118)\\ =& \left[\mathbf{D}^{(N)}\right]^{-1}\left[\widetilde{\mathbf{D}}^{(N)}\mathbf{S}\left[\mathbf{D}^{(N)}\right]^{-1}\right]^{-1}\widetilde{\mathbf{D}}^{(N)}\\ =:& \left[\mathbf{D}^{(N)}\right]^{-1}\widehat{\mathbf{S}}^{-1}\widetilde{\mathbf{D}}^{(N)} \qquad (119)\end{aligned}$$

where $\widehat{\mathbf{S}}$ is obtained by balancing $\mathbf{S}$ analogously to $\widehat{\mathbf{T}}_N^{(j,i)}$. Note that equation (110) can be factored in two ways:

$$\mathbf{T}_N^{(N,N)} = \mathbf{T}_1^{(N)}\mathbf{S}_R^{-1} = \mathbf{S}_L^{-1}\mathbf{T}_1^{(N)} \qquad (120)$$

where

$$\mathbf{S}_R = \left[\mathbf{I} - \sum_{j=1}^{N-1}\mathbf{O}^{(N,j)}\underline{\left(\sum_{i=1}^{N-1}\mathbf{T}_{N-1}^{(j,i)}\mathbf{O}^{(i,N)}\right)}\mathbf{T}_1^{(N)}\right] \qquad (121)$$

$$\mathbf{S}_L = \left[\mathbf{I} - \mathbf{T}_1^{(N)}\sum_{j=1}^{N-1}\mathbf{O}^{(N,j)}\underline{\left(\sum_{i=1}^{N-1}\mathbf{T}_{N-1}^{(j,i)}\mathbf{O}^{(i,N)}\right)}\right] \qquad (122)$$

either of which may be preferred if $\mathbf{T}_1^{(N)}$ is difficult to invert. To facilitate the inversion of $\mathbf{S}_L$ and $\mathbf{S}_R$, slightly different balancing (and subsequent unbalancing) should be used:

$$\mathbf{S}_L^{-1} = \left[\mathbf{D}_N^{(N)}\right]^{-1}\left[\mathbf{D}_N^{(N)}\mathbf{S}_L\left[\mathbf{D}_N^{(N)}\right]^{-1}\right]^{-1}\mathbf{D}_N^{(N)} \qquad (123)$$

$$=: \left[\mathbf{D}_N^{(N)}\right]^{-1}\widehat{\mathbf{S}}_L^{-1}\mathbf{D}_N^{(N)}, \qquad (124)$$

$$\mathbf{S}_R^{-1} = \widetilde{\mathbf{D}}_N^{(N)}\left[\left[\widetilde{\mathbf{D}}_N^{(N)}\right]^{-1}\mathbf{S}_R\widetilde{\mathbf{D}}_N^{(N)}\right]^{-1}\left[\widetilde{\mathbf{D}}_N^{(N)}\right]^{-1} \qquad (125)$$

$$=: \widetilde{\mathbf{D}}_N^{(N)}\widehat{\mathbf{S}}_R^{-1}\left[\widetilde{\mathbf{D}}_N^{(N)}\right]^{-1}. \qquad (126)$$

Here $\mathbf{S}_L$ is balanced using only irregular weights, while $\mathbf{S}_R$ is balanced like a T-matrix using only regular weights. In our experience, $\widehat{\mathbf{S}}_R$ is much better conditioned for inversion than $\widehat{\mathbf{S}}$.

In TERMS program the balancing formulas are implemented in the subroutines `balanceVecJ` and `balanceMatJI`.

3.8.2. Mackowski & Mishchenko's scheme

From equations (43) and (45) it follows that

$$\begin{bmatrix} \mathbf{I} & \cdots & -\mathbf{T}_1^{(1)}\mathbf{O}^{(1,N)} \\ \vdots & \ddots & \vdots \\ -\mathbf{T}_1^{(N)}\mathbf{O}^{(N,1)} & \cdots & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{T}_N^{(1,1)} & \cdots & \mathbf{T}_N^{(1,N)} \\ \vdots & \ddots & \vdots \\ \mathbf{T}_N^{(N,1)} & \cdots & \mathbf{T}_N^{(N,N)} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_1^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{T}_1^{(N)} \end{bmatrix}, \tag{127}$$

or, equivalently,

$$\mathbf{T}_N^{(i,i)} - \sum_{j\neq i} \mathbf{T}_1^{(i)}\mathbf{O}^{(i,j)}\mathbf{T}_N^{(j,i)} = \mathbf{T}_1^{(i)}, \tag{128}$$

which is a linear system of the general form $\mathbf{AX} = \mathbf{B}$, where we want to find matrix $\mathbf{X}$ for a given $\mathbf{A}$ and $\mathbf{B}$, which contains multiple right hand sides. That is, each column of $\mathbf{X}$ and $\mathbf{B}$ can be treated independently, so we have to solve many linear systems of the form $\mathbf{Ax}_\nu = \mathbf{b}_\nu$, where $\mathbf{x}_\nu$ and $\mathbf{b}_\nu$ are the ν'th column of $\mathbf{X}$ and $\mathbf{B}$, respectively.

Mackowski & Mishchenko (M&M)[17,18] "contract" the second particle index of $\mathbf{T}_N^{(j,i)}$, using $\mathbf{T}_N^{(j)} = \sum_i \mathbf{T}_N^{(j,i)}\widetilde{\mathbf{O}}^{(i,0)}$, to rewrite the linear system in terms of *individual* (as opposed to pairwise) scatterer-centred T-matrices $\mathbf{T}_N^{(j)}$, i.e.

$$\begin{bmatrix} \mathbf{I} & \cdots & -\mathbf{T}_1^{(1)}\mathbf{O}^{(1,N)} \\ \vdots & \ddots & \vdots \\ -\mathbf{T}_1^{(N)}\mathbf{O}^{(N,1)} & \cdots & \mathbf{I} \end{bmatrix} \begin{pmatrix} \mathbf{T}_N^{(1)} \\ \vdots \\ \mathbf{T}_N^{(N)} \end{pmatrix} = \begin{pmatrix} \mathbf{T}_1^{(1)}\widetilde{\mathbf{O}}^{(1,0)} \\ \vdots \\ \mathbf{T}_1^{(N)}\widetilde{\mathbf{O}}^{(N,0)} \end{pmatrix}, \tag{129}$$

or, equivalently,

$$\mathbf{T}_N^{(i)} - \sum_{j\neq i} \mathbf{T}_1^{(i)}\mathbf{O}^{(i,j)}\mathbf{T}_N^{(j)} = \mathbf{T}_1^{(i)}\widetilde{\mathbf{O}}^{(i,0)}, \tag{130}$$

where the number of independent linear systems of the form $\mathbf{Ax}_\nu = \mathbf{b}_\nu$ is now reduced. M&M use the biconjugate gradient method to solve $\mathbf{Ax}_\nu = \mathbf{b}_\nu$ for each ν, where the row order (i.e. length of each column vector) is predetermined from Mie theory for each (spherical) scatterer in isolation, and the truncation limit for the column order (i.e. the maximum value of ν) is determined on-the-fly from the convergence of each scatterer's contribution to the collective rotationally-averaged extinction cross-section (see Eq. 66 in Ref. 17).

Mackowski & Mishchenko's scheme is implemented as `Scheme = 3` in TERMS, with the addition of balancing discussed above, though we use a direct linear solver instead of an iterative one.

4. Conclusion and outlook

We have introduced TERMS, an open-source Fortran program to simulate light scattering by rigid clusters of nanoparticles, in fixed or random orientation with respect to incident light. TERMS implements several superposition T-matrix algorithms and recently-derived formulas for analytical orientation-averaging of far-field and near-field optical properties. This manuscript provides a brief summary of the method and references the key formulas implemented in the program. A companion website[13] includes a comprehensive suite of self-contained examples illustrating the program's capabilities in realistic calculations. We hope this program will be useful to the light scattering community of researchers, and we welcome contributions to extend the program's use cases. As noted in the introduction, the superposition T-matrix method has been implemented in several other publicly-available programs, each with its own set of features, and we welcome collaboration to combine these efforts. We conclude below with an outlook of the possible extensions we are considering for the future development of TERMS.

Code improvements

- Optional use of an iterative solver to solve large linear systems
- Import/export of T-matrices in HDF5 format
- Import of T-matrices for more general particle shapes, from Scuff-EM[44]
- Built-in calculation of spheroid T-matrices (port of SMARTIES[8])
- Improved methods for the calculation of TACs[58]
- Additional built-in material dielectric functions
- Optimisation of time-consuming calculations

New features

- Calculation of internal fields for non-spherical particles obtained via EBCM[8] (exporting matrix $\mathbf{R} = \mathbf{Q}^{-1}$)
- Orientation-averaged internal fields for coated spheres and nonspherical particles, adapting Ref. 14
- Orientation-averaged partial absorptions for layered particles
- Orientation-averaged absorption and scattering circular dichroism in Stout's scatterer-centred formalism (not from the collective T-matrix)
- Chiral media, following Ref. 59
- T-matrix for anisotropic core-shell spheres, following Ref. 60
- T-matrix for coated spheroids, following Refs. 61,62
- Dipolar incident field
- T-matrix of model molecules, following Ref. 63
- Conversion from T-matrix to "Higher-Order Polarizability Tensors", following Ref. 41
- Extension to infinite periodic arrays, following Ref. 31
- Integral representation of near-fields in the Rayleigh region, following Ref. 43
- Plane-wave expansion for particles with intersecting circumscribed spheres, following Ref. 42
- Geometry optimisation (via external libraries)

We also consider porting the codebase to the Julia language[64], to benefit from an interactive environment to develop and test new features, without sacrificing run-time performance.

Acknowledgments

The authors acknowledge the support of the Royal Society Te Apārangi for support through a Rutherford Discovery Fellowship (B.A., grant number RDF-VUW1603), and the MacDiarmid Institute. We dedicate this work to the memory of Michael I. Mishchenko, who had a tremendous influence on the electromagnetic and light scattering community, and left us too soon.

References

[1] P. Waterman, Matrix formulation of electromagnetic scattering, Proceedings of the IEEE 53 (8) (1965) 805–812. `doi:10.1109/PROC.1965.4058`.

[2] P. Waterman, New formulation of acoustic scattering, The journal of the acoustical society of America 45 (6) (1969) 1417–1429. `doi:10.1121/1.1911619`.

[3] P. C. Waterman, Symmetry, unitarity, and geometry in electromagnetic scattering, Physical review D 3 (4) (1971) 825–839. `doi:10.1103/PhysRevD.3.825`.

