The T-matrix code Tsym for homogeneous dielectric particles with finite symmetries

Citation for the original published paper (version of record):
Kahnert, M. (2013). The T-matrix code Tsym for homogeneous dielectric particles with finite symmetries. Journal of Quantitative Spectroscopy and Radiative Transfer, 123: 62-78.
http://dx.doi.org/10.1016/j.jqsrt.2012.12.019

The T-matrix code *Tsym* for homogeneous dielectric particles with finite symmetries

Michael Kahnert[a,b]

[a]*Swedish Meteorological and Hydrological Institute, Folkborgsvägen 17, SE-601 76 Norrköping, Sweden*
[b]*Chalmers University of Technology, Department of Earth and Space Science, SE-412 96 Gothenburg, Sweden*

Abstract

A T-matrix code tailored to non-axisymmetric particles with finite symmetries is described. The code exploits geometric symmetries of particles by use of group theoretical methods. Commutation relations of the T-matrix are implemented for reducing CPU-time requirements. Irreducible representations of finite groups are employed for alleviating ill-conditioning problems in numerical computations. Further, an iterative T-matrix method for particles with small-scale surface perturbations is implemented. The code can compute both differential and integrated optical properties of particles in either fixed or random orientation. Methods for testing the convergence and correctness of the computational results are discussed. The package also includes a database of pre-computed group-character tables, as well as an interface to the GAP programming language for computational group theory. The code can be downloaded at `http://www.rss.chalmers.se/~kahnert/Tsym.html`.

Keywords: Scattering, T-matrix, mineral dust, cosmic dust, ice clouds

1. Introduction

The T-matrix formulation of electromagnetic and acoustic scattering, which was first introduced by Waterman [1], has become the basis for several numerical approaches for numerically solving scattering problems described by Helmoltz' equation. Several specific methods have been developed for

Email address: `michael.kahnert@smhi.se` (Michael Kahnert)

computing T-matrices, such as the null-field method (also known as the extended boundary condition method), which was the method originally proposed by Waterman. Other methods are based on, e.g., volume-integral equation methods [2, 3], the separation of variables method [4], the generalised point-matching method [5], or the superposition T-matrix method [6]. For star-shaped particles, the null-field method is by far the most widely used T-matrix method in numerical computations. (A star-shaped particle has a boundary surface that can be parameterised in spherical coordinates.) Several implementations of the method for axisymmetric [7, 8] and non-axisymmetric particles [9] are publicly available; they can be found on the light-scattering information portal ScattPort [10, 11].

The T-matrix formalism has a theoretical advantages over other numerical methods for solving the light-scattering problem, which results in two main practical advantages. The T-matrix is a property of the particle that is decoupled from the incident field; it contains the complete information about a particle's absorption and scattering properties at a given wavelength. Thus, for any given particle and wavelength, the T-matrix is computed once and for all; all optical properties of interest, such as differential and total scattering cross sections, can be computed from that T-matrix. One resulting practical advantage is that the T-matrix can be rotated [12], so that optical properties of particles in any fixed orientation, as well as ensemble-averaged optical properties of particles in random orientations, can be computed analytically from the T-matrix [13, 14]. By contrast, other numerical methods that are not based on the T-matrix concept compute the scattered field for one specific incident field and one specific particle orientation; if the optical properties of particles in different or random orientations are needed, then the computations need to be repeated for each new particle orientation.

The other practical advantage is that the T-matrix formulation lends itself easily to exploiting geometrical symmetries of the scatterer [15, 16]. This, too, is a direct consequence of the T-matrix being a pure property of the particle independent of the form or direction of the incident field. The exploitation of symmetries is the only known approach that is able to *substantially* reduce CPU-time requirements in numerical light-scattering computations (e.g. [16, 17]). It can also greatly alleviate ill-conditioning problems [18, 19] that have limited the range of applicability of T-matrix computations.

The code *Tsym* (pronounced "tee-simm") presented in this tutorial can be downloaded via the *Tsym* homepage [20]. It has been specifically developed for testing the use of group theory in T-matrix computations. The group

theoretical methods that are implemented in this code are the commutation relations of the T-matrix and the irreducible representations of finite groups. The former were first derived in [15] and later extended in [18], and the latter method was first formulated for T-matrix computations in [18]. The *Tsym* code is, to the best of my knowledge, the only T-matrix code that fully exploits irreducible representations of finite symmetry groups. Apart from group theoretical methods, the code has also been extended for testing a Lippmann-Schwinger T-matrix equation for particles with small-scale surface roughness [21, 17], and for investigating different morphological models for surface roughness effects [22].

A main practical disadvantage of *Tsym* is its low degree of user-friendliness. Many other light-scattering codes have been geared to be used by a large group of users with varying degree of background knowledge about numerical light-scattering methods. T-matrix codes for axisymmetric particles (e.g. [7, 8]) are often particularly user-friendly, since they contain automatic routines for ensuring the convergence of the computational results. By contrast, *Tsym* is a research code that has mainly been created for testing new theoretical and numerical approaches in T-matrix computation; no efforts have been invested to make the code foolproof. The code places all the responsibility for ensuring and testing the correctness of the results on the user. Thus, the program will be most useful for highly experienced users and program developers. It can hardly be emphasised enough that newcomers and users with little background knowledge of T-matrix theory are not encouraged to use this code. This applies specifically to those users who wish to quickly obtain computational results for some specific problem, and who can spend very limited time for thoroughly familiarising themselves with the code and its underlying theory.

The article is organised as follows. Section 2 briefly sketches the theoretical basics of the T-matrix approach and symmetries. Section 3 describes the code, its structure, and use. Section 4 provides the essential information on how to test the convergence of the computational results. Several utilities that are contained in the *Tsym* package are described in Sect. 5. A short summary is given in Sect. 6.

2. T-matrix formulation of electromagnetic scattering

In the T-matrix approach the incident, scattered, and internal fields are expanded in the basis of vector spherical wave functions $\vec{\Psi}^{(j)}_{n,m,\tau}$ according to

$$\mathbf{E}^{\mathrm{inc}}(\mathbf{r}) = \sum_{n=1}^{\infty}\sum_{m=-n}^{n}\sum_{\tau=1}^{2} a_{n,m,\tau}\vec{\Psi}^{(1)}_{n,m,\tau}(k,\mathbf{r}) \tag{1}$$

$$\mathbf{E}^{\mathrm{sca}}(\mathbf{r}) = \sum_{n=1}^{\infty}\sum_{m=-n}^{n}\sum_{\tau=1}^{2} p_{n,m,\tau}\vec{\Psi}^{(3)}_{n,m,\tau}(k,\mathbf{r}). \tag{2}$$

$$\mathbf{E}^{\mathrm{int}}(\mathbf{r}) = \sum_{n=1}^{\infty}\sum_{m=-n}^{n}\sum_{\tau=1}^{2} c_{n,m,\tau}\vec{\Psi}^{(1)}_{n,m,\tau}(k_s,\mathbf{r}). \tag{3}$$

$$\tag{4}$$

k and k_s are the wavenumbers in the surrounding medium and inside the particle, respectively. The indices n, m, and τ denote the degree, order, and mode of the wave functions. The superscript $j = 1$ denotes vector spherical wave functions that are regular at the origin, while $j = 3$ stands for outgoing wave functions that satisfy the radiation condition at infinity. The linearity of the boundary conditions results in linear relations among the expansion coefficients in the form

$$a_{n,m,\tau} = \sum_{n'=1}^{\infty}\sum_{m'=-n'}^{n'}\sum_{\tau'=1}^{2} Q_{n,m,\tau,n',m',\tau'}\, c_{n',m',\tau'} \tag{5}$$

$$p_{n,m,\tau} = -\sum_{n'=1}^{\infty}\sum_{m'=-n'}^{n'}\sum_{\tau'=1}^{2} RgQ_{n,m,\tau,n',m',\tau'}\, c_{n',m',\tau'} \tag{6}$$

$$p_{n,m,\tau} = \sum_{n'=1}^{\infty}\sum_{m'=-n'}^{n'}\sum_{\tau'=1}^{2} T_{n,m,\tau,n',m',\tau'}\, a_{n',m',\tau'}. \tag{7}$$

The quantity of interest is the T-matrix, which expresses the linear relation between the expansion coefficients of the unknown scattered field in terms of the known coefficients of the incident field. Combining the last three equations, one obtains

$$T_{n,m,\tau,n',m',\tau'} = -\sum_{n''=1}^{\infty}\sum_{m''=-n''}^{n''}\sum_{\tau''=1}^{2} RgQ_{n,m,\tau,n'',m'',\tau''}Q^{-1}_{n'',m'',\tau'',n',m',\tau'}. \tag{8}$$

From the T-matrix all optical properties of interest can be computed. Formally, the T-matrix has infinite dimension. In practice, the infinite sum, and thus the T-matrix, needs to be truncated at some finite value $n = n_{\max}$, i.e.

$$\sum_{n=1}^{\infty}\sum_{m=-n}^{n}\sum_{\tau=1}^{2}\cdots \approx \sum_{n=1}^{n_{\max}}\sum_{m=-n}^{n}\sum_{\tau=1}^{2}\cdots. \tag{9}$$

Often, it is even possible to limit the summation over m to some range $m = -m_{\max}, \ldots, m_{\max}$, where $m_{max} \leq n$. One can rearrange the summation according to

$$\sum_{n=1}^{\infty}\sum_{m=-n}^{n}\sum_{\tau=1}^{2}\cdots \approx \sum_{m=-m_{\max}}^{m_{\max}}\sum_{n=\max\{1,|m|\}}^{n_{\max}}\sum_{\tau=1}^{2}\cdots. \tag{10}$$

The truncation parameters $n_{\max}$ and $m_{\max}$ are the most important parameters that determine the numerical accuracy of T-matrix computations.

In Waterman's null-field method [1], the T-matrix is computed from the matrices $\mathbf{Q}$ and $Rg\mathbf{Q}$, as can be seen in Eq. (8). The elements of these two matrices are calculated from surface integrals over various vector products of vector spherical wave functions. The numerical evaluation of these integrals uses a Gauss-Legendre double quadrature, i.e.

$$\int_0^{\pi}\sin\theta \mathrm{d}\theta \int_0^{2\pi}\mathrm{d}\phi\, f(\theta,\phi) \approx \sum_{i=1}^{N_\theta} w_i^{\theta}\sin\theta_i \sum_{j=1}^{N_\phi} w_j^{\phi} f(\theta_i,\phi_j), \tag{11}$$

where the integrand f is a function of the spherical coordinates θ and ϕ. w_i^{θ} and w_j^{ϕ} denote the integration weights, and N_θ and N_ϕ are the number of polar and azimuthal Gauss-Legendre quadrature points, respectively. The main practical problem in T-matrix computations is to determine the convergence-controlling parameters $n_{\max}$, $m_{\max}$, N_θ, and N_ϕ. One usually wants to choose these parameters as large as necessary and as small as possible. At too small values the computations may not yield convergent results, while larger values require more computation time.

2.1. Symmetries

T-matrix computations for particles with geometric symmetries can be substantially expedited by exploiting the so-called commutation relations of the the T-matrix. For instance, a particle with hexagonal symmetry, such as

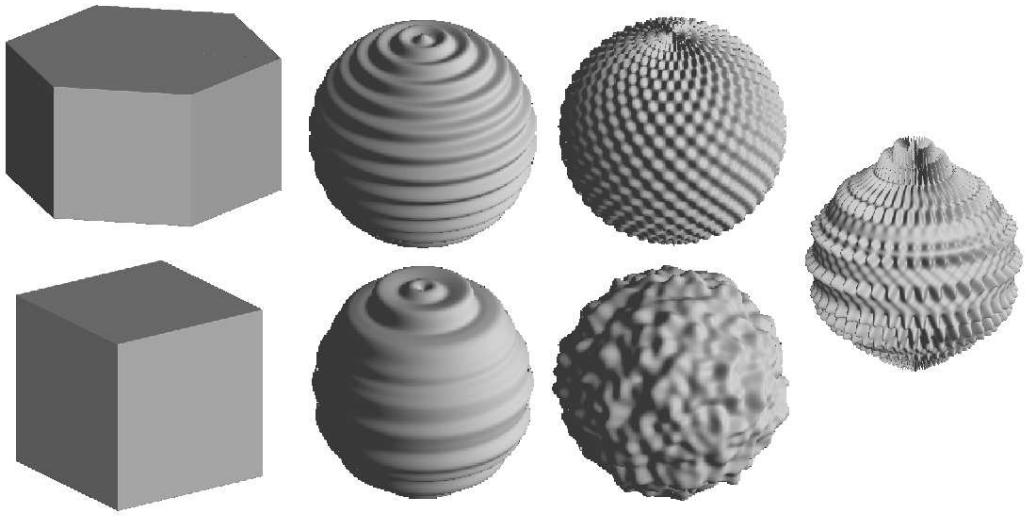

Figure 1: Examples of geometries implemented in *Tsym*.

the one shown in Fig. 1 (upper left) is invariant under a rotation by $2\pi/6$ about its main symmetry axis. The abstract rotation operation by $2\pi/6$ is denoted by $\hat{C}_6$. This operation can be represented in the vector space on which the T-matrix operates by a unitary matrix $\mathbf{U}(\hat{C}_6)$ with elements

$$U_{n,m,\tau,n',m',\tau'} = \delta_{n,n'}\delta_{m,m'}\delta_{\tau,\tau'} \exp\left(-2\pi \mathrm{i} m/6\right), \tag{12}$$

where $\delta_{l,l'}$ denotes the Kronecker delta, and where i is the imaginary unit. (More detailed information about representation of groups in T-matrix computations can be found in [18]). A symmetry operation $\mathbf{U}$ leaves the T-matrix invariant, i.e. $\mathbf{T} = \mathbf{U} \cdot \mathbf{T} \cdot \mathbf{U}^{-1}$, or

$$[\mathbf{T}, \mathbf{U}] = \mathbf{0}, \tag{13}$$

where $[\mathbf{T}, \mathbf{U}]$=$\mathbf{T}\cdot\mathbf{U}$-$\mathbf{U}\cdot\mathbf{T}$ denotes the commutator of $\mathbf{T}$ and $\mathbf{U}$. For the example in Eq. (12), the commutation relation of the T-matrix yields $T_{n,m,\tau,n',m',\tau'}$= $\exp[-2\pi\mathrm{i}(m-m')/6]T_{n,m,\tau,n',m',\tau'}$, or

$$T_{n,m,\tau,n',m',\tau'} = 0 \quad \text{unless} \quad |m - m'| = 0, 6, 12, \ldots. \tag{14}$$

Similar commutation relations hold for the Q- and RgQ-matrices. In this example, the number of Q-, RgQ-, and T-matrix elements that need to be

computed is reduced by a factor of 6. The remaining elements are zero owing to the particle's symmetries.

One obtains commutation relations as in Eq. (13) for each generator of the particle's symmetry group. In total, the commutation relations reduce the number of matrix elements that need to be computed by a factor of $1/M_o$ [18], where M_o denotes the order of the symmetry group. In addition, one can proof [16] by use of the commutation relations that the surface area of the particle, over which the integrals in Eq. (11) need to be evaluated, can be reduced by a factor of $1/M_o$. In total, the computation time is reduced by a factor of $1/M_o^2$. Depending on the order of the symmetry group, this typically amounts to a reduction in CPU-time requirements by 3–6 orders of magnitude!

Equation (8) requires the inverse of the Q-matrix. The *Tsym* code uses LU decomposition for inverting the Q-matrix, which has been shown to be the most robust method in T-matrix calculations [23]. However, numerical matrix inversion can introduce ill-conditioning problems in the calculations. The most elegant way to alleviate such problems is to exploit particle symmetries (if applicable) by changing from the basis of vector spherical wave functions to the so-called irreducible basis of the symmetry group. *Tsym* uses methods of computational group theory to automatically construct a similarity transformation $\mathbf{P}$ that represents the transformation into the irreducible basis [18]. In this basis, the Q- and T-matrices become block-diagonal. This is illustrated in Fig. 2 for a Q-matrix with truncation $n_{\max} = m_{\max}$=6, in which case the Q-matrix has 96 columns and rows. The upper panel schematically shows the symmetry structure of a Q-matrix for a particle belonging to the symmetry group $\mathcal{D}_{4h}$. (For instance, a brick with a regular square cross section belongs to this symmetry group.) Rather than showing the Q-matrix elements as numerical entries, this schematic picture of the matrix represents each independent, non-zero matrix element by a black square, while white squares represent elements that are zero due to symmetry, and grey squares represent matrix elements that are non-zero but related by symmetry to other non-zero matrix elements. It can be seen that the commutation relations strongly reduce the number of non-zero, independent matrix elements. After application of the transformation $\mathbf{P}$, the Q-matrix becomes block-diagonal (bottom), where the number of block matrices is always equal to the number of irreducible representations of the symmetry group (which, in case of the group $\mathcal{D}_{4h}$, happens to be 10). Rather than numerically inverting one large Q-matrix, the problem is now reduced to separately inverting each

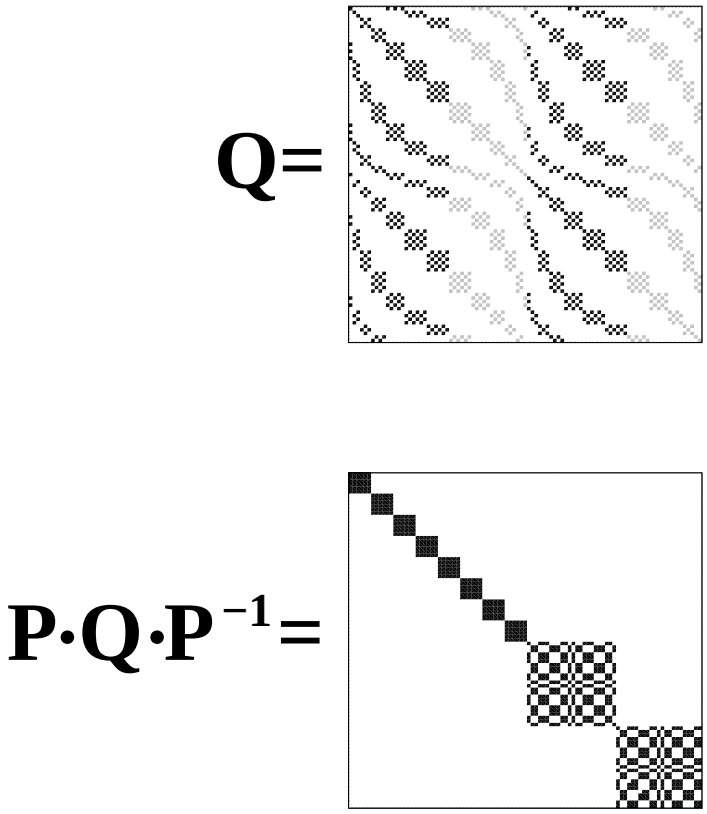

Figure 2: Schematic representation of the structure of the Q-matrix of a particle belonging to the symmetry group $\mathcal{D}_{4h}$ (top). After transformation into the irreducible basis, the Q-matrix becomes block-diagonal (bottom).

of the smaller block matrices. This can significantly reduce ill-conditioning problems in T-matrix computations.

A more detailed explanation of this method can be found in [18, 19]. The *Tsym* code is currently the only T-matrix code with the capability of exploiting irreducible representations of symmetry groups.