[4] M. I. Mishchenko, L. D. Travis, A. A. Lacis, Scattering, absorption, and emission of light by small particles, Cambridge University Press, 2002.

[5] B. Peterson, S. Ström, T-matrix for electromagnetic scattering from an arbitrary number of scatterers and representations of E(3), Phys. Rev. D 8 (1973) 3661–3678. `doi:10.1103/PhysRevD.8.3661`.

[6] W. Chew, Waves and Fields in Inhomogeneous Media, IEEE Press series on electromagnetic waves, Van Nostrand Reinhold, 1990.

[7] D. W. Mackowski, Analysis of radiative scattering for multiple sphere configurations, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 433 (1889) (1991) 599–614. `doi:10.1098/rspa.1991.0066`.

[8] W. Somerville, B. Auguié, E. C. Le Ru, Smarties: User-friendly codes for fast and accurate calculations of light scattering by spheroids, J. Quant. Spectrosc. Ra. 174 (2016) 39–55. `doi:10.1016/j.jqsrt.2016.01.005`.

[9] D. Mackowski, The Extension of Mie Theory to Multiple Spheres, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 223–256. `doi:10.1007/978-3-642-28738-1_8`.

[10] M. A. Yurkin, Performance of iterative solvers in the discrete dipole approximation, in: 2016 URSI International Symposium on Electromagnetic Theory (EMTS), 2016, pp. 488–491. doi:10.1109/URSI-EMTS.2016.7571433.

[11] J. Markkanen, A. J. Yuffa, Fast superposition T-matrix solution for clusters with arbitrarily-shaped constituent particles, Journal of Quantitative Spectroscopy and Radiative Transfer 189 (2017) 181–188. doi:10.1016/j.jqsrt.2016.11.004.

[12] D. M. Solís, J. Taboada, F. Obelleiro, L. M. Liz-Marzán, F. J. García de Abajo, Toward ultimate nanoplasmonics modeling, ACS Nano 8 (8) (2014) 7559–7570. doi:10.1021/nn5037703.

[13] D. Schebarchov, A. Fazel-Najafabadi, E. C. Le Ru, B. Auguié, TERMS website, 2021 [cited 2021]. doi:10.5281/zenodo.5703291.
URL http://nano-optics.ac.nz/terms

[14] B. Stout, J.-C. Auger, J. Lafait, A transfer matrix approach to local field calculations in multiple-scattering problems, Journal of Modern Optics 49 (13) (2002) 2129–2152. doi:10.1080/09500340210124450.

[15] B. Stout, J. C. Auger, A. Devilez, Recursive T matrix algorithm for resonant multiple scattering: applications to localized plasmon excitations, J. Opt. Soc. Am. A 25 (10) (2008) 2549–2557. doi:10.1364/JOSAA.25.002549.

[16] D. W. Mackowski, Calculation of total cross sections of multiple-sphere clusters, J. Opt. Soc. Am. A 11 (11) (1994) 2851–2861. doi:10.1364/JOSAA.11.002851.

[17] D. W. Mackowski, M. I. Mishchenko, Calculation of the T matrix and the scattering matrix for ensembles of spheres, J. Opt. Soc. Am. A 13 (11) (1996) 2266–2278. doi:10.1364/JOSAA.13.002266.

[18] D. Mackowski, M. Mishchenko, A multiple sphere T-matrix fortran code for use on parallel computer clusters, Journal of Quantitative Spectroscopy and Radiative Transfer 112 (13) (2011) 2182 – 2192. doi:10.1016/j.jqsrt.2011.02.019.

[19] E. C. Le Ru, P. Etchegoin, Principles of Surface-Enhanced Raman Spectroscopy: and related plasmonic effects, Elsevier, 2009.

[20] D. W. Mackowski, R. A. Altenkirch, M. P. Menguc, Internal absorption cross sections in a stratified sphere, Appl. Opt. 29 (10) (1990) 1551–1559. doi:10.1364/AO.29.001551.

[21] R. N. S. Suryadharma, C. Rockstuhl, Predicting observable quantities of self-assembled metamaterials from the T-matrix of its constituting meta-atom, Materials 11 (2) (2018). doi:10.3390/ma11020213.

[22] A. Fazel-Najafabadi, S. Schuster, B. Auguié, Orientation averaging of optical chirality near nanoparticles and aggregates, Physical Review B 103 (11) (2021) 115405. doi:10.1103/PhysRevB.103.115405.

[23] D. Schebarchov, E. C. Le Ru, J. Grand, B. Auguié, Mind the gap: testing the Rayleigh hypothesis in T-matrix calculations with adjacent spheroids, Optics express 27 (24) (2019) 35750–35760. doi:10.1364/OE.27.035750.

[24] The HDF Group, Hierarchical data format version 5 (2000-2010).
URL http://www.hdfgroup.org/HDF5

[25] F. J. García de Abajo, Multiple scattering of radiation in clusters of dielectrics, Phys. Rev. B 60 (1999) 6086–6102. doi:10.1103/PhysRevB.60.6086.

[26] W. Vargas, L. Cruz, L. F. Fonseca, M. Gómez, T-matrix approach for calculating local fields around clusters of rotated spheroids, Appl. Opt. 32 (12) (1993) 2164–2170. doi:10.1364/AO.32.002164.

[27] Y.-l. Xu, Electromagnetic scattering by an aggregate of spheres, Applied optics 34 (21) (1995) 4573–4588. doi:10.1364/AO.34.004573.

[28] B. Stout, J.-C. Auger, J. Lafait, Individual and aggregate scattering matrices and cross-sections: Conservation laws and reciprocity, Journal of Modern Optics 48 (14) (2001) 2105–2128. doi:10.1080/09500340108235502.

[29] M. Fruhnert, I. Fernandez-Corbaton, V. Yannopapas, C. Rockstuhl, Computing the T-matrix of a scattering object with multiple plane wave illuminations, Beilstein Journal of Nanotechnology 8 (2017) 614–626. doi:10.3762/bjnano.8.66.

[30] M. I. Mishchenko, Comprehensive thematic T-matrix reference database: a 2017–2019 update, Journal of Quantitative Spectroscopy and Radiative Transfer 242 (2020) 106692. doi:10.1016/j.jqsrt.2019.106692.

[31] M. N. . P. Törmä, Multiple-scattering T-matrix simulations for nanophotonics: Symmetries and periodic lattices, Communications in Computational Physics 30 (2) (2021) 357–395. doi:10.4208/cicp.oa-2020-0136.
URL http://dx.doi.org/10.4208/cicp.OA-2020-0136

[32] F. Kahnert, Numerical methods in electromagnetic scattering theory, Journal of Quantitative Spectroscopy and Radiative Transfer 79-80 (2003) 775–824. doi:10.1016/S0022-4073(02)00321-7.

[33] J. Jin, The Finite Element Method in Electromagnetics, 3rd Edition, Wiley-IEEE Press, 2014.

[34] M. A. Yurkin, A. G. Hoekstra, The discrete dipole approximation: An overview and recent developments, Journal of Quantitative Spectroscopy and Radiative Transfer 106 (1) (2007) 558–589. doi:10.1016/j.jqsrt.2007.01.034.

[35] B. Archambeault, O. M. Ramahi, C. Brench, O. M. Ramahi, The Finite-Difference Time-Domain Method, Springer US, Boston, MA, 1998, Ch. 3, pp. 35–67. doi:10.1007/978-1-4757-5124-6{_}3.

[36] M. Mishchenko, Interstellar light absorption by randomly oriented nonspherical dust grains, Soviet Astronomy Letters 15 (1989) 299.

[37] M. Mishchenko, Extinction of light by randomly-oriented non-spherical grains, Astrophysics and space science 164 (1) (1990) 1–13.

[38] N. G. Khlebtsov, Orientational averaging of light-scattering observables in the T-matrix approach, Applied Optics 31 (25) (1992) 5359–5365. doi:10.1364/AO.31.005359.

[39] F. Borghese, P. Denti, R. Saija, Scattering from model nonspherical particles: theory and applications to environmental physics, Springer, 2007.

[40] A. Fazel-Najafabadi, B. Auguié, Orientation dependence of optical activity in light scattering by nanoparticle clusters, (submitted) NA (NA) (2021) NA. arXiv:2111.00997.

[41] J. Mun, S. So, J. Jang, J. Rho, Describing meta-atoms using the exact higher-order polarizability tensors, ACS Photonics 7 (5) (2020) 1153–1162. doi:10.1021/acsphotonics.9b01776.

[42] D. Theobald, A. Egel, G. Gomard, U. Lemmer, Plane-wave coupling formalism for T-matrix simulations of light scattering by nonspherical particles, Phys. Rev. A

96 (2017) 033822. doi:10.1103/PhysRevA.96.033822.

[43] B. Auguié, W. R. C. Somerville, S. Roache, E. C. Le Ru, Numerical investigation of the Rayleigh hypothesis for electromagnetic scattering by a particle, Journal of Optics 18 (7) (2016) 075007. doi:10.1088/2040-8978/18/7/075007.

[44] M. T. Homer Reid, S. G. Johnson, Efficient Computation of Power, Force, and Torque in BEM Scattering Calculations, preprint (Jul. 2013). arXiv:1307.2966.

[45] V. L. Y. Loke, T. A. Nieminen, N. R. Heckenberg, H. Rubinsztein-Dunlop, *T*-matrix calculation via discrete dipole approximation, point matching and exploiting symmetry, Journal of Quantitative Spectroscopy and Radiative Transfer 110 (14) (2009) 1460–1471. doi:10.1016/j.jqsrt.2009.01.013.

[46] B. R. Johnson, Invariant imbedding T matrix approach to electromagnetic scattering, Appl. Opt. 27 (23) (1988) 4861–4873. doi:10.1364/AO.27.004861.

[47] A. M. Kern, O. J. F. Martin, Surface integral formulation for 3d simulations of plasmonic and high permittivity nanostructures, J. Opt. Soc. Am. A 26 (4) (2009) 732–740. doi:10.1364/JOSAA.26.000732.

[48] N. Stefanou, V. Karathanos, A. Modinos, Scattering of electromagnetic waves by periodic structures, Journal of Physics: Condensed Matter 4 (36) (1992) 7389–7400. doi:10.1088/0953-8984/4/36/013.

[49] A. Doicu, T. Wriedt, Y. Eremin, Light Scattering by Systems of Particles-Null-Field Method with Discrete Sources-Theory and Programs, Vol. 124, Springer, 2006. doi:10.1007/978-3-540-33697-6.

[50] F. J. García de Abajo, Colloquium: Light scattering by particle and hole arrays, Rev. Mod. Phys. 79 (2007) 1267–1290. doi:10.1103/RevModPhys.79.1267.

[51] S. Lee, H. Hwang, W. Lee, D. Schebarchov, Y. Wy, J. Grand, B. Auguié, D. H. Wi, E. Cortés, S. W. Han, Core–shell bimetallic nanoparticle trimers for efficient light-to-chemical energy conversion, ACS Energy Letters 5 (12) (2020) 3881–3890. doi:10.1021/acsenergylett.0c02110.

[52] M. Herran, A. Sousa-Castillo, C. Fan, S. Lee, W. Xie, M. Doblinger, B. Auguié, E. Cortés, Tailoring plasmonic bimetallic nanocatalysts towards sunlight-driven H2 production, (submitted) (2021).

[53] J.-C. Auger, B. Stout, Local field intensity in aggregates illuminated by diffuse light: T matrix approach, Applied optics 47 (16) (2008) 2897–2905. doi:10.1364/AO.47.002897.

[54] C. Multiphysics, Introduction to comsol multiphysics, COMSOL Multiphysics, Burlington, MA 9 (1998) 2018.

[55] W. Chew, Recurrence relations for three-dimensional scalar addition theorem, Journal of Electromagnetic Waves and Applications 6 (1-4) (1992) 133–142. doi:10.1163/156939392X01075.

[56] M. I. Mishchenko, M. A. Yurkin, On the concept of random orientation in far-field electromagnetic scattering by nonspherical particles, Opt. Lett. 42 (3) (2017) 494–497. doi:10.1364/OL.42.000494.

[57] M. Hentschel, Complex 2D & 3D plasmonic nanostructures : Fano resonances, chirality, and nonlinearities, Ph.D. thesis, University of Stuttgart (2013).

[58] W. C. Chew, Vector addition theorem and its diagonalization, Communications in Computational Physics 3 (2) (2008) 330–341. doi:https://doi.org/.

[59] D. W. Mackowski, A multiple sphere T -matrix FORTRAN code for use on parallel computer clusters Version 3.0 36.

[60] C. Tang, B. Auguié, E. C. Le Ru, Refined effective-medium model for the optical properties of nanoparticles coated with anisotropic molecules, Phys. Rev. B 103 (2021) 085436. doi:10.1103/PhysRevB.103.085436.

[61] B. Peterson, S. Ström, T-matrix formulation of electromagnetic scattering from multilayered scatterers, Phys. Rev. D 10 (1974) 2670–2684. doi:10.1103/PhysRevD.10.2670.

[62] A. Quirantes, A T-matrix method and computer code for randomly oriented, axially symmetric coated scatterers, Journal of Quantitative Spectroscopy and Radiative Transfer 92 (3) (2005) 373–381. doi:10.1016/j.jqsrt.2004.08.004.

[63] I. Fernandez-Corbaton, D. Beutel, C. Rockstuhl, A. Pausch, W. Klopper, Computation of electromagnetic properties of molecular ensembles, ChemPhysChem 21 (9) (2020) 878–887. doi:10.1002/cphc.202000072.

[64] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, Julia: A fresh approach to numerical computing, SIAM Review 59 (1) (2017) 65–98. doi:10.1137/141000671. URL https://epubs.siam.org/doi/10.1137/141000671

[65] A. D. Rakić, A. B. Djurišić, J. M. Elazar, M. L. Majewski, Optical properties of metallic films for vertical-cavity optoelectronic devices, Applied optics 37 (22) (1998) 5271–5283.

[66] D. E. Aspnes, A. Studna, Dielectric functions and optical parameters of Si, Ge, GaP, GaAs, GaSb, InP, InAs, and InSb from 1.5 to 6.0 eV, Physical review B 27 (2) (1983) 985.

[67] A. D. Rakić, Algorithm for the determination of intrinsic optical constants of metal films: application to aluminum, Applied optics 34 (22) (1995) 4755–4767.

[68] E. D. Palik, Handbook of optical constants of solids, Vol. 3, Academic press, 1998.

[69] M. Daimon, A. Masumura, Measurement of the refractive index of distilled water from the near-infrared region to the ultraviolet region, Applied optics 46 (18) (2007) 3811–3820.

A. Appendix

A.1. Keywords

List of case-sensitive keywords and corresponding arguments supported by TERMS. Optional arguments are enclosed in square brackets (nested in some cases).

Main input parameters

`ModeAndScheme M S`

If present, this keyword must appear first in the input file. It takes two arguments: positive integer M specifying the desired calculation mode; and non-negative integer S specifying the solution scheme to be used. The default values are $M = 2$ and $S = 3$.

Mode of calculation

- $M = 1$ triggers a single- or multi-wavelength calculation of near fields $\mathbf{E}$, $\mathbf{B}$ and optical chirality $\mathscr{C}$, at fixed incidence directions and/or orientation-averaged
- $M = 2$ triggers a single- or multi-wavelength calculation of far-field properties (e.g. spectra of optical cross-sections), at fixed incidence directions and/or orientation-averaged
- $M = 3$ triggers a single- or multi-wavelength calculation of polarimetric properties, such as Stokes scattering vectors, phase matrix, and differential scattering cross sections for multiple incidence and/or scattering angles

Scheme of solution

- $S = 0$ will seek solutions for the given angles of incidence, without seeking the collective T-matrix
- $S > 0$ will calculate the collective T-matrix either ($S = 1$) by direct inversion of the complete linear system to obtain $T^{(i,j)}$, or ($S = 2$) by using Stout's iterative scheme for $T^{(i,j)}$, or ($S = 3$) by using Mackowski & Mishchenko's scheme for $T^{(i)}$. Note that fixed-orientation cross-sections are also calculated when $S > 0$.

`Scatterers N`

This keyword must appear last in the `inputfile`, with a single positive integer argument N specifying the number of scatterers. The following N lines specify all the required parameters per scatterer, and each line must contain five or more space-separated fields, i.e.

```
Tag x y z R [ a b c [ d ] ]      ( if Tag(1:2)  = "TF" )
            [ a [ b [ c ] ] ]    ( if Tag(1:2) != "TF" )
```

where *Tag* is a contiguous string, which may contain one underscore to separate the root from the suffix; x, y, z are the cartesian coordinates (in the lab frame) for the scatterer, whose smallest circumscribing sphere has radius R. All other subsequent parameters (inside square brackets) depend on the root of *Tag*.

Before the root of *Tag* is parsed, the code first looks for a suffix of the form *_S?* and associates it with a multipole selection predefined using the `MultipoleSelections` keyword.

If the root of *Tag* is either "TF1", "TF2", ... , or "TF9", which correspond to a 1-body T-matrix file listed under the `TmatrixFiles` keyword, floats a, b, and c can be supplied to specify the Euler angles describing the scatterer orientation (default angle values are all zero). Another float d may be included to specify the aspect ratio for spheroids, which is currently only used for mapping local field intensity and visualising the geometry. Note that d is interpreted as the ratio of polar (along z) to equatorial (along x or y) length, so that $d > 1$ for prolate spheroids, $d < 1$ for oblate spheroids, and $d = 1$ for spheres (default). Note that for nonspherical particles the circumscribing sphere radius R is used to check for potential geometrical overlap between particles, but also in the balancing scheme.

If the root of *Tag* is not "TF?", the 1-body T-matrix is computed using Mie theory, which is applicable to coated spheres. The expected *Tag* format is `L0@L1@L2@L3`, with the character "@" delimiting substrings that specify the material dielectric function of each concentric region inside the scatterer, starting from the core (*L0*) and going *outward*. The number of coats is inferred from the number of instances of "@" and is currently capped at 3. *Tag* of a homogeneous sphere (without layers) should not contain any "@", i.e. *Tag* = *L0*. Currently accepted values for *L?* are: "Au", "Ag", "Al", "Cr", "Pt", "Pd", "Si", and "Water" which trigger internal calculation of the wavelength-dependent dielectric functions for the required material, or "DF1", "DF2", . . . , "DF9" to impose a custom dielectric function listed under the `DielectricFunctions` keyword. For coated spheres, the outer radius of each region must be specified by floats R, a, b, c in the order of decreasing size (i.e. going radially *inward*).

`TmatrixFiles Nfiles`
Specifies the number of external T-matrix files (default: *Nfiles = 0*). The subsequent *Nfiles* lines are each read as a string and then interpreted as a filename. Wrap the string in quotation marks if it contains the relative path or special characters, e.g. `"../../tmatrix_Au_spheroid_50x20_water.tmat"`. Note that the wavelengths in each file must *exactly* correspond to the values specified by the `Wavelength` keyword.

The T-matrix file format is as follows:

- First line is a comment (starts with a `#`) describing the format `# s sp n np m mp Tr Ti`
- Second line is also a comment and starts with `# lambda= N1 nelements= N2` where N1 is the wavelength in nanometres, and N2 is the number of T-matrix elements to be read below
- Subsequent lines contain the indices and T-matrix values for this particular wavelength,

1. `s`, `sp` are the row (resp. column) index of the multipole mode (1: magnetic, or 2: electric)
2. `n`, `np` index the multipole degree
3. `m`, `mp` index the multipole order
4. `Tr`, `Ti` give the real and imaginary part of the T-matrix element

- If the file contains multiple wavelengths each wavelength-block is appended below the others, starting with a line `# lambda= N1 nelements= N2`

An example is show below,

```
# s sp n np m mp Tr Ti | prolate Au spheroid in water, a = 10 c = 20
# lambda= 400 nelements= 136 epsIn= -1.649657+5.771763j
  1   1   1   1  -1  -1 -1.189109253832815e-04 -2.161746691903687e-05
  1   1   1   1   0   0 -5.597968829951113e-05 -3.444956295771378e-05
... [truncated]
  2   2   4   4   3   3 -3.794740062663782e-11 5.636725538124517e-11
  2   2   4   4   4   4 -1.113090425618089e-11 1.707927691863483e-11
# lambda= 402 nelements= 136 epsIn= -1.661947+5.778032j
  1   1   1   1  -1  -1 -1.160926707256971e-04 -2.119092055798298e-05
  1   1   1   1   0   0 -5.467319805259745e-05 -3.371696756234449e-05
... [truncated]
  2   2   4   4   3   3 -1.279170882307354e-15 1.378894188143029e-13
  2   2   4   4   4   4 -3.752182192799965e-16 4.101975575297762e-14
... [truncated]
# lambda= 800 nelements= 136 epsIn= -24.236565+1.458652j
```

```
  1   1   1   1  -1  -1 -7.146139984375531e-07 -1.120611667309835e-05
  1   1   1   1   0   0 -4.379156367712547e-07 -7.955074171282911e-06
... [truncated]
  2   2   4   4   3   3 -1.240958755455683e-15 1.346747233206165e-13
  2   2   4   4   4   4 -3.640885008022631e-16 4.006450678480949e-14
... [truncated]
```

`DielectricFunctions Nfuns`
Specifies the number of custom dielectric functions (default: *Nfuns = 0*). The subsequent *Nfuns* lines are each read as a string and then interpreted as either (i) a filename with a relative path or (ii) real and imaginary parts of a constant (i.e. wavelength independent) value. Wrap each string in quotation marks, e.g. `"../../epsAg.dat"` or `"2.25d0 0.0d0"`. The files should be in three-column format: the wavelength in nm followed by the real and imaginary parts of the relative dielectric function on each line. The wavelength range in the file must fully contain the range specified by the `Wavelength` keyword, but the values need not correspond exactly as they will be linearly interpolated.