2.2. Lippmann-Schwinger T-matrix equation for particles with small-scale surface perturbations

Another way for reducing or, in some cases, even eliminating ill-conditioning problems has been proposed for particles with small-scale surface roughness [17, 21]. The idea is to compute the matrix $\mathbf{Q}$ of a particle with a small surface perturbation, and the matrix $\mathbf{Q}_0$ of the corresponding unperturbed geometry, and to formally introduce the difference $\Delta\mathbf{Q}=\mathbf{Q}-\mathbf{Q}_0$. Substitution into Eq. (8) yields after some manipulations a Lippmann-Schwinger T-matrix equation, from which we obtain an iterative T-matrix scheme

$$\mathbf{T}^{(0)} = -Rg\mathbf{Q}\cdot\mathbf{Q}_0^{-1} \tag{15}$$

$$\mathbf{T}^{(p)} = -(Rg\mathbf{Q}+\mathbf{T}^{(p-1)}\cdot\Delta\mathbf{Q})\cdot\mathbf{Q}_0^{-1}, \quad p\geq 1 \tag{16}$$

(see [17] for details). Here p denotes the iteration order. Inversion of $\mathbf{Q}_0$ is usually much more well-conditioned than that of $\mathbf{Q}$. If the unperturbed geometry is a homogeneous sphere, then $\mathbf{Q}_0$ is a diagonal matrix, and its inversion becomes trivial. In that case the ill-conditioning problems are entirely eliminated. However, the iterative scheme is limited to particles with small surface perturbations.

3. Description of the T-matrix code *Tsym*

3.1. Main purpose of the code

Tsym has been developed as a research code for developing and testing new theoretical and numerical methods (e.g. [17, 18]), for developing models for particles with small-scale surface roughness (e.g. [17, 19]), and for comparing optical properties of simple and morphologically more complex model particles (e.g. [24, 25]). Therefore, the emphasis in the development of the code has mainly been on flexibility rather than user-friendliness. Nevertheless, experienced user have, in the past, successfully used the code in rather ambitious applications that involved numerical computations for large

ensembles of particles, such as comparisons of model computations and laboratory measurements of mineral dust particles [26], and computation of ice cloud optical properties for operational remote sensing retrieval algorithms [27].

3.2. Applicability and general capabilities of the code

The T-matrix routines in *Tsym* are sufficiently general to compute the T-matrix for any homogeneous, star-shaped particle geometry. However, the routines are most efficient for particles with finite geometric symmetries. By making use of group theoretical methods, the implementation both expedites and stabilises the numerical computations for particles with symmetries. The code is optimised for non-axisymmetric particles with discrete symmetries. Although axisymmetric geometries and spheres can also be computed with *Tsym*, there exist other T-matrix implementations (e.g. [7, 8]) that are specifically tailored to such geometries; such codes are likely to be more efficient for computing optical properties of axisymmetric particles.

The *Tsym* code can compute optical properties of particles in both fixed and random orientations. More specifically, for particles in fixed orientation the code computes the polarised differential scattering cross section $(\mathrm{d}\sigma/\mathrm{d}\Omega)_{\alpha\beta}$, where $(\alpha\beta) = hh$, hv, vh, or vv, and where h and v represent polarisation in or perpendicular to the scattering plane. The first subscript α denotes the polarisation state of the incident field, while the second subscript β represents the polarisation state of the scattered field. The code further computes the Mueller matrix M_{ij}, ij =11, 22, 33, 44, 12, 34 (where, for particles in fixed orientation, the element M_{11} is not normalised), the extinction cross section C_{ext}, and the total scattering cross section C_{sca}.

For particles in random orientations, the code uses analytical orientation-averaging [13, 14, 6] to compute the Stokes scattering matrix F_{ij}, as well as C_{ext} and C_{sca}. The first element F_{11} of the Stokes scattering matrix is the phase function, which is properly normalised, i.e.

$$\frac{1}{2}\int_0^\pi F_{11}(\Theta)\sin\Theta\mathrm{d}\Theta = 1. \tag{17}$$

(where Θ is the scattering angle). Various derived parameters are also computed, such as the single scattering albedo $\omega = C_{\mathrm{sca}}/C_{\mathrm{ext}}$, the absorption cross section $C_{\mathrm{abs}} = C_{\mathrm{ext}} - C_{\mathrm{sca}}$, the asymmetry parameter

$$g = \frac{1}{2}\int_0^\pi F_{11}(\Theta)\cos\Theta\sin\Theta\mathrm{d}\Theta, \tag{18}$$

the backscattering cross section

$$C_{\mathrm{bak}} = \frac{1}{4\pi} C_{\mathrm{sca}} F_{11}(\pi), \tag{19}$$

the linear backscattering depolarisation ratio

$$\delta_L = \frac{F_{11}(\pi) - F_{22}(\pi)}{F_{11}(\pi) + F_{22}(\pi)}, \tag{20}$$

and the circular backscattering depolarisation ratio

$$\delta_C = \frac{F_{11}(\pi) + F_{44}(\pi)}{F_{11}(\pi) - F_{44}(\pi)}. \tag{21}$$

3.3. Main shortcomings of the code

The main drawback of the code is its limited user-friendliness. Most importantly, the code does not have any automated method for checking the convergence of the computations and for determining the precision-controlling parameters $n_{\max}$, $m_{\max}$, N_θ, and N_ϕ. It is entirely the user's responsibility to determine these parameters and to ensure the convergence and correctness of the computational results. Therefore, the code is mainly meant to be applied by expert users. Beginners with little prior experience with numerical methods in electromagnetic scattering usually find it difficult to use the code. Inexperienced users may fare better with a code that offers less flexibility but a higher degree of user-friendliness, such as T-matrix codes for axisymmetric particles.

Apart from these technical drawbacks, the T-matrix method is limited in the range of particle sizes, shapes, and dielectric properties for which convergent results can be obtained. The computations can become numerically unstable for particles with large size parameters $x = 2\pi r/\lambda$ (where r is the radius of a volume-equivalent sphere, and λ is the wavelength of light). Also, numerical instabilities are often encountered for dielectric particles with large real or imaginary parts of the refractive index $m = n + \mathrm{i}\kappa$, or for geometries that deviate significantly from spherical shape, such as highly prolate or highly oblate morphologies. Similar problems can be encountered for particles with surface perturbations with high deformation amplitudes. The exact parameter ranges for which convergent results can be obtained are interdependent. For instance, for strongly elongated polygonal prisms the range of size parameters is usually more limited than for mildly elongated

prisms. Similarly, for particles with high real or imaginary parts of the refractive index, convergent results are often obtained for a limited range of size parameters only. It is absolutely vital that the user carefully checks the convergence and accuracy of the computational results. Section 4.1 provides more information on how such convergence tests should be performed.

3.4. Structure and compilation

The code is written in Fortran 90, using double precision. Although the code is not parallelised, it can be an advantage to run it on a single node of a computing cluster rather than on a desktop, because this usually provides more memory. The user needs to edit the Makefile and specify the Fortran compiler. Running the *make* command will produce an executable called *tsym.x*.

The distribution of the program contains the *Tsym* source code, as well as several other routines that have been borrowed from other programs. The main directory contains the following program files.

- `Tsym.F`: This file contains the *Tsym* source code, including the main program, the null-field routines for computing the T- and Q-matrices, group theoretical routines for computing reducible representations and for constructing the transformation into the irreducible basis, as well as subroutines for computing the surface parameterisations of different particle geometries.

- `functions.F`: This file contains subroutines for computing spherical Bessel, spherical Hankel, and Wigner d-functions, which have been borrowed from the code by Michael Mishchenko [7].

- `gsphere.F`: This is a slightly edited version of the program by Karri Muinonen and Timo Nousiainen for generating 2D and 3D Gaussian random spheres [28].

- `lapack.F`: This contains various LAPACK and BLAS routines for LU decomposition and singular value decomposition [29].

- `OrientAvg.F`: This file contains routines from the program by Dan Mackowski [6] for computing vector-coupling coefficients and the orientation-averaged Stokes scattering matrix.

- `propack.F`: This is the PROPACK package for singular value decomposition of sparse, large matrices [30, 31].

- `smatrix.F`: This file contains routines from the program by Heikki Laitinen and Kari Lumme [32] for computing the amplitude scattering matrix and Mueller matrix for particles in fixed orientation.

- `NumRec.F`: In the publicly available distribution of *Tsym*, this file only contains two empty subroutines. Users with a license for Numerical Recipes can replace these by the subroutines `zroots` and `laguer` given in [33]. These routines are needed to use the improved quadrature method for polygonal prisms. More detailed instructions on how to enable this functionality in *Tsym* is given in Sect. 5.4. Users who do not have a license for Numerical Recipes can only run the code with the ordinary double Gauss-Legendre quadrature method.

3.5. Input file params

The input file is named `params`; it has to be placed in the same directory from which the program is called. The file contains comments that briefly explain the different parameters. The file has a fixed format, i.e. the number of comment lines must not be changed. A description of the parameters follows.

The header of the file contains a non-commented line that specifies the version of *Tsym*. Different program versions can require different formats of the input parameter file. The version number is read in by the program, and a quick test of the file `params` is performed to ensure that the file conforms with the format requirements of the particular *Tsym* version that is being run.

3.5.1. Parameters determining the particle geometry, symmetries, size, and dielectric properties

- `Geom`: A character string variable that specifies the geometry of the particle. The following choices are currently possible (see Figure 1 for some examples):

  - `'PRISMS'`: Prisms with regular polygonal cross sections. Figure 1 shows two examples, a prism with a hexagonal cross section (upper left), and a cube (lower left), which is a special case of a prism with a rectangular cross section. The number of corners

of the polygon is specified by the parameter `Nsym` (see below). A general expression for the surface parameterisation $r(\theta, \phi)$ for such geometries can be found in [16]. Note that in *Tsym* the standard orientation of the prisms is such that the z-axis is along the main rotational symmetry axis of the prism, while the x-axis intercepts one of the vertical edges of the prism. The centre of mass of the prism is in the origin of the coordinate system. Thus the cross section of the xy plane and the prism is a regular polygon, and the positive x-axis goes through one of the corners of the polygon. This even applies for a cube; so the positive z axis goes through the centre of one of the faces of the cube, while the positive x and y axes each intercept one of the edges (rather than one of the faces) of the cube. Different orientations can be computed as described farther below.

– `CHEB2D`: Axisymmetric Chebyshev particles (see Fig. 1, top centre). The rotational symmetry axis is along the z axis. The surface parameterisation is given by

$$r(\theta, \phi) = r_0(1 + \epsilon \cos \ell\theta), \tag{22}$$

where r_0 is the size of the unperturbed sphere, θ and ϕ denote the polar and azimuth angles, respectively, ϵ is the deformation parameter, and ℓ is the order of the Chebyshev polynomial.

– `'CHEB3D'`: Non-axisymmetric Chebyshev particles, defined by the surface parameterisation

$$r(\theta, \phi) = r_0(1 + \epsilon \cos \ell\theta \cos \ell\phi). \tag{23}$$

(see Fig. 1, top right).

– `'SPHERE'`: Homogeneous spheres. Note that it is more efficient to compute the optical properties of such particles with a standard Mie program. They have been included in *Tsym* mainly for being used in conjunction with the iterative T-matrix method defined in Eq. (16) — see also [17].

– `'GRSPHR'`: Gaussian random spheres (Fig. 1, bottom right). The construction of these geometries is discussed in [28]. T-matrix computations for these particles are rather limited. As they have

no symmetries, CPU time requirements are rapidly increasing with size parameters. However, for Gaussian random spheres with low correlation angles, other numerically exact and semi-exact methods for solving the electromagnetic scattering program seem to be equally, if not even more limited — see [22] for some examples.

  - `'GRSP2D'`: Axisymmetric Gaussian random spheres (Fig. 1, bottom centre). Owing to the high symmetry of these particles, T-matrix computations are much more efficient, and numerically somewhat more stable than for `GRSPHR`.
  - `'GRSCHB'`: A hybrid particle with a Gaussian random surface perturbation in the polar direction, and a regular Chebyshev perturbation in the azimuthal direction (Fig. 1, far right). This model particle incorporates the effects of random as well as 3D surface perturbations. By contrast to GRSPHR, it retains a high degree of symmetry, thus allowing for fast computations.

  The geometries `CHEB2D`, `CHEB3D`, `GRSPHR`, `GRSP2D`, and `GRSCHB` can be used as models for particles with small-scale surface roughness. Some merits and drawbacks of these models are discussed in [22].

- `cheborder`: The order ℓ of a Chebyshev particle as defined in Eqs. (22) and (23). This parameter setting is only effective for `Geom='CHEB2D'`, `'CHEB3D'`, or `'GRSCHB'`.

- `ngrs`: A cardinal number used for initialising the random number generator in those geometries based on the Gaussian random sphere, i.e. `GRSPHR`, `GRSP2D`, and `GRSCHB`. Different choices of this parameter will generate different pseudo-random geometries. For other geometries, this parameter is not used.

- `lreadgeom`: A logical switch for reading in a geometry from a previous run. This parameter is only effective for `GRSPHR`, `GRSP2D`, and `GRSCHB`. Gaussian random sphere computations, especially for the 3D random geometry class `GRSPHR`, can be very time consuming for low angular correlations. If computations are to be repeated for the same geometry and different sizes, then it can save a lot of time to set `lreadgeom=.true.`. However, this will only work if the same settings of the parameters `nth_int` and `nphi_int` are used throughout (see below).

- `igeom`: The particle size and shape of polygonal prisms are specified by the parameters `geom1` and `geom2`, (see below). But the program offers four different ways of specifying the dimensions of prisms. This is controlled by setting the parameter `igeom` to 1, 2, 3, or 4.

- `gpar1, gpar2, gpar3`: The meaning of these geometry parameters depends on the choice of `Geom` and, for `Geom='PRISMS'`, on the choice of `igeom`.

  - `Geom='PRISMS'`:

    * `igeom=1`:
      `gpar1`=l, the side length of the polygon (see Fig. 3);
      `gpar2`=h, the height of the polygon (see Fig. 3).
    * `igeom=2`:
      `gpar1`=a, the distance from the centre of the regular polygon to each of its corners (see Fig. 3);
      `gpar2`=$h/2$, the half-height of the prism.
    * `igeom=3`:
      `gpar1`=r_v, the radius of a volume-equivalent sphere;
      `gpar2`=$2a/h$, the "aspect ratio" of the prism.
    * `igeom=4`:
      `gpar1`=r_a, the radius of a surface-area equivalent sphere;
      `gpar2`=$2a/h$, the "aspect ratio" of the prism.

    `gpar3` is not used for `Geom='PRISMS'`.

  - `Geom='CHEB2D'` or `'CHEB3D'`:
    `gpar1`=r_0, the radius of the unperturbed sphere;
    `gpar2`=ϵ, the deformation parameter of the Chebyshev particle (where $-1 \leq \epsilon \leq 1$. `gpar3` is not used

  - `Geom='SPHERE'`:
    `gpar1`=r, the radius of the sphere. `gpar2` and `gpar3` are not used.

  - `Geom='GRSPHR'` or `'GRSP2D'`:
    `gpar1`=r_0, the radius of the unperturbed sphere;
    `gpar2`=σ, the relative standard deviation of the Gaussian random sphere (see [28]);
    `gpar3`=Γ, the correlation angle (in degrees) of the Gaussian random sphere (see [28]). Γ should be $\geq 4°$; for smaller angles, nu-

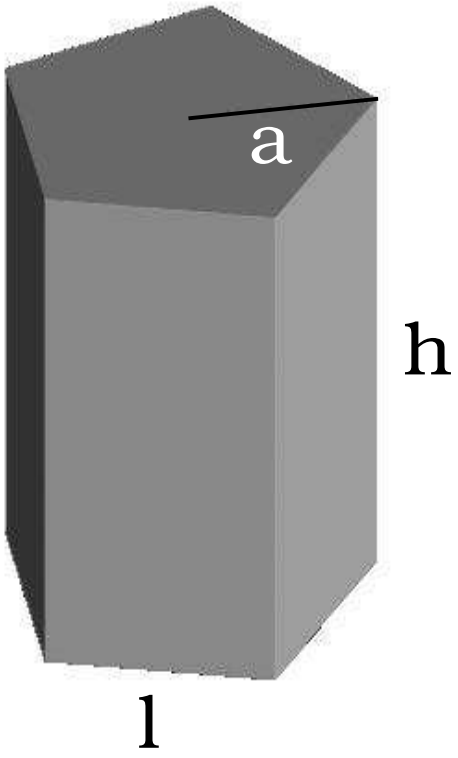

Figure 3: Height h, side length l, and "radius" a of a prism with a regular polygonal cross section.

merical instabilities in the Gaussian random sphere algorithm have been reported [22].

  - `Geom='GRSCHB'`:
    `gpar1`=r_0, the radius of the unperturbed sphere;
    `gpar2`=σ=ϵ, i.e. the relative standard deviation of the Gaussian random perturbation in the polar direction is equal to the deformation parameter of the Chebyshev perturbation in the azimuthal direction;
    `gpar3`=Γ, the correlation angle of the Gaussian random sphere.

- `Pgroup`: The point group of the particle. This parameter should be set — depending on the geometry — as follows.

  - `Geom='PRISMS'`: `Pgroup='Dnh'`
  - `Geom='CHEB2D'` or `'CHEB3D'`: For even values of the Chebyshev order `cheborder`, set `Pgroup='Dnh'`; For odd values of `cheborder`, set `Pgroup='Cnv'`;
  - `Geom='SPHERE'`: `Pgroup='Dnh'`. Note that this is not an optimal choice, since the dihedral symmetry group $\mathcal{D}_{nh}$ is only a subgroup of the full symmetry group of a sphere. However, *Tsym* has been designed for non-axisymmetric particles, so it does not fully exploit spherical symmetry. This is why it is more advantageous to do computations for spheres with dedicated Mie programs.
  - `Geom='GRSPHR'`: `Pgroup='Cn'`.
  - `'GRSP2D'`, or `Geom='GRSCHB'`: `Pgroup='Cnv'`.

  A summary is given in Table 1.

- `Nsym`: A cardinal number specifying the index of the main rotational symmetry axis of the particle. A rotation about the z-axis by an angle of $2\pi/N_{\rm sym}$ brings the particle into a new orientation that is indistinguishable from the original one. The meaning of `Nsym` for the different geometries is as follows.

  - `Geom='PRISMS'`: `Nsym` is the number of corners of the regular polygonal cross section of the prism.
  - `Geom='CHEB3D'` and `'GRSCHB'`: `Nsym` has to be set equal to `cheborder`, i.e. the order ℓ of the perturbing Chebyshev polynomial.

– `Geom='CHEB2D'`, `'SPHERE'`, or `'GRSP2D'`: These particles are axisymmetric. Thus, mathematically, `Nsym`$= \infty$. Technically, this can be achieved by setting `Nsym` equal to some large number. However, one does not want to set `Nsym` equal to some unnecessarily large number, since, otherwise, the group theoretical computations would take more time than necessary. It turns out to be sufficient to set `Nsym`=2 `mmax`+1. To understand this, we first generalise Eq. (14) to particles with $\hat{C}_N$-symmetry (N=`Nsym`). Only those T-matrix elements are non-zero for which $|m - m'| = N, 2N, 3N, \ldots$. In the limit $N \to \infty$, the T-matrix becomes block-diagonal in the m index, i.e., only those elements are non-zero for which $m - m' = 0$. Numerically, the T-matrix is truncated. From Eq. (10) one can see that $|m - m'|$ never becomes larger than 2 `mmax`. Thus, by setting `Nsym`=2 `mmax`+1, the *truncated* T-matrix has effectively the symmetry structure of an axisymmetric particle.