`Medium X`
Sets the real-valued dielectric constant of the host medium (default value is *1.0*). If $X < 0$ then its magnitude is interpreted as a refractive index (s), from which the dielectric constant is calculated as $X = s^2$.

`Wavelength L1 [ L2 n ]`
Without the optional arguments, this keyword changes the default wavelength of 666.0 nm to a new value $L1$. Including the optional arguments will specify a closed interval [$L1$, $L2$] divided into n regular grid spacings, thus producing *n+1* wavelengths.

`Incidence a b c [ p ] / [ na nb nc ]`
or
`Incidence file filename [p]`
This keyword modifies the incident plane-wave. The default travel direction (along z in lab-frame) can be changed by the Euler angles a in the range $[0, 2\pi)$ and b in the range $[0, \pi]$, coinciding with the azimuthal and the polar angles, respectively, of the spherical polar coordinates in the lab frame. In addition, the amplitude vector can then be rotated about the new travel direction by the third Euler angle c in the range $[0, 2\pi)$. All three Euler angles are defined in accordance with the right-hand rule, and the sequence of rotation angles a,b,c corresponds to the intrinsic ZY'Z' convention. That is: rotate by a about the current z-axis, then by b about the new y-axis, and finally by c about the new z-axis.

Near-field and polarimetric calculations, i.e. in modes *M = 1* and *M = 3*, require the polarisation of incident light to be specified. The polarisation is set by integer p, with */p/ = 1* setting linear polarisation, */p/ = 2* setting circular polarisation, and the sign selecting one of the two Jones vectors in each case (positive: x-linear-polarised or R-circular-polarised; negative: y-linear polarised or L-circular-polarised). Note: for a circularly polarised wave travelling along z, right-circular (R) polarisation means that the amplitude vector is rotating clockwise in the xy-plane from the receiver's viewpoint (looking in the negative z direction).

The integer p can be omitted in mode *M = 2*, because its output is always calculated for all four polarisations.

A negative value of argument a, b, and/or c will trigger discretisation of the corresponding angle range to produce $-a$ grid points (resp. $-b$ or $-c$). The grid points are uniformly spaced for the first and the third Euler angles, but for the second (i.e. polar) angle the discretisation is such that the cosine is uniformly spaced. Note that the discretisation is constructed so that orientational averages are computed as a uniformly weighted Riemann sum with the midpoint rule. The weight w_i of each grid point i is simply $w_i = 1/n_{\text{gps}}$, where n_{gps} is the total number of grid points. The range maximum of each angle can be divided by an (optional) integer na, nb, and nc, to help avoid evaluating redundant grid points in the presence of symmetry.

Multiple incidences can also be read from a file, in which case the argument a must be a string starting with ‘f’ or ‘F’, and b must specify the filename. The file’s first line must contain the total incidence count, *ninc*, and the subsequent *ninc* lines each must contain four space-separated values: the three Euler angles (*ai*, *bi*, *ci*) and the weight w_i of each incidence. The weights are only used to compute rotational averages for convenience, which is a common use-case.

In `Mode = 1` (near-field calculations), if `p` is set to `p=1` (default value, linear polarisaiton), the orientation average of the local degree of optical chirality $\langle\mathscr{C}\rangle$ will be calculated for both RCP and LCP (noting that linear polarisation would give 0 everywhere, when orientation-averaged). Since the calculation can be time-consuming, setting `p=+/-2` triggers the calculation for only that specific circular polarisation.

`MultipoleCutoff n1 [ n2 [ t ] ]`
Change the primary multipole cutoff (used for irregular offsetting when staging the linear system) from the default value of 8 to *n1*. Another cutoff (used for regular offsetting when “contracting” the collective T-matrix) can be set to *n2* $>=$ *n1* (equality by default). A relative tolerance 10^t (with $t < 0$ and $t = -8$ by default) is used in the test for convergence of cross-sections with respect to multipole order $n = 1 \dots n_2$ (the summation can terminate below n_2 if the relative tolerance is reached).

`MultipoleSelections Ns`
This keyword defines optional multipole selections for individual T-matrices, and it must be followed by *Ns* lines with two fields: (i) a string *range* specifying the selection range; and (ii) a string *type* specifying the selection type. For example:

```
MultipoleSelections 3
MM1:4_EM1:4_ME1:4_EE1:4  blocks
MM1:0_EM1:15_ME1:8_EE1:0  rows
EM1:1_ME1:1  columns
```

The *range* string must be of the form MM?:?_EM?:?_ME?:?_EE?:?, with the underscores separating the ranges for each T-matrix block (e.g. MM or ME), and each range specified by a closed multipole interval ?:? (e.g. $n$lo:$n$hi = 1:4). No selection will be applied to blocks not included in *range*, so these “missing” blocks will remain unmasked (left "as is" in the original T-matrix). On the other hand, a whole block can be masked (zeroed out) by setting $n$lo $>$ $n$hi (e.g. `MM1:0` will set the whole `MM` block of the T-matrix to 0).

The *type* string must either start from “c”, “r”, or “b”, to indicated that the selection is either applied to T-matrix columns, rows, or both (producing non-zero blocks). To clarify, if *type(1:1)* = *“c”*, then all T-matrix columns corresponding to multipole orders $n <$ $n$lo and $n >$ $n$hi will be set to zero. For *type(1:1)* = *“b”*, columns **and** rows for $n <$ $n$lo and $n >$ $n$hi will be set to zero.

Output control

`OutputFormat F [ filename ]`
If present, the output file format F can be switched between plain text (“TXT”, default) and HDF5 (“HDF5”). With “HDF5”, the results will be stored in a file with name “results.h5”, or a user-specified filename (extension *.h5* added automatically).

`Verbosity L`
Keyword specifying integer-valued verbosity level L. Silent mode ($L = 0$) prints only error statements and warnings. Physical quantities and some status indicators are printed at low verbosity ($L = 1$, default value), with various timings and convergence indicators released at medium verbosity ($L = 2$). The highest level ($L = 3$) is intended for debugging, releasing all print statements throughout the code.

Near-field specific keywords

`SpacePoints filename`
or
`SpacePoints xlo xhi nx ylo yhi ny zlo zhi nz`
Read (from a file) or calculate (on a regular grid) the cartesian coordinates of points in space, where the local field quantities are to be evaluated. The file's first line should contain the total number of space-points, and the subsequent lines must contain the x, y, and z coordinates of each point. A regular grid is specified by a closed interval, e.g. *[xlo, xhi]*, and the number of bins (nx) the interval is to be divided into (thus producing $nx+1$ grid points along that dimension).

`MapQuantity [p] [E] [B] [C]`
Specify the near-field quantities of interest, in `Mode = 1`. Integer argument p selects the raising power applied to the field amplitude $|\mathbf{E}|^p$ or $|\mathbf{B}|^p$. The default is $p = 2$ yielding the field intensity, $p = 1$ is for the field amplitude $|E|$, $p = 4$ for the (approximate) Raman enhancement factor $|E|^4$. Setting $p = 0$ will output the real and imaginary parts of the (vector!) field components at each space-point.

The optional letters $[E]$ $[B]$ $[C]$ (default: E only) determine which of the near-field properties (electric and magnetic fields and normalised value of local degree of optical chirality) will be calculated.

`MapOaQuantity [E] [B] [C]`
This is a keyword applicable in `Mode = 1`, to request the calculation of analytical orientation-averaged near-field quantities $\langle|\mathbf{E}|^2\rangle$, $\langle|\mathbf{B}|^2\rangle$, or $\langle\overline{\mathscr{C}}\rangle$. If this keyword is not included in the input file, by default none will be calculated. Note that the `Incidence` keyword is used to select LCP and RCP (or both).

Polarimetry keywords

`ScatteringAngles a b c / [ na nb nc ]`
This keyword specifies the scattering angles in `Mode = 3` (polarimetry), for the calculation of Stokes scattering vectors at different scattering angles. The parameters have the same interpretation as for `Incidence`.

Multiple scattering angles can also be read from a file, in which case the argument `a` must be a string starting with 'f' or 'F', and `b` must specify the filename. The file's first line must contain the number of scattering angles, *nsca*, and the subsequent *nsca* lines each must contain three space-separated values: the three Euler angles (*ai*, *bi*, *ci*) for each scattering angle.

Advanced use / development

`ScattererCentredCrossSections`
Applicable in `Scheme` 1 and 2. Triggers Stout's formulae for fixed and orientation-averaged cross-sections based on scatterer-centred matrices; otherwise, the default behaviour is to collapse the coefficients to a common origin. Note that this does not affect the calculation of fixed-orientation partial shell absorptions for layered spheres, as they are calculated separately.

`DumpCollectiveTmatrix [ filename ]`
If the collective T-matrix is computed, this keyword will dump it to a file "tmat_col.txt" or a user-specified *filename*. The file format is self-consistent, so that the generated T-matrix can be fed back into TERMS for subsequent calculations.

`DumpPrestagedA`
If present, dumps a sparse-format representation of the full matrix comprising the individual T-matrices after potential masking followed by rotation in their respective frame.

`DumpStagedA`
If present, dumps a sparse-format representation of the full matrix comprising the individual T-matrices along the diagonal blocks, and translation matrices in the off-diagonal blocks. The exact form of this matrix is scheme-dependent.

`DumpScaCoeff`
If present, dumps the scattering coefficients into a file "Sca_coeff" for different incidence angles.

`DumpIncCoeff`
If present, dumps the incident coefficients in to a file "Inc_coeff" for different incidence angles.

`DisableStoutBalancing`
If present, switches off the balancing.

`DisableRTR`
Switches off the three-step translation of T-matrices, where a general translation is decomposed into a rotation, z-axial translation, and then the inverse rotation. Instead, a one-step transformation is performed by pre- or post-multiplying by a single matrix containing the general translation-addition coefficients.

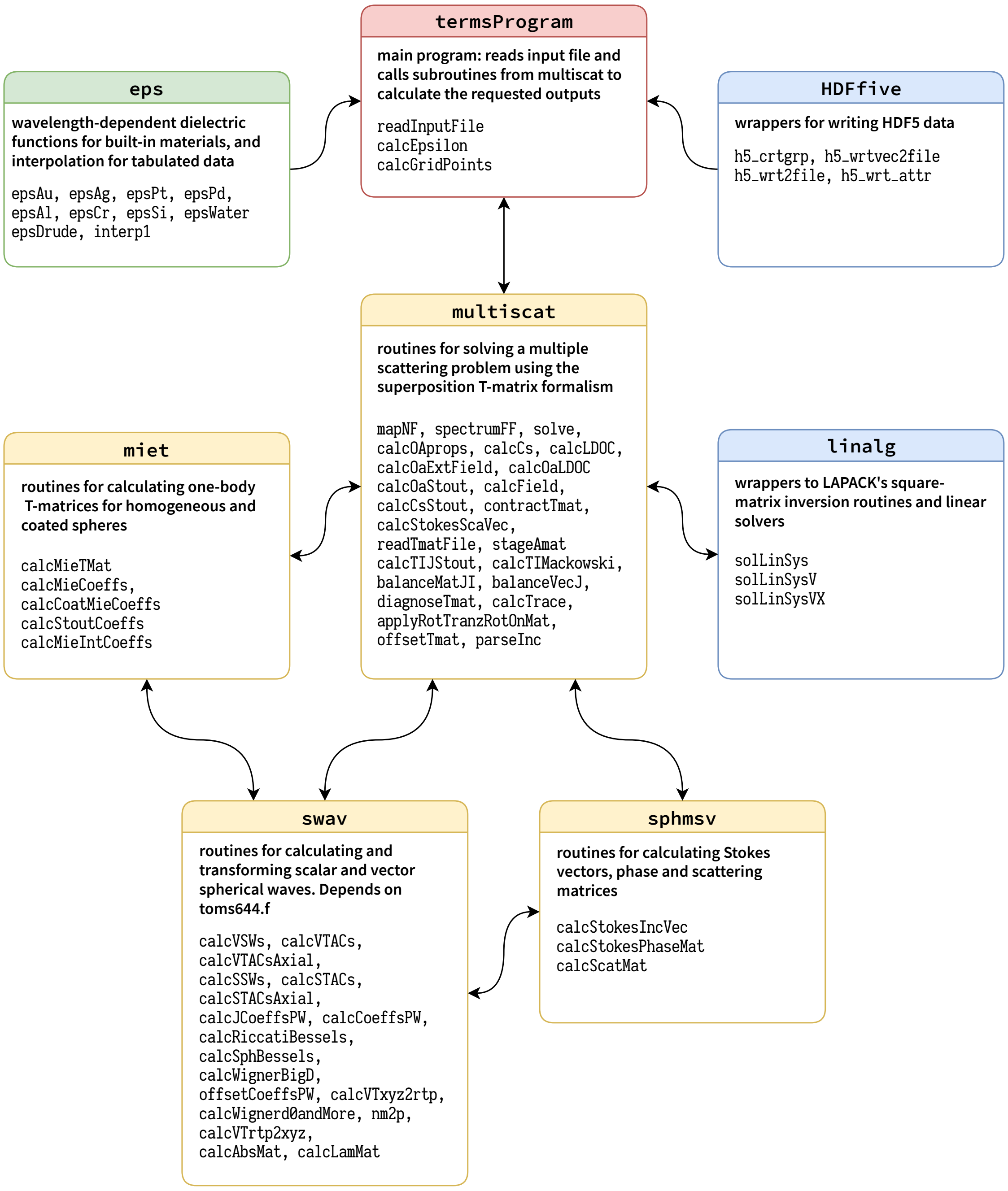
termsProgram
main program: reads input file and calls subroutines from multiscat to calculate the requested outputs
readInputFile
calcEpsilon
calcGridPoints
eps
wavelength-dependent dielectric functions for built-in materials, and interpolation for tabulated data
epsAu, epsAg, epsPt, epsPd,
epsAl, epsCr, epsSi, epsWater
epsDrude, interp1
HDFfive
wrappers for writing HDF5 data
h5_crtgrp, h5_wrtvec2file
h5_wrt2file, h5_wrt_attr
multiscat
routines for solving a multiple scattering problem using the superposition T-matrix formalism
mapNF, spectrumFF, solve,
calcOAprops, calcCs, calcLDOC,
calcOaExtField, calcOaLDOC
calcOaStout, calcField,
calcCsStout, contractTmat,
calcStokesScaVec,
readTmatFile, stageAmat
calcTIJStout, calcTIMackowski,
balanceMatJI, balanceVecJ,
diagnoseTmat, calcTrace,
applyRotTranzRotOnMat,
offsetTmat, parseInc
miet
routines for calculating one-body T-matrices for homogeneous and coated spheres
calcMieTMat
calcMieCoeffs,
calcCoatMieCoeffs
calcStoutCoeffs
calcMieIntCoeffs
linalg
wrappers to LAPACK's square-matrix inversion routines and linear solvers
solLinSys
solLinSysV
solLinSysVX
swav
routines for calculating and transforming scalar and vector spherical waves. Depends on toms644.f
calcVSWs, calcVTACs,
calcVTACsAxial,
calcSSWs, calcSTACs,
calcSTACsAxial,
calcJCoeffsPW, calcCoeffsPW,
calcRiccatiBessels,
calcSphBessels,
calcWignerBigD,
offsetCoeffsPW, calcVTxyz2rtp,
calcWignerd0andMore, nm2p,
calcVTrtp2xyz,
calcAbsMat, calcLamMat
sphmsv
routines for calculating Stokes vectors, phase and scattering matrices
calcStokesIncVec
calcStokesPhaseMat
calcScatMat

Figure A.6: Organigram of the code structure in TERMS, organised in 8 modules. The `termsProgram` module is the general entry point to the program, dispatching the calculations to `multiscat`. In turn, `multiscat` uses subroutines from `miet`, `swav`, `sphmsv`, as well as linear algebra wrappers for LAPACK in `linalg`. The module `eps` provides definitions of common dielectric functions and associated routines, while `HDFfive` provides wrappers for the HDF5 output data format.

A.3. List of subroutines

In this section we briefly describe the main program, its different modules, subroutines, and functions. The arguments of each subroutine or function are shown in parentheses with output arguments highlighted in colour (note that some of them are both input and output arguments).

Main program `termsProgram`

`termsProgram` is the main module, with subroutines listed below; it reads the keywords and their corresponding values in the input file and then calls different subroutines of `multiscat` module for the calculation of requested outputs.

- `readInputFile(inputfile)`
  Reads the input file containing specific keywords and the corresponding parameter values.

- `errorParsingArguments(keyword)`
  If there is an error with the parameter values assigned to a keyword, this subroutine informs the user and stops the program.

- `calcEpsilon()`
  Updates the `escat` array storing the relative dielectric function(s) for each scatterer evaluated at the specified wavelengths.
  `escat` is an array for which the number of rows, columns and the 3^{rd} dimension correspond to the number of shells, scatterers, and wavelengths, respectively.

- `calcGridPoints(points)`
  For a near-field calculation, this subroutine calculates the grid points based on the number of steps, lower and upper values along the desired axes.
  `points(3, nGridPoints)` is an in/output matrix storing the cartesian coordinates (x, y, z) of the grid points.

- `sentence2words(sentence, words, nwords_)`
  Reads each line of the input file as a sentence and splits it into space-separated words.
  `sentence`, `words` are in/output character arrays, and `nwords_` is an optional output integer containing the number of words in the sentence.

- `dumpNFs2TXTFile(filename, incidences, Epower, wavelen, work, Ef, p_label)`
  Exports electric and magnetic near fields into a plain text file. `filename` is the name of the text file. `incidences, Epower, wavelen, work` are the incidence angles, selected power for mapping fields, wavelength, and near-field quantities, respectively. `Ef` is a logical flag which selects either electric field or magnetic field. `p_label` is an integer array indexing the position of each grid point, whether it is inside the surrounding medium or a particle, and in which layer.

- `dumpNFs2HDF5File(fname, groupname, filename, incidences, Epower, wavelen, work, p_label)` Exports electric and magnetic near fields into a HDF5 file. `filename`, `fname`, `groupname` are the names of the HDF5 file, group, and subgroup name, respectively. `incidences, Epower, wavelen, work` are the incidence angles, selected power for mapping fields, wavelength, and near-field quantities, respectively. `Ef` is a logical flag which selects either electric field or magnetic field. `p_label` is an integer array indexing the position of each grid point, whether it is inside the surrounding medium or a particle, and in which layer.

- `countLines(filename) result(nlines)`
  Counts lines in a text file.

`multiscat` module

This module consists of a mix of high-level, core, low-level and supplementary routines for solving a multiple scattering problem using the T-matrix formalism. We list below the subroutines of the `multiscat` module with a brief explanation. A list of common arguments and their brief description is at the end of this section. The other arguments are explained after each subroutine.

- `mapNF(ncut, wavelen, inc,ehost, geometry, scheme, tfiles_, escat_, nselect_, verb_, noRTR_, dump_oaE2, dump_oaB2, field, Bfield, N_OC, orAvextEB_int, oa_ldoc, p_label)`
  Calculates the electric and magnetic near fields, and normalised optical chirality ($\overline{\mathscr{C}}$) for a multiple scattering problem, for different incidence directions and wavelengths, as well as the orientation-averaged value of external $\langle|\mathbf{E}|^2\rangle, \langle|\mathbf{B}|^2\rangle$ and $\langle\overline{\mathscr{C}}\rangle$. `escat_, tfiles_, nselect_, verb_, noRTR_` are optional inputs. `dump_oaE2, dump_oaB2` are logical flags selecting whether the orientation-averaged values $\langle|\mathbf{E}|^2\rangle$ and $\langle|\mathbf{B}|^2\rangle$ will be calculated, respectively.