– `Geom='GRSPHR'`: This particle has no symmetries, so one has to set `Nsym`=1.

A summary is given in Table 1.

Note that for particles such as prisms and 3D Chebyshev particles, `Nsym` has to be ≥ 3. The special case of `Nsym`=2 would require some case distinctions in the group theoretical routines that have not yet been implemented.

- `lambda`: The wavelength of the incident electromagnetic field. The user can choose any units of length for specifying the wavelength (e.g. mm, μm, or nm), as long as this choice agrees with that for specifying the size of the particle.

- `mr`: Complex refractive index of the particle. The imaginary part of the refractive index has to be *positive*.

3.5.2. Parameters concerning the orientation of the particle

Tsym can compute optical properties of particles in one or several fixed orientations, in random orientations, or both. This is controlled by the following parameters.

Geom	Point group	Pgroup	Nsym
'PRISMS', N-gonal cross section	$\mathcal{D}_{Nh}$	Dnh	N
'CHEB2D', even order ℓ	$\mathcal{D}_{\infty h}$	Dnh	2mmax + 1
'CHEB2D', odd order ℓ	$\mathcal{C}_{\infty v}$	Cnv	2mmax + 1
'CHEB3D', even order ℓ	$\mathcal{D}_{\ell h}$	Dnh	ℓ
'CHEB3D', odd order ℓ	$\mathcal{C}_{\ell v}$	Cnv	ℓ
'GRSP2D'	$\mathcal{C}_{\infty v}$	Cnv	2mmax + 1
'GRSPHR'	$\mathcal{C}_1$	Cn	1
'GRSCHB', order ℓ	$\mathcal{C}_{\ell v}$	Cnv	ℓ

Table 1: Geometries implemented in *Tsym* and the corresponding settings for the parameters specifying the point group, Pgroup, and the index of the main rotational symmetry operation, Nsym

- norient: A cardinal number, for which the following choices are possible.
  - norient=1: Only compute orientation-averaged optical properties for randomly oriented particles.
  - norient=2: Only compute optical properties for particles in fixed orientations.
  - norient=3: Compute both orientation-averaged and fixed-orientation optical properties.

  Note that *Tsym* only performs analytic orientation-averaging. Numerical orientation-averaging is not supported.

- neuler: The number of fixed orientations for which computations are to be performed (only effective for norient=2 or 3).

- alpha, beta, and gamma: The three Euler angles (in degrees) describing the particle orientation. Each of these is an array with neuler elements, one for each of the discrete orientations of the particle for which the computations are to be performed. In other words, alpha(i), beta(i), gamma(i) are the three Euler angles for the ith orientation. These angles describe an *active* rotation of the particle in a *fixed* coordinate system, i.e. the particle is first rotated about the z axis by an angle gamma, then it is tilted by an angle beta about the y axis, then

it is rotated by an angle `alpha` about the fixed z axis. (Note that in *Tsym* the incident field is propagating in the positive z direction.)

3.5.3. Parameters determining the numerical precision and computational methods

- `nmax`, `mmax`: These parameters determine the truncation of the field expansions, thus of the T-matrix — see Eq. (10). The most critical step in successfully using the *Tsym* program is to determine these two parameters. More on that will be said in Sect. 4.

- `th_nint`, `phi_nint`: The number of Gauss-Legendre subintervals in the polar and azimuthal direction, respectively. The numerical evaluation of the surface integrals in Waterman's method uses `th_nint` integration intervals in the polar direction, and `phi_nint` intervals in the azimuthal direction, where each subinterval contains 16 Gauss-Legendre quadrature points. Thus the total numbers of quadrature points in Eq. (11) are given by $N_\theta = 16\times$`th_nint`, and $N_\phi = 16\times$`phi_nint`. It is important for the accuracy of the computations to carefully determine these parameters — see Sect. 4 for more information.

- `lprtb`: A logical flag. If true, then the T-matrix is computed by use of the iterative scheme given in Eq. (16). Otherwise the direct method in Eq. (8) is used. Note that the iterative method can only be used in conjunction with `Geom='CHEB2D'`, `'CHEB3D'`, `'GRSPHR'`, `'GRSP2D'`, and `'GRSCHB'`.

- `prtb_order`: The maximum iteration order p in Eq. (16). This will not be determined automatically. It is the user's responsibility to ensure that the results have converged with respect to the perturbation order. Note that the convergence with respect to `prtb_order` will become slower for larger perturbation amplitudes, i.e., for larger deformation parameters ϵ (for Chebyshev geometries) or relative standard deviations σ (for Gaussian random sphere geometries). If the perturbation amplitude becomes too large, the results may actually diverge. The larger the particle, the more limited the range of perturbation amplitudes for which convergent results can be obtained.

- `Cyl_Quad`: For `Geom='PRISMS'`, there exists an optimised method for evaluating the surface integrals that is superior to the double Gauss-Legendre quadrature method. This method has been introduced in [16].

However, this method makes use of an implementation of Laguerre's method from Numerical Recipes [33], which is not publicly available. The subroutine for Laguerre's method could therefore not be included in the distribution of *Tsym*, so this option had to be disabled. Users who have a licence for Numerical Recipes can easily include them in *Tsym* and enable the alternative quadrature method by just making minor adjustments. Section 5.4 explains how to do the necessary changes to the code.

- `lirredrep`: A logical flag. If true, the program will exploit irreducible representations of the particle's symmetry group to perform the inversion of the Q-matrix in Eq. (8). This can substantially reduce ill-conditioning problems. For low-order symmetry groups, it can also speed up the computations. This requires that the user specifies a character table file (see below). Note that this method cannot be combined with the iterative approach; so `lprtb` and `lirredrep` must not be true simultaneously.

  Also, note that the group theoretical methods only have been implemented for finite symmetry groups. For axisymmetric particles the only way of using irreducible representations is by setting `Pgroup` to the corresponding finite subgroup, and `Nsym`=2`mmax`+1 (see Table 1 and accompanying text). However, this is less efficient than using a T-matrix program that has been specifically hard-coded for axisymmetric geometries, such as those described in [7, 8]. So why, then, have axisymmetric geometries been included in *Tsym* in the first place? It is because other T-matrix programs do not currently use the iterative T-matrix method given in Eq. (16). The axisymmetric geometry options (`Geom='CHEB2D'` and `'GRSP2D'` have been included to be tested in conjunction with `lprtb=.true.` rather than `lirredrep=.true.`.

- `chartabfile`: If `lirredrep=.true.`, the user needs to provide the character table of the symmetry group. For many groups, the subdirectory `CHARACTER_TABLES` contains precomputed tables. The naming convention of the files in that directory is rather self explaining, as illustrated by the following examples (see also Table 1).

  - If `Geom='PRISMS'`, `Nsym=6`, then the corresponding symmetry group is the group $\mathcal{D}_{6h}$ (`Pgroup='Dnh'`). The file containing the character table of that symmetry group is named `D0006h.char`.

- If `Geom='CHEB3D'`, `cheborder=70`, one needs to set `Nsym=70` and `Pgroup='Dnh'`. The corresponding character table is named `D0070h.char`.
- If `Geom='CHEB3D'`, `cheborder=71`, one needs to set `Nsym=71` and `Pgroup='Cnv'`. The corresponding character table is named `C0071v.char`.
- If `Geom='GRSCHB'`, `cheborder=55`, one needs to set `Nsym=55` and `Pgroup='Cnv'`. The corresponding character table is named `C0055v.char`.
- If `Geom='GRSPHR'`, then the particle has no symmetries. In such case, one needs to set `Pgroup='Cn'`, `Nsym=1`, and `lirredrep=.false.`. No character table will be read in by the program.
- If `Geom='GRSP2D'`, the particle has axial symmetry. The idea of including axisymmetric geometries was to test them in conjunction with the iterative T-matrix method. However, it is also possible to use irreducible representations for these geometries. To this end, one needs to set `Pgroup='Cnv'` and `Nsym`=2`mmax`+1, and the table will be named `Cxv.char`, where `x=2 mmax+1`. However, depending on how large `mmax` is, the use of irreducible representations may become inefficient. In such case it may be best to set `lirredrep=.false.`, and no character table will be read in, although the program will still exploit symmetries for speeding up the computations. Rather than using irreducible representations, one may either try to use the iterative T-matrix method, or a different T-matrix code that has been optimised for axisymmetric particles.

If the required table is not found in the directory `CHARACTER_TABLES`, then the directory `GAP` contains some script programs for computing the table. More information on how to precompute character tables is given in Sect. 5.3.

Note that the program does not currently perform any check of the character table specified by `chartabfile` to ensure that it is consistent with the specified symmetry group. Therefore, if one should forget to specify the correct character table in the input file, the program will crash, but it will not issue an easily interpreted error message that points to the source of the error.

- `Pmatmethod`: If `lirredrep=.true.`, one needs to specify the method by which the transformation into the irreducible basis shall be computed.

There are currently three choices, which are explained in detail in [19].

- `Pmatmethod=1`: The transformation matrix is computed by a fairly simple and fast method that works well for low-order symmetry groups, such as the group $\mathcal{D}_{6h}$ of hexagonal prism. However, this method often fails for higher-order symmetry groups.
- `Pmatmethod=2`: This uses a general singular value decomposition (SVD) algorithm. This is the most robust method that works well for all groups. However, this is also the slowest method.
- `Pmatmethod=3`: This invokes an SVD algorithm that exploits the sparsity of the transformation matrix. This method is much faster than method 2, but it is also less stable. It seems to work best for symmetry groups of intermediate order.