- `spectrumFF(ncut, wavelen, ehost, geometry, scheme, escat_, tfiles_, nselect_, noRTR_, verb_, sig_oa_, sig_, sig_abs_, jsig_abs_oa)`
  Calculates cross-section spectra for (multiple) fixed orientations, partial absorptions, and orientation-averaged cross-sections for a particle cluster. T-matrices for individual scatterers are either constructed using Mie theory or read from an optional argument `tfiles_`. `escat_, tfiles_, nselect_, verb_, noRTR_` are optional inputs. `jsig_abs_oa` contains the orientation-averaged absorption cross-section of each particle (valid for homogeneous spheres only, at present).

- `solve(wavelen, ehost, geometry, nselect_, scheme_, verb_, noRTR_, TIJ, cJ_, cJint_, csAbs_, ierr_ )`
  This routine is the crux of TERMS and solves a given multiple scattering problem by operating in a specified scheme. `TIJ` is an in/output argument, `cJ_, cJint_, csAbs_` are optional in/output arguments, `nselect_, scheme_, verb_, noRTR_` are optional inputs, and `ierr_` is an optional output. `TIJ` (l_{max} × nscat, l_{max} × nscat) as the input argument stores the T-matrix of nonspherical particles as the diagonal blocks of the matrix, or dielectric values of different shells for spherical particles as the diagonal elements of the matrix. `nscat` is the number of scatterers. This subroutine updates and returns `TIJ` for the whole system as the output. `cJ_(nscat x` l_{max}`, 2, nfi)` as the input argument contains details of the incident field and as a output argument contains incident plane wave coefficients in the first column and scattering coefficients in the second column. `nfi` is the number of incident angles. `cJint_(nscat x` l_{max}`, 4, 2)`: contains the regular and irregular field coefficients for each concentric region inside spherical scatterers. `csAbs_(nscat,4)`: contains absorption cross section inside each shell of each spherical scatterer.

- `stageAmat(scatXYZ, scatMiet, rtr, right_, balance_, verb_, A, Tmats_)`
  Stages a pre-staged matrix A.
  `A (`l_{max} `x nscat,` l_{max} `x nscat)`: an in/output matrix, which must contain 1-body T-matrices in the diagonal blocks on input and is a pre-staged matrix on the output; `right_, balance_, verb_` are optional inputs; `Tmats_(`l_{max}`,` l_{max}`, nscat)`: an optional output matrix which contains the 1-body T-matrix of each particle. `balance_`: a logical input argument which determines whether balancing is applied or not.

- `calcTIJStout(scatXYZ, scatMiet, rtr, TIJ)`
  Calculates the scatterer-centred T-matrix using the recursive scheme presented in Refs. 14 and 15. The relevant equations are 33 and 35 in Ref. 14, and 20, 22 and 24 in 15. `TIJ` is an in/output argument. `TIJ(`l_{max} `x nscat,` l_{max} `x nscat)` as the input argument stores the T-matrix of nonspherical particles as the diagonal blocks, or dielectric values of different shells for spherical particles as diagonal blocks.

- `calcTIMackowski(scatXYZ, scatMiet, rtr, TIJ)`
  Calculates the cluster's T-matrix using Mackowski & Mishchenko's formulation. `TIJ` is an in/output

argument. `TIJ(`l_{max} `x nscat,` l_{max} `x nscat)` as the input argument stores the T-matrix of non-spherical particles as the diagonal blocks, or dielectric values of different shells for spherical particles as diagonal blocks. The output `TIJ` is the scatterer-centred T-matrices calculated using Mackowski & Mishchenko's scheme[16–18].

- `balanceMatJI(j, jregt, iregt, i, rev_, mnq_, Mat)`
Performs balancing on a matrix (`Mat`) using two weights (indexed by j and i). `Mat` is here taken as relating two vectors of VSWF coefficients, c_j (centred at j) and c_i (centred at i), such that $c_j = \mathrm{Mat}\, c_i$. Logical inputs `jregt` and `iregt` specify whether c_j and c_i are regular or not. `Mat` is an in/output argument.

- `balanceVecJ(j, jregt, rev_, Vec)`
Performs balancing on a single vector (`V`) with a weight indexed by j. `V` corresponds here to the VSWF coefficients of particle j. `Vec` is an unbalanced/ balanced vector as the in/output argument. `j` specifies the scatterer.

- `calcCsStout(scatXYZR, aJ, fJ, nmax2_, tol_, verb_, sig)`
Calculates the extinction, scattering and absorption cross-sections from the incident and scattered coefficients using the Stout formulae[14]. `nmax2_, tol_, verb_` are optional inputs and `sig` is an in/output matrix.

- `calcCs(scatXYZR, inc, fJ, nmax2_, tol_, verb_, sig)`
Calculates the extinction, scattering and absorption cross-sections from the incident and scattered coefficients which are collapsed to the common origin. Depending on the dimension of the `sig`, each cross-section is either just a total sum, or resolved into contributions from the multipole orders. `inc`: a vector of incidence angles.

- `calcOAprops(Tmat, rtol_, sigOA, verb_)`
Calculates orientation-averaged cross-sections and circular dichroism (CD) by transforming the T-matrix (`Tmat`) from "parity" (M–N) basis to "helicity" (L–R) basis, following Ref. 21. `rtol_` is an optional input, `verb_` is an optional output, and `sigOA` is an in/output matrix containing orientation-averaged cross-sections and CD in each column for $n = 1, \ldots, n_{max}$.

- `contractTmat(Tin, scatXYZR, rtr, mack_, Tout, verb_)`
Combines the scatterer-centred T-matrices into a common origin; the output will be the collective T-matrix (`Tout`). `verb_` is an optional in/output, `mack_` is an optional logical input to calculate the collective T-matrix based on Mackowski & Mishchenko's scheme[16–18].

- `diagnoseTmat(mode_, verb_, Tmat)`
Determines the value of $n \leq n_{max}$ when $\mathrm{Tr}(\Re(\mathrm{Tcol}))$ converges to $\mathrm{rtol\_G} := 10^{-\mathrm{ncut}(3)}$. If $\mathrm{mode\_} > 0$, also tests for the general symmetry, which applies to all T-matrices. (See equation 5.34 on p. 121 of Mishchenko[4]).

- `calcOaStout(TIJ, scatXYZ, verb_, sigOA, cdOA_, jAbsOA)`
Calculates the orientation-averaged extinction and scattering cross-sections defined in equations 44 and 47 of Stout[14]. The absorption cross-section is then deduced as the difference. `TIJ` is the collective T-matrix, `sigOA(3)` contains orientation-averaged extinction and scattering cross-sections, and `cdOA_` is an optional output containing the corresponding values of CD. `jAbsOA`: contains the orientation-averaged absorption cross-section for each particle.

- `applyRotTranzRotOnMat(vtacs, bigdOP, rightOP, mat)`
Performs the factorised translation of T-matrices when changing origin. Instead of a single multiplication of a T-matrix by a dense matrix containing the general translation-addition coefficients, this routine executes three multiplications by sparse matrices representing 1) a rotation, 2) a translation along the z-axis; and 3) an inverse rotation. This is meant to be more efficient when high multipole

orders are included.
`vtacs(2x pmax,2 x pmax)`: axial VTACs with (m, n, q) indexing, `bigdOP(pmax, pmax)`: optional input for rotation, `rightOP`: an optional logical input argument for applying the product from the right. `mat`: a non/translated matrix as the in/output.

- `calcField(r, geometry, ipwVec, ipwE0, scaCJ, intCJreg_, intCJirr_, scatK_, verb_, reE, imE, reB, imB, reE_sca,imE_sca, reB_sca,imB_sca, p_label)`
  Calculates the electric and magnetic near-field values at the determined grid points.
  `r`: a matrix containing the coordinates of the grid points; `ipwVec(3)`,`ipwE0(3)`: contain the wavevector and amplitude of the incident field, respectively; `scaCJ`: a vector containing scattering coefficients, `intCJreg_, intCJirr_`: contain the regular and irregular parts of the incident field coefficients transformed to the origin of each particle, respectively, `scatK_` is the wavenumber in the host medium, and `reE, imE, reB, imB, reE_sca,imE_sca, reB_sca,imB_sca`: contain real(re) and imaginary(im) parts of the total electric (E) and magnetic (B) fields and the scattered field values at the grid points.

- `dumpTmat(tmat, filename, lambda, eps_med, tol_, verb_)`
  Routine for dumping the collective T-matrix (`tmat`) to a file in the format:

  ```
  s, s', n, n', m, m', T_re, T_im
  ```

  `filename` is an argument of type character corresponding to the name of the output file; `lambda`: the value of wavelength; `eps_med`: the dielectric value of the host medium.

- `dumpMatrix(mat, ofile, tolOP, verb_)`
  Outputs matrix `mat` to a desired optional tolerance (`tolOP`). `ofile`: the name of the output file.

- `offsetTmat(off, miet, rtr, right, bigD_, useD_, balJI_, Tmat)`
  Offsets the supplied T-matrix `Tmat` by `off`, which can be either a square matrix of VTACs or a (note: complex!) displacement vector $k\mathbf{r}(3)$ from which VTACs will be generated. Regular or irregular VTACs will be generated depending on whether $k\mathbf{r}(3)$ is purely real or purely imaginary. If the logical input `miet` is true, Tmat will be treated as diagonal. If the logical input `rtr` is true, then offsetting will be based on factorised translation. If the logical input `right` is true, then offsetting will be done by post-multiplying Tmat from the right. `balJI_` triggers balancing of the VTACs and the T-matrices individually, before offsetting, but currently works only without factorised translation.

- `readTmatFile(filename, unit, wavelen, verb_, Tmat)`
  Reads a T-matrix from the input file (`filename`) and import it into the matrix `Tmat`. `unit`: an integer indexing the name of the T-matrix file. `wavelen` is the value of the wavelength.

- `parseInc(inc, verb_, inc_dirn, inc_ampl)`
  Calculates the amplitude and direction vector of the incident plane wave based on the input Euler angles (α, β, γ). `inc_dirn` and `inc_ampl` are vectors containing the wavevector and amplitude of the incident electric field in cartesian coordinates, respectively. `inc` is a vector consisting of polarisation type and Euler angles of the incidence direction.

- `calcStokesScaVec(sca_angles, inc2, ncut, wavelen, ehost, geometry, scheme, tfiles_, escat_, nselect_, noRTR_, verb_, StokesPhaseMat, StokesScaVec, diff_sca)`
  Calculates the Stokes phase matrix (`StokesPhaseMat`), Stokes scattering vector (`StokesScaVec`), and differential scattering cross-sections (`diff_sca`).
  `sca_angles` is a matrix of desired scattering angles; if it is not specified in the input file, they are taken equal to the incidence angles. `inc2` is a matrix containing incidence angles.