Generally speaking, one should use method 1 for low symmetry orders. For higher orders one may want to try method 3. If it fails, one should use method 2. It does not cost much time to test out different methods. The transformation into the irreducible basis is computed before the time-consuming Q- and T-matrix computations are started. Thus, if the method for computing the transformation into the irreducible basis fails, then the program will crash relatively fast, and the program can be restarted with a different choice of `Pmatmethod`.

3.5.4. Output options

- `ntout`: Specifies the number of discrete angles at which the Stokes scattering matrix and the polarised differential scattering cross sections are written to the output files.
- `tmin`: The smallest scattering angle (in degrees) for which the differential scattering properties are written. It has to be $\geq 0^\circ$. Usually, one will set `tmin`=0.
- `tmax`: The largest output angle (in degrees). If only orientation-averaged optical properties are computed, then one usually sets `tmax`=180. Otherwise, if optical properties of particles in fixed orientations are computed, one sets `tmax`=360.

3.6. Output files

Runtime information and error messages are written to a file called `logfile`. The computational results are written to the following output files.

3.6.1. Particles in fixed orientation

- `C000001, C000002, ...`: These files contain integrated optical properties. There is one output file for each set of Euler angles. The output files are numbered in the order in which the Euler angles are specified in the input parameter file `params`.

  More specifically, the files contain the following results.

  - C_{ext}, extinction cross section;
  - C_{sca}, total scattering cross section;
  - C_{abs}, absorption cross section;
  - $Q_{\mathrm{ext}} = C_{\mathrm{ext}}/(\pi r^2)$, extinction efficiency;
  - $Q_{\mathrm{sca}} = C_{\mathrm{sca}}/(\pi r^2)$, total scattering efficiency;
  - $Q_{\mathrm{abs}} = C_{\mathrm{abs}}/(\pi r^2)$, absorption efficiency.

  The dimension of the cross sections is L^2, and the units of length agree with those used in the specification of the size and wavelength in the file `params`. The efficiencies are dimensionless. The particle size r used in the definition of the efficiencies is the particle radius for spheres, the volume-equivalent radius for prisms, and the radius of the unperturbed sphere for all Chebyshev and Gaussian random sphere geometries.

- `D000001, D000002, ...`: These files contain the elements of the polarised differential scattering cross section $(\mathrm{d}\sigma/\mathrm{d}\Omega)_{\alpha\beta}$. There is one file for each discrete orientation of the particle. More specifically, the columns of the files contain the elements

$$\Theta,\ k^2\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{hh},\ k^2\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{hv},\ k^2\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{vh},\ k^2\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{vv}, \tag{24}$$

  where Θ is the scattering angle. Note that the scaling with the squared wavenumber k^2 makes the differential scattering cross sections dimensionless. The rows of the files display results for equidistantly-spaced scattering angles, where the number and range of scattering angles has been specified in the file `params`.

- `F000001, F000002, ...`: These files contain the Mueller matrix $\mathbf{M}$ for particles in fixed orientation. There is one such file for each particle orientation. The columns contain the following data.

$$\Theta,\ M_{11},\ M_{22}/M_{11},\ M_{33}/M_{11},\ M_{44}/M_{11},\ -M_{12}/M_{11},\ M_{34}/M_{11}. \tag{25}$$

The element M_{11} is not normalised. The rows contain entries for different scattering angles.

3.6.2. Particles in random orientation

- `C.dat`: This file contains the following orientation-averaged, integrated optical properties.
  - C_{ext}, extinction cross section;
  - C_{sca}, total scattering cross section;
  - C_{abs}, absorption cross section;
  - C_{bak}, backscattering cross section;
  - $Q_{\text{ext}} = C_{\text{ext}}/(\pi r^2)$, extinction efficiency;
  - $Q_{\text{sca}} = C_{\text{sca}}/(\pi r^2)$, total scattering efficiency;
  - $Q_{\text{abs}} = C_{\text{abs}}/(\pi r^2)$, absorption efficiency.
  - $Q_{\text{bak}} = C_{\text{bak}}/(\pi r^2)$, backscattering efficiency;
  - $\omega = C_{\text{sca}}/C_{\text{ext}}$, single scattering albedo;
  - g, asymmetry parameter;
  - δ_L, linear backscattering depolarisation ratio
  - δ_C, circular backscattering depolarisation ratio

- `F.dat`: The orientation-averaged Stokes scattering matrix $\mathbf{F}$ with the following columns.

$$\Theta,\ F_{11},\ F_{22}/F_{11},\ F_{33}/F_{11},\ F_{44}/F_{11},\ -F_{12}/F_{11},\ F_{34}/F_{11}. \tag{26}$$

  The phase function F_{11} is normalised according to Eq. (17). The rows of the file contain entries for different scattering angles.

- `E.dat`: The expansion coefficients of the elements of the orientation-averaged Stokes scattering matrix in the basis of generalised spherical functions. The expansion coefficients, rather than the matrix elements, are often required as input to polarised radiative transfer codes, such as VDISORT [34, 35]. The columns contain the following entries.

$$l,\ a_1(l),\ a_2(l),\ a_3(l),\ a_4(l),\ b_1(l),\ b_2(l), \tag{27}$$

  where l is the expansion order, and the elements a_1, a_2, a_3, a_4, b_1, and b_2 are the expansion coefficients of the Stokes scattering matrix elements F_{11}, F_{22}, F_{33}, F_{44}, F_{12}, F_{34}, respectively. The rows contain entries for increasing expansion orders l.

4. Testing the accuracy of the computations

This section contains important information on how to test the correctness of the computational results. Accuracy tests need to be performed scrupulously. Failure to do so will almost certainly lead to wrong results.

4.1. Convergence tests

A necessary criterion for the correctness of the computations is that the results have converged with respect to the parameters `nmax`, `mmax`, `th_nint`, and `phi_nint`.

4.1.1. Computations for a single particle size

If computations are to be performed for just a single particle size, then the following procedure can be recommended.

- For any given `nmax`, set `mmax`=`nmax`. Since `mmax` is always $\leq$`nmax`, this is the safest choice in terms of computational accuracy, but also the most time-consuming. Any attempts to reduce `mmax` will only be advantageous when performing computations for many particle sizes (see below).

- Start by performing computations for some relatively low but fixed value of `nmax` that will not take much computation time, say, somewhere in the range between 10–20, depending on particle size. For that fixed value of `nmax`, perform computations for different values of the parameters `th_nint`, and `phi_nint`. The parameters should be increased until the computational results have converged. Then one should fix `th_nint` and `phi_nint` at those values where convergence has just been reached. (Choosing larger values will, of course, also give correct results, but it will require more computation time than necessary).

  Note that the relative number of polar and azimuthal quadrature intervals depend on symmetry and aspect ratio. As a rough guideline, if the particle has an aspect ratio near unity, one can set, as a first guess, `th_nint`$\approx$`Nsym`$\times$`phi_nint`. Strongly prolate prisms may require a larger number of polar angles, strongly oblate prisms will need a larger number of azimuthal angles. Note also that `phi_nint` and `th_nint` must be ≥ 2.

- Once the values of `th_nint` and `phi_nint` have been fixed, run computations for increasing values of `nmax`=`mmax`, and investigate the convergence of the optical properties. The smaller the size parameter, the smaller values of `nmax` will be required for obtaining convergent results. So, for size parameters much smaller than unity, one should choose a starting value of `nmax`=1. For particles with size parameters on the order of or much larger than unity, correspondingly larger starting values of `nmax` can be chosen. To save time in the convergence tests, one can increase `nmax` in steps of 3.

  An example is given in Fig. 4, which is based on computations for a hexagonal ice prism with dimensions l=200 μm, h=400 μm, and for a sub-millimetre wavelength of λ=500 μm. The refractive index of ice at that wavelength is m=1.79+0.014i. The top left panel shows $C_{\rm sca}$ (solid line) and $C_{\rm abs}$ (dashed line) as a function of `nmax`. There is an initial increase of the cross sections with `nmax` followed by a broad range up to about `nmax`=38 in which the results have converged. Finally, at higher values of `nmax` the results start diverging, and they even become non-physical ($C_{\rm abs} < 0$).

  Within the range of `nmax` values for which the cross sections have converged, one can investigate the elements of the Mueller matrix (for particles in fixed orientation) and/or of the Stokes scattering matrix (for particles in random orientations). The recommended way to do this is to compare computational results obtained for `nmax` and `nmax`+3. One wants to find the smallest value of `nmax` for which this comparison indicates that the results have converged. Fig. 4 shows such a comparison for the example above. The phase function F_{11} (bottom left) and the degree of linear polarisation $-F_{12}/F_{11}$ (bottom right) are shown for `nmax`=10 and `nmax`=13, each computed for particles in random orientation. The results are virtually indistinguishable. For smaller values of `nmax`, the results had not yet converged (not shown). Note that the minimum value of `nmax` for which the elements of the Stokes scattering matrix have converged is larger than the minimum value of `nmax` for which the cross sections have converged.

  More examples on the convergence behaviour of cubes of different size parameters and refractive indices can be found in [36].

  Figure 5 shows a similar investigation for a 2D-Gaussian random sphere

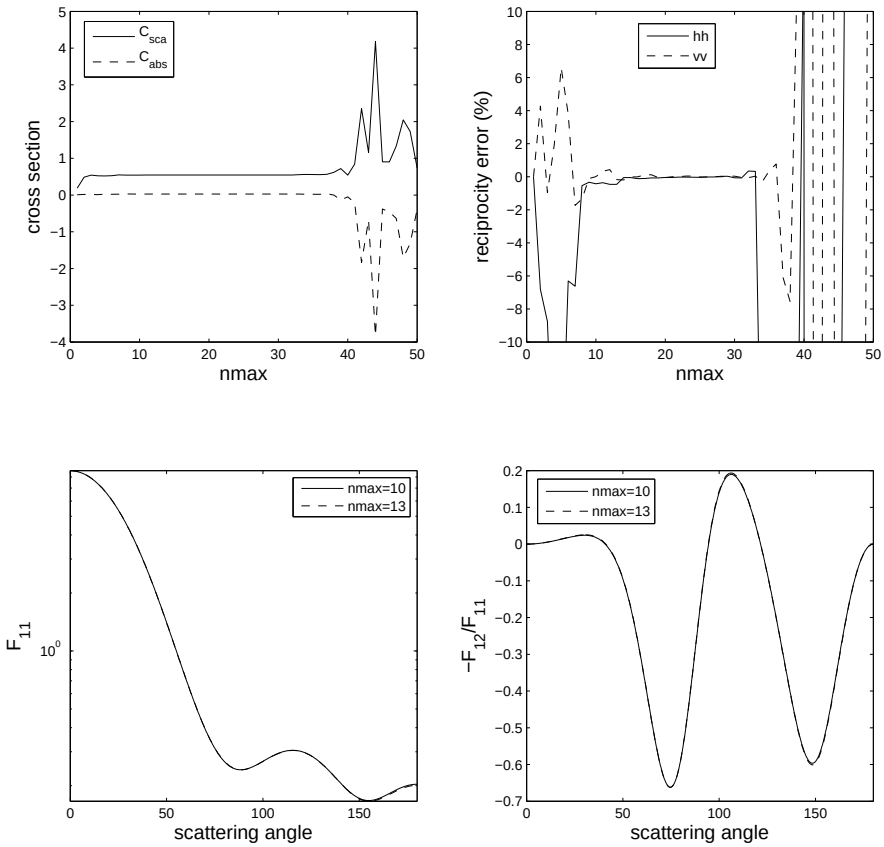

Figure 4: $C_{\rm sca}$ and $C_{\rm abs}$ (top left) and percent reciprocity errors of the hh and vv components of the polarised differential scattering cross section (top right) as a function of `nmax`. Both plots are based on computations for a hexagonal prism (as explained in the text) in a fixed orientation. The bottom row shows a comparison of the phase function (left) and degree of linear polarisation (right) for corresponding hexagonal prisms in random orientation, computed with `nmax`=10 and `nmax`=13.

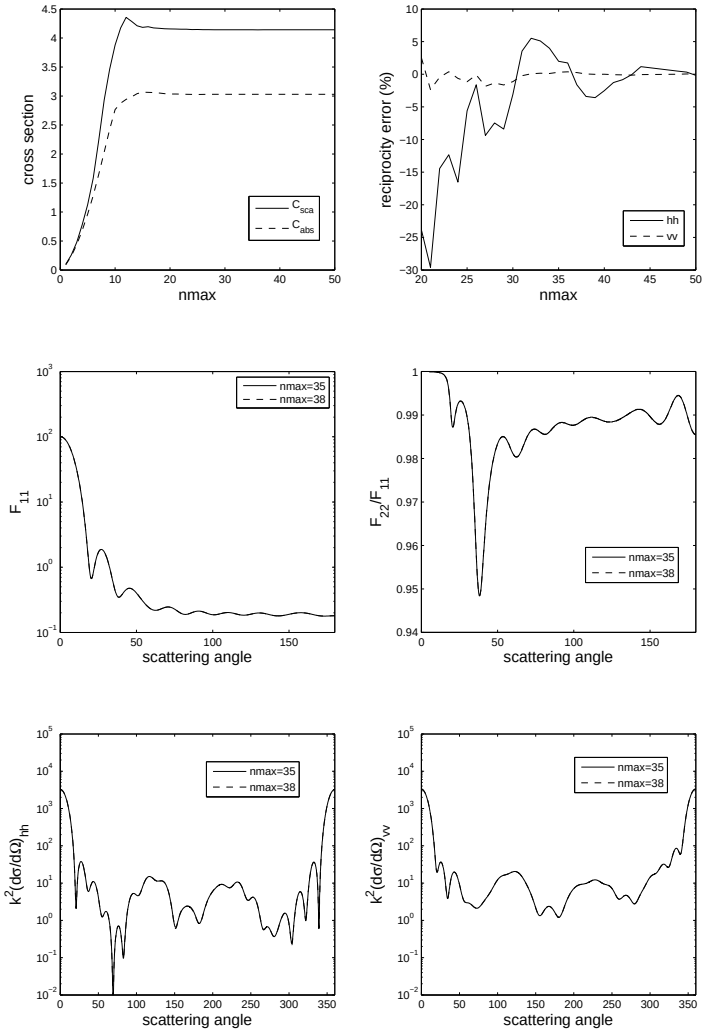

Figure 5: $C_{\rm sca}$ and $C_{\rm abs}$ (top left) and percent reciprocity errors of the hh and vv components of the polarised differential scattering cross section (top right) as a function of `nmax`. Both plots are based on computations for a 2D-Gaussian random sphere (as explained in the text) in a fixed orientation. The bottom row displays the hh and vv components of the polarised differential scattering cross section computed with `nmax`=35 and `nmax`=38. The middle row shows a comparison of the phase function (left) and degree of linear polarisation (right) for corresponding 2D-Gaussian random spheres in random orientation, computed with `nmax`=35 and `nmax`=38.

(`Geom='GRSP2D'`), where the radius of the unperturbed sphere is r=1.0 μm, the relative standard deviation is σ=0.05, the correlation angle is $\Gamma = 18^\circ$, the wavelength is λ=0.628 μm, and the refractive index is m=3+0.1i (typical for hematite at visible wavelengths). The top panels are analogous to those in Fig. 4. The middle panels show the phase function (left) and the phase matrix element F_{22}/F_{11} (right) for randomly oriented particles, comparing computations for `nmax`=35 and 38. A corresponding comparison is shown in the bottom row for the hh (left) and vv components of the polarised differential scattering cross sections for particles in a fixed orientation. One main difference to Fig. 4 is the much slower increase of the cross sections with increasing `nmax`. Higher values of `nmax` are needed for obtaining stable results. Within the range of `nmax` values considered here, no divergence at the higher end of the range is observed.

- For larger size parameters, the final result obtained for `nmax` can be significantly larger than the initial value of `nmax` used for determining the parameters `th_nint` and `phi_nint`. Thus one needs to check once more with the new value of `nmax` if the computations are still stable with respect to increasing `th_nint` and `phi_nint`. If not, then `th_nint` and `phi_nint` need to be further increased until convergence is reached. If new values of `th_nint` and `phi_nint` are found, one needs to use these values to double-check once more convergence with respect to `nmax`, etc. In practice, this procedure does not require more than one iteration. Convergence with respect to `th_nint` and `phi_nint` is usually much less critical than convergence with respect to `nmax`.

More detailed discussions of convergence tests and error measures of T-matrix results can be found in [37, 38, 36]. For instance, a systematic procedure for analysing curves such as those shown in the upper left panels of Figs. 4 and 5, and for obtaining error estimates of T-matrix results is presented in Ref. [36].

In practice, one needs to expedite the convergence tests without compromising the reliability of the results. One important way to achieve this is to find a good starting value for `nmax`. Even with much experience, it is not always easy to guess a reasonable starting value, especially when doing calculations for different refractive indices. In some cases, it can be helpful to obtain a first guess by running computations for axisymmetric particles with a T-matrix program with automatic convergence procedures. For instance,

for those geometries that are based on imposing a surface perturbation on a sphere, i.e. `Geom='CHEB2D'`, `'CHEB3D'`, `'GRSCHB'`, `'GRSPHR'`, or `'GRSP2D'`, one can run a code such as *mieschka* [8] for axisymmetric Chebyshev particles of comparable size and deformation parameter, and use the truncation of the T-matrix determined with the automatic convergence routines of that code as a first guess for *Tsym*. Note that 3D-Chebyshev particles and, even more so, particles with stochastic surface perturbations can require higher values of `nmax` than 2D-Chebyshev particles. So, the short-cut for obtaining a first guess for `nmax` described here is not a substitute for carefully testing the convergence of the *Tsym* calculations.

4.1.2. Computations for many particle sizes

Consider the case of an ensemble of particles with the same geometry but with different sizes within a size range $r_i \leq r \leq r_f$. Computing optical properties averaged over a size distribution usually involves calculations for many discrete sizes. It would be extremely impractical and time consuming to repeat the procedure described in Sect. 4.1.1 for each and every particle size in the ensemble. However, it is, in most cases, safe to assume that a set of parameters `nmax`, `mmax`, `th_nint`, `phi_nint` that gives convergent results for one particle size r_0 will also give convergent results for particles of the same geometry with size $r < r_0$ (as long as r does not become much smaller than r_0, in which case the value of `nmax` applicable to r_0 may lie in the divergent regime for $r << r_0$). If the size distribution covers a large range with many discrete particle sizes, it would be inefficient to determine the convergence-controlling parameters for just the largest size r_f in the size distribution, and use these values for all other sizes. Rather, the following procedure should be followed.

- Choose a number of "milestone sizes" $r_f = r_n > r_{n-1} \ldots > r_1$ that stake out the whole size range, where $r_1 > r_i$.
- For each of these n sizes determine the convergence-controlling parameters `nmax`, `th_nint`, `phi_nint` as described in Sect. 4.1.1, where one initially sets `mmax=nmax`.
- Try to reduce `mmax` and test how low you can go before the results deviate from the results obtained for `mmax=nmax`. Trying to reduce `mmax` will help to save computation time in the following calculations.