- `calcLDOC(Ef, Bf, verb_, N_OpC)`
  Calculates the normalised optical chirality ($\overline{\mathscr{C}}$) relative to the optical chirality of circularly polarised light. `Ef`, `Bf`, and `N_OpC` are matrices containing the electric and magnetic field, and $\overline{\mathscr{C}}$ values at the

grid points, respectively.

- `calcOaExtField(r, geometry, TIJ, lambda, ehost, escat, p_label, verb_, orEB2)`
  Calculates the orientation average of the total external electric and magnetic field intensities. `r` is a matrix containing the cartesian coordinates of the grid points. `TIJ` is the scatterer-centred T-matrix of the cluster. `orEB2` is a vector containing the value of orientation-averaged external electric and magnetic field intensities at the grid points.

- `calcOaLDOC(pol_type, r, geometry, TIJ, verb_, Or_OC)`
  Calculates the orientation average of normalised optical chirality $\langle\overline{\mathscr{C}}\rangle$. `pol_type` is the polarisation type, `r` is a matrix containing the cartesian coordinate of the grid points. `TIJ` is the scatterer-centred T-matrix of the structure.

- `calcTrace(TRANSA, TRANSB, A, B, tr)`
  Calculates the trace of a product of two matrices, `op(A)*op(B)`. The input characters `TRANSA` and `TRANSB` determine the operation `op`, following the convention of BLAS' `gemm`. Specifically, `op = 'N'` corresponds to `op(A)` = `A` (no operation), whereas `op = 'C'` corresponds to `op(A)` = $\mathtt{A}^\dagger$.

- `RotMatX(ang) result(rotMat)`
  Calculates a rotation matrix along the x axis using input argument angle(`ang`).

- `RotMatY(ang) result(rotMat)`
  Calculates a rotation matrix along the y axis using input argument angle(`ang`).

- `RotMatZ(ang) result(rotMat)`
  Calculates a rotation matrix along the z axis using input argument angle(`ang`).

- `rotZYZmat(angles) result(mat)`
  Calculates rotation matrix `mat` for ZY'Z', using the Euler `angles`$=(\alpha, \beta, \gamma)$

List of common arguments

- `acs_int_`: a matrix containing partial internal absorption inside each scatterer and for each shell.
- `aJ(nscat` $\times\, l_{max}$`)`, `fJ(nscat` $\times\, l_{max}$`)`: contains incident and scattering coefficients.
- `Bfield`: contains the real and imaginary parts of the magnetic near field at the specified grid points, wavelengths, and incidence.
- `ehost`: a vector of dielectric permittivity of the host medium at specified wavelengths.
- `escat_(nscat, 4, size(wavelen))`: depending on the number of wavelengths, it is a 2D or 3D array of dielectric values for each scatterer, for each shell and wavelength.
- `field`: contains the real and imaginary parts of the electric near field at the specified grid points, wavelengths, and incidence.
- `geometry`: a matrix containing physical information of different scatterers such as centre, dimensions and direction.
- `ierr_`: an integer value (0 or 1 or 2); 0 indicates solving was successful, 1 means there is an error in processing arguments, and 2 means an error in prestaging, staging, or solving/inverting $Ax = b$.
- `iregt`: logical input, specifies whether vectors are regular or not.
- `jregt`: logical input, specifies whether vectors are regular or not.

- `mnq_`: an optional logical argument which is false by default, but if true will change the indexing convention from (q,n,m) to (m,n,q), which is used to make the z-axial VTACs block-diagonal. Note that index q corresponds to s in this user guide.
- `ncut`: a vector in the form $[n_1,\ n_2,\ \mathrm{tol}]$, which contains the values corresponding to the keyword `"MultipoleCutoff"`. Default values: $[8, 8, -8]$.
- `nmax2_`: an integer value equals to ncut(2).
- `noRTR_`: an optional input with logical value `.true.` or `.false.` for the keyword `DisableRTR`. Default: `.false.`.
- `nselect_`: an optional input matrix which includes information about multipole selection for different scatterers.
- `oa_ldoc` (npts $\times$ 4 $\times$ nwavelen): contains the orientation averaged value of $\overline{\mathscr{C}}$ at different grid points and wavelengths.
- `orAvextE_int`(npts $\times$ nwavelen): contains the orientation averaged electric field intensity values at different grid points and wavelengths.
- `p_label`: a matrix determining the position of each grid point, whether it is inside the surrounding medium or particles, and in which layer.
- `rev_`: an optional logical input which is false by default; triggers the reverse of balancing – "unbalancing".
- `right_`: a logical input. According to (Eqs. 44, 45) there are two ways for obtaining the `TIJ` matrix. This argument determines whether the product is taken from the left or from the right.
- `rtr`: a logical input that is the reverse of `noRTR_`.
- `scatMiet`(nscat): a logical vector with `.true.` and `.false.` values, determining whether a scatterer is spherical or not.
- `scatXYZ`(3,nscat): a matrix containing the cartesian coordinates (in lab frame) of the particle's centre.
- `scatXYZR`(4,nscat): a matrix containing the cartesian coordinates (in lab frame) and the radius of the smallest circumscribed sphere of each particle.
- `scheme, scheme_`: an integer value specifying the selected scheme.
- `sig_`: a matrix containing cross-sections (Extinction, Scattering, Absorption) for different polarisation(s), wavelength(s), and incidence(s).
- `sig_abs_`(4 $\times$ nscat $\times$ 4 $\times$ nwavelen $\times$ nfi): a 5D array containing absorption cross-sections inside each shell for each scatterer for 4 Jones vectors, different wavelengths and different incidence directions.
- `sig_oa_`(6$\times$n$\times$nwavelen): a matrix consisting of orientation-averaged cross-sections and CD at different wavelengtha. The first column gives the values for n_{max} and other columns contain values for different value of $n = 1, \dots,$ ncut(2).
- `tfiles_`: a matrix of character type, includes the T-matrix filename and filepath for non-spherical scatterers.
- `tol_, rtol_`: a real value `rtol_G = ` $10^{\texttt{ncut(3)}}$.
- `N_OC`: contains $\overline{\mathscr{C}}$ at the specified grid points, wavelengths, and incidence.
- `verb_`: an integer variable containing the verbosity value ($\in [0, 1, 2, 3]$) (the default value is 1).
- `wavelen`: a vector of specified wavelength(s).

`miet` module

This module contains routines for calculating one-body T-matrices (currently limited to spherical scatterers, using Mie theory).

- `calcMieTMat(x, s, zeropad_, tmat)`
  Calculates the diagonal T-matrix of a spherical scatterer for a given size parameter $x = kR$, relative refractive indices ($s = k_{in}/k_{out}$); `zeropad_=nmax` maximum value of the multipole index inferred from `tmat`'s dimensions.

- `calcMieCoeffs(x, s, gammas, deltas)`
  Calculates the Mie coefficients for a spherical scatterer as defined by equations H.46 and H.47 of Ref. 19. The coefficients are interpreted as magnetic and electric susceptibilities (Γ_n and Δ_n, respectively) of the scattered field. Note the relation to standard Mie coefficients[4]: $a_n = -\Delta_n$ and $b_n = -\Gamma_n$.

- `calcCoatMieCoeffs(x, s, gammas, deltas)`
  Calculates the Mie coefficients for a coated sphere based on the equations H.110 and H.113 of Ref. 19.

- `calcStoutCoeffs(x, rri, nmax, Cn, Dn)`
  Calculates the `Cn,Dn` coefficients as defined by equation (50) in Stout[14]. These coefficients are used to calculate absorption cross-sections. `rri` is the relative refractive index, `nmax` is the maximum value of the multipole index.

- `calcMieIntCoeffs(a, k, scaCoeffs, intCoeffsReg, intCoeffsIrr, csAbs)`
  Calculates the regular and irregular VSWF coefficients for the field inside each concentric region of a (layered) Mie scatterer. The formulae are based on Eqs. H.117–H.123 of Ref. 19. `a`, `k`, and `scaCoeffs` are vectors of the radius of the concentric interfaces, relative refractive index, and scattered field coefficients for the host medium, respectively. `intCoeffsReg` and `intCoeffsIrr` are matrices of regular and irregular field coefficients for each concentric region inside the scatterers and `csAbs` contains the partial absorptions calculated using equation (29) in Mackowski[20].

`swav` module

This module contains routines for calculating and transforming scalar (SSWs) and vector spherical waves (VSWFs). It depends on `Amos (toms644.f)` to calculate spherical Bessel and Hankel functions using recurrence. In order to limit redundancy, parameter definitions are renewed only where they are changed.

- `calcVTACs(r0, k, regt, vtacs)`
  Calculates the irregular (if `regt=.false.`) or the regular (if `regt=.true.`) vector translation-addition coefficients for a given `kr0`.
  `r0` is a relative position vector, `k` is the wavenumber, `regt` is a logical argument which determines the type: regular or irregular, and `vtacs(1:2*pmax,1:2*pmax)` is the input/output array.

- `calcSTACs(r0, k, pmax, regt, scoeff)`
  Calculates the scalar translation-addition coefficients.($\alpha_{nu,mu;n,m}$ or $\beta_{nu,mu;n,m}$). The output corresponds to the scalar translation-addition coefficients α(irregular, for `regt=.false.`) or β(regular, for `regt=.true.`).
  `pmax` is a maximal composite index and `scoeff(0:pmax,0:pmax)` is the coefficients matrix.

- `calcVTACsAxial(r0, k, pmax, regt, flip, mqn_, vtacs)`
  Calculates the irregular (if `regt=.false.`) or the regular (if `regt=.true.`) vector translation-addition coefficients for a given `kr0`, along the z-axis.
  `r0` is the z-axial displacement distance, flip is a logical argument, `mqn_` is a logical argument for changing from `qnm` to `mqn` indexing, and `vtacs(1:2*pmax,1:2*pmax)` is the matrix of coefficients.

- `calcSTACsAxial(r0, k, pmax, regt, flip, stacs)`
  Calculates the normalised scalar translation-addition coefficients along the z-axis for a given `kr0`.
  `r0` is a displacement distance and `stacs(0:pmax,0:pmax)` is the coefficients matrix corresponding to α (irregular, for `regt=.false.`) or β.

- `calcVSWs(r, k, pmax, regt, cart, waves, wavesB)`
  Calculates (at `r`) the normalised vector spherical waves, M_{nm} and N_{nm} for evaluation of electric and magnetic fields
  `r(3)` is the cartesian coordinate of a point in 3D; `cart` is a logical argument which triggers conversion to cartesian coordinates; `waves(2*pmax,3)` contains elements (M_{nm} and N_{nm}) of the abstract column vector defined in Eq. B1 of Ref. 14 and `wavesB(2*pmax,3)` is similar to `waves`, only swapping the position of M_{nm} and N_{nm} and multiplying by $-ik$ for calculation of the magnetic field.