- At this stage you have determined for each size r_j $(j = 1, \ldots, n)$ the parameters `nmax`(j), `mmax`(j), `th_nint`(j), `phi_nint`(j). Now use these values in the computations for all sizes r with $r_{j-1} < r \leq r_j$.

The user needs to decide how many size intervals are to be used. The decision will be a compromise between the amount of manual work and the amount of computation time one can afford to invest. If only a few size intervals are selected, then the intervals $r_{j-1} < r < r_j$ will be rather large. Thus many computations for smaller sizes near the lower end of the interval will be performed with unnecessary high values of the truncation parameters `nmax`, `mmax`, `th_nint`, and `phi_nint`, which can result in high CPU time requirements. On the other hand, increasing the number of intervals will require more manual work to determine the truncation parameters `nmax`(j), `mmax`(j), `th_nint`(j), `phi_nint`(j) for a large number of size-interval boundaries r_j.

4.2. Reciprocity tests

The reciprocity condition expresses the invariance of electromagnetic scattering under the exchange of the source and detector point, In other words, the polarised differential scattering cross section is invariant under reversal of the optical path. This is illustrated in Fig. 6 (left). We denote the wave vector and polarisation state of the incident field by $\mathbf{k}_{\rm inc}$ and α, respectively, and those of the scattered field by $\mathbf{k}_{\rm sca}$ and β. Formally, the reciprocity condition states that

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\alpha\beta}(\mathbf{k}_{\rm inc}, \mathbf{k}_{\rm sca}) = \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\beta\alpha}(-\mathbf{k}_{\rm sca}, -\mathbf{k}_{\rm inc}). \tag{28}$$

In *Tsym* the incident field is always assumed to be propagating in the positive z direction. Hence, reciprocity tests need to be performed by rotating the particle, rather than changing the direction of the incident field. This is illustrated in the example in Fig. 6. In the original case we consider a particle in its original orientation (which we denote by $\theta_p = 0°$), and we are interested in the scattered field at a scattering angle of $\theta_s = 90°$. The reciprocal case is obtained by reversing the optical path (left). Alternatively, one can rotate the particle by an angle of $\theta_p = 90°$ about an axis perpendicular to and pointing into the plane of the figure, and investigate the scattered field at an angle of $\theta_s = 270°$ (right). Comparison of the left and right panels in Fig. 6 shows that the reciprocal light paths in both cases are identical. More

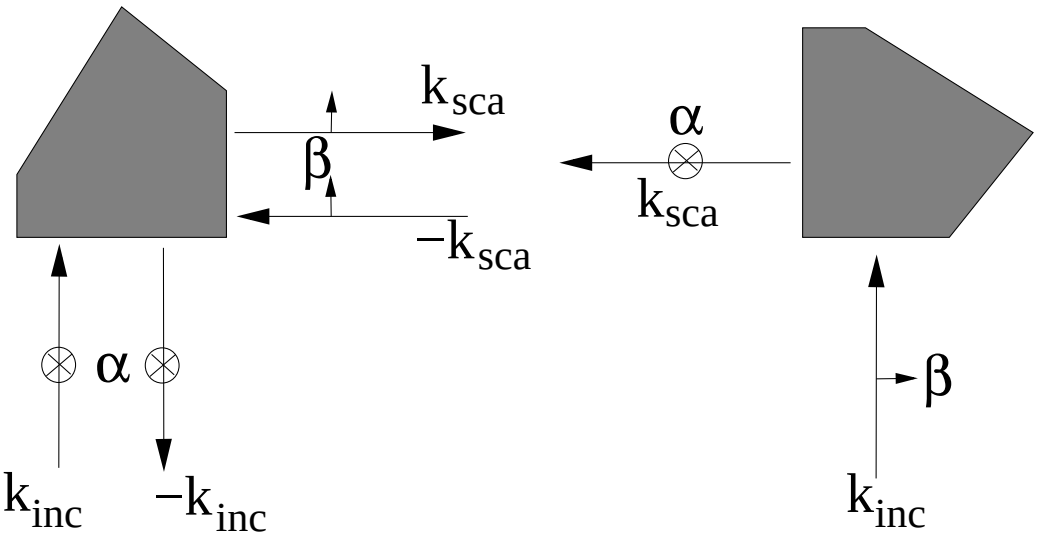

Figure 6: The reciprocity condition is based on reversing the optical path, i.e. exchanging source and detector point (left). In *Tsym*, the direction of the incident field is fixed, and the reciprocal case is obtained by rotating the particle (right).

formally, the reciprocity condition in this specific example becomes

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\alpha\beta}(\theta_p = 0^\circ, \theta_s = 90^\circ) = \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\beta\alpha}(\theta_p = 90^\circ, \theta_s = 270^\circ). \qquad (29)$$

This example can be implemented in the parameter file `params` by the following settings.

```
     3  !  norient
     2  !  neuler
   0.0   0.0  !  alpha
  0.0   90.0  !  beta
   0.0   0.0  !  gamma
```

The parameter `norient` has to be either 2 (only fixed orientations) or 3 (both fixed and random orientations). The program will perform computations for two fixed particle orientations characterised by the Euler angles $(0^\circ, 0^\circ, 0^\circ)$ and $(0^\circ, 90^\circ, 0^\circ)$, respectively. In the second orientation, the particle is rotated by an angle of 90° about the y axis. The polarised differential

scattering cross sections for these two particle orientations will be written to the output files `D000001` and `D000002`, respectively. The results in the first file at a scattering angle of 90° need to be compared to those in the second file at 270°. One must not forget to perform this comparison for the correct polarisation states: The component hh (column 2) in the first and second file are compared with each other, and likewise the component vv (column 5). However, recall that the component hv (column 3) in the first file needs to be compared to the component vh (column 4) in the second file — see Eq. (29). Similarly, the vh component in the first file is compared to the hv component in the second file. Note that in many cases the cross-polarisation components hv and vh are much smaller than the other two components; in such case, the reciprocity tests should concentrate on the hh and vv components.

Clearly, one can set up more reciprocity tests by choosing other scattering angles and corresponding particle orientations. Obviously, one must not select a case in which the rotation of the particle happens to be a symmetry operation of the particle (i.e., in the reciprocal case the particle orientation must not be indistinguishable from the original particle orientation), otherwise the reciprocity condition is trivially fulfilled due to symmetry. Also, it is usually a good idea to test the reciprocity condition at a scattering angle where the polarised differential scattering cross section does not have any narrow peaks or troughs. At such angles one can obtain unrepresentatively high reciprocity errors even if the overall accuracy of the results is fairly high.

The reciprocity condition tends to be a highly sensitive test of the accuracy of the computations. It is particularly useful if one needs to compute differential scattering properties with high accuracy. As a rough guideline, reciprocity errors of less than 5 % usually indicate highly accurate results; errors less than 3 % can be considered excellent. It is not unusual that one observes convergence of the results with respect to an increase of the truncation parameters, such as `nmax`, but violation of the reciprocity condition. This can indicate that one should choose larger values of the truncation parameters than suggested by the convergence tests. This can be clearly seen in the example given in Fig. 4. The cross sections (upper left) seem to have converged at values of `nmax` as low as 3. However, the reciprocity errors (upper right) are still highly oscillating at such low values of `nmax`. Consistently low reciprocity errors are not reached before increasing `nmax` to 8–10 and higher. A close inspection confirms that values of `nmax`$\geq$10 are needed for obtaining convergent results of the Mueller matrix elements. Similarly, the cross sections appear to be computationally stable for values of `nmax` up

to 38, while the reciprocity test indicates numerical instabilities at values of `nmax`>33. This clearly illustrates the high sensitivity of the reciprocity condition to numerical inaccuracies that can easily be overlooked when inspecting integrated optical properties only. Similar observations can be made by inspecting the reciprocity error in Fig. 5 (top right panel). The total cross sections (top left) seem to have converged at `nmax` as low as 13. However, much higher values of `nmax` (around 35) are needed for obtaining stable results for the Mueller matrix elements (middle panels). This is confirmed by the reciprocity error of the hh component. It is quite large for `nmax`=13, and it starts falling below the 5 % mark at around `nmax`=33.

These examples illustrate the potential usefulness of the reciprocity condition as an additional test of the computational accuracy. The reciprocity condition is particularly useful for testing the correctness of computations for which no other independent electromagnetic scattering method exists. For instance, it will be difficult to compare computations with *Tsym* for high-order 3D-Chebyshev particles of large size parameters to computations performed with other codes. In the absence of such independent information, reciprocity tests can help to stake out the range of applicability of the program. A major disadvantage of the reciprocity condition is that it requires extra computation time. Usually, one does not need to invest this extra computation time if one is only interested in total optical cross sections. A more detailed account of reciprocity tests in light scattering computations can be found in [39].

5. Utilities

5.1. Examples

The subdirectory `EXAMPLES` contains a few test cases with corresponding input and output files. Each directory contains a `README` file that describes the test case. In some cases, the `README` file also contains suggestions for exercises related to the test case that the novice practitioner can do to become familiar with the code. One test case contains shell-scripts and Fortran programs for performing a detailed analysis of the convergence of the code for a particular test case. The user can modify the scripts and input file of this test case to set up convergence analyses for other applications. Instructions on how to use and modify the scripts are given in the `README` file.

5.2. Matlab plotting script

When running *Tsym*, the program automatically generates three files `matlabx.out`, `matlaby.out`, and `matlabz.out`. These files contain the geometry of the particle in a format that can be read in by Matlab for plotting the particle. The subdirectory `MATLAB/` contains a simple Matlab script `plot_geometry.m` for this purpose. The particle is always displayed in its standard orientation, i.e. for Euler angles `alpha`=0, `beta`=0, and `gamma`=0.

5.3. GAP

As explained in Sect. 