- `calcSSWs(xyz, k, pmax, regt, psi)`
  Calculates (at `xyz`) the scalar spherical waves ψ_{nm}.
  `xyz` is the cartesian coordinates of a point in 3D; `psi(0:pmax)` contains elements of the spherical waves ψ_{nm} as defined by equation 13a in Chew[55].

- `calcJCoeffsPW(ipwE0, kVec, xyz, ipwCoeffsJ)`
  Translates the supplied `ipwCoeffs` coefficients to different centres for an incident plane wave.
  `ipwE0(3)` is the incident plane wave's amplitude vector, `kVec(3)` is the incident wave vector, `xyz(3,nscat)` is a matrix containing the centre of different scatterers, and `ipwCoeffsJ`(nscat $\times$ lmax) is a vector containing the translated incident plane wave coefficients to the centre of different scatterers (according to equation 38 of Ref. 14).

- `calcCoeffsPW(ipwE0, ipwDirn, ipwCoeffs)`
  Calculates the coefficients for expressing an incident plane wave in terms of vector spherical waves M_{nm} and N_{nm}.
  `ipwDirn`(3) is the normalised direction vector of the incident plane wave and `ipwCoeffs`(2*pmax) contains coefficients for expressing an incident plane wave in terms of vector spherical waves M_{nm} and N_{nm}, up to a maximum n_{max}. Follow equations C.57-59 on p.377 of Ref. 4.

- `offsetCoeffsPW(a, kVec, xyzr, aJ)`
  Translates the VSWF coefficients (`a`) of an incident plane wave (centred at the origin) to another origin.
  `a`(l_{max}) contains coefficients for a regular VSWF expansion centred at the origin for an incident plane wave, `xyzr` includes centres of different scatterers, and `aJ` contains scatterer centred coefficients.

- `calcWignerBigD(angles, pmax, bigD)`
  Calculates the Wigner D-functions ($D^s_{m,n}(\alpha, \beta, \gamma)$).
  `angles`(3) includes (α, β, γ) in radians and `bigD`(pmax,pmax) contains Wigner D-coefficients.

- `calcWignerLittled(theta, pmax, d)`
  Calculates the Wigner d-functions ($d^s_{m,n}(\theta)$).
  `theta` is angle in radians and `d`(0:pmax,0:pmax) are values for $d^s_{m,n}$ in block diagonal matrix form.

- `calcWignerd0andMore(x, pmax, d, pi, tau)`
  Calculates the Wigner d-functions for $n = 0$ and also computes the derivative functions for optional outputs `pi` and `tau`.
  `x` is $\cos(\theta)$, `d`(0:pmax), `pi`(0:pmax), `tau`(0:pmax) contain values for $d^s_{m,0}$, $\pi_{m,s}$, and $\tau_{m,s}$ respectively.

- `calcRiccatiBessels(z, nmax, regt, f, df)`
  Calculates the Riccati-Bessel functions ψ_n (if `regt=.true.`) or ξ_n (`regt=.false.`), and their derivatives, for $n = 1, \ldots, n_{\max}$.
  `z` is a scalar complex argument, `f`(1:nmax) is a matrix containing Riccati-Bessel functions $\psi_n(z) = z * j_n(z)$ or $\xi_n(z) = z * h_n(z)$ for $n = 1, \ldots, n_{\max}$, and `df`(1:nmax) are the corresponding derivatives of `f`.

- `calcSphBessels(z, nmax, regt, bes)`
  A wrapper routine for computing spherical Bessel/Hankel functions of the first kind for a complex argument `z`.
  `bes(0:nmax)` contain Bessel ($J_{n+1/2}$) or Hankel ($H_{n+1/2}$) function (of 1^{st} kind) values for $n = 0, \ldots, n_{\max}$ for a complex argument `z`.

- `xyz2rtp(xyz, rtp, cth)`
  Transforms the cartesian coordinates `(x,y,z)` of a point in 3D space to spherical polar coordinates (r, θ, ϕ)
  `xyz(3)` is a vector of cartesian coordinates, `rtp(3)` is a vector of spherical polar coordinates, and `cth` is $\cos(\theta)$.

- `rtp2xyz(rtp, xyz)`
  The inverse of `xyz2rtp`. Transforms the spherical polar coordinates (r, θ, ϕ) of a point in 3D space to cartesian coordinates (x, y, z).

- `calcVTrtp2xyz(rtp, transform)`
  Calculates the matrix of transformation from a vector in spherical coordinates to a vector in cartesian coordinates at point (r, θ, ϕ) (in spherical polar coordinates).

- `calcVTxyz2rtp(rtp, transform)`
  The inverse of `calcVTrtp2xyz`. Calculates the matrix of transformation from a vector in cartesian coordinates to a vector in spherical coordinates at point (r, θ, ϕ) (in spherical polar coordinates).

- `calcAbsMat(Xi, ro, mat)`
  Calculates the absorption matrix $\Gamma_j = \texttt{mat}(\texttt{l}_{\texttt{max}}, \texttt{l}_{\texttt{max}})$ for the input arguments `Xi` and `ro` (Eq. (49) of Ref. 14). Γ_j is used in the evaluation of the orientation-averaged absorption cross-section inside each particle.

- `calcLamMat(Xi, ro, mat)`
  Calculates the "Lambda" matrix $\Lambda_j = \texttt{mat}(\texttt{l}_{\texttt{max}}, \texttt{l}_{\texttt{max}})$ for the input arguments `Xi` and `ro` (Eq. (53) of Ref. 14). Λ_j is used in the evaluation of the orientation-averaged internal electric field inside homogeneous spheres.

- `nm2p(n, m, l)`
  Calculates a generalised index `l=n(n+1)+m`, for a unique `(n,m)`, (Vector spherical harmonics are spanned by two indices: n and m, such that $0 \le n \le n_{max}$ and $-n \le m \le n$).
  `n,m,l` are integers.

- `p2nm(p, n, m)`
  Calculates unique `(n,m)` from a given composite index `p`.
  `p` is a real value.

- `nm2pv2(n, m, p)`
  Some recurrences are defined only for $m \ge 0$, in which case we shall use a second version of the composite index $p_{v2} = n(n+1)/2 + m$.

- `testPmax(name, pmax, nmax)`
  Tests `pmax` for commensurability, i.e. is $p_{max} == n_{max}(n_{max} + 2)$ and $n_{max} = m_{max}$? If not, then the program will be stopped.

`sphmsv` module

This module contains routines for calculating Stokes incident vector, Stokes phase matrix and scattering matrix for an input T-matrix. The formulae are based on Mishchenko[4].

- `calcStokesIncVec(ehost_, ipwDirn_, ipwAmpl_, verb_, Stokes_Vec)`
  Calculates the Stokes incident vector `Stokes_Vec`.

- `calcStokesPhaseMat(SMat, verb_, Z)`
  Calculates the Stokes phase matrix for the specified incident and scattered angles. `SMat(2,2)` and `Z(4,4)` are the scattering and Stokes phase matrices which follow Eqs. (5.11-14) and (2.106-121) of Ref. 4.

- `calcScatMat(tmat, host_K, spwDirn_, ipwDirn_, verb_, SMat)`
  Calculates the scattering matrix using the T-matrix, for the specified incident and scattering angles.

`linalg` module

This module contains wrappers to drive LAPACK's square-matrix inversion routines and linear solvers.

- `invSqrMat(trans_, verb_ A)`
  Calculates inverse of a complex-valued square matrix `A(n,n)`, using the ZGETRF and ZGETRI routines in LAPACK. `A` is overwritten by inv(A) on the output. `trans_` is an optional logical input, in case `.true.` the routine considers transpose of A and finally returns the transpose of the inverted matrix as the output. `verb_`: an optional input of the verbosity value.

- `solLinSys(isol_, verb_, A, X)`
  Solves a complex-valued linear system of equations $Ax = b$, where `A`(n,n) is a square matrix, `b`(n) is a known vector, and `x`(n) is the vector to be determined. Depending on the value `isol_`, calls `solLinSysV` or `solLinSysVX`. Both `A` and `X` are overwritten on output.

- `solLinSysV(verb_, A, X)`
  For solving a linear system, uses LAPACK's "simple" driver ZGESV.

- `solLinSysVX(verb_, A, X)`
  For solving a linear system, uses LAPACK's "simple" driver ZGESVX.

`eps` module

This module contains wavelength-dependent dielectric functions `epsXX(lambda)` for various materials including Au, Ag, Al, Cr, Pd, Pt, Si, and Water).

- `interp1( x1, y1, x2, y2 )`
  Calculates the interpolated data `y2` using the input values `x1,y1` at the points `x2`.

- `epsAu(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of gold. This function uses the analytical expression given in Eq. (E.2) of Ref. 19.

- `epsAg(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of silver. This function uses the analytical expression given in Eq. (E.1) of Ref. 19.

- `epsPt(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of a Lorentz-Drude metal, with the parameters for Pt[65].

- `epsPd(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of a Lorentz-Drude metal, with the parameters for Pd[65].

- `epsSi(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of Silicon in the range 206.6 nm to 1200.0 nm interpolated from[66].

- `epsAl(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of Aluminum in the range 103.32 nm to 2755.2 nm[67].

- `epsCr(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of Aluminum in the range 100.8 nm to 31 μm, from the tabulated data in Ref. 68 pages: 382-385.

- `epsWater(wavelength) result(eps)`
  Returns the wavelength-dependent relative dielectric function of Water at temperature 20^{o}C in the range 200 nm to 3000 nm[69].

- `epsDrude(wavelength, eps_infty, lambda_p, mu_p) result(eps)`
  Returns the wavelength-dependent relative dielectric function of a Drude metal. The analytical expression is given in Eq. (3.2) of Ref. 19.

`HDFfive` module

This module contains subroutines for reading and writing data in HDF5 format.

- `h5_crtgrp(filename_, main_grpname, subgrpsname)`
  This subroutine creates subgroups in an existing group.

- `h5_wrtvec2file(filename_, groupname, dsetname, dset_data)`
  This subroutine writes vector data in a dataset in an existing group.

- `h5_wrt2file(filename_, groupname, dsetname, dset_data)`
  This subroutine writes data in a dataset in an existing group.

- `h5_wrt_attr(attribute, dataset_id)`
  This subroutine adds an attribute to an existing dataset, typically a brief explanation about the contents of the dataset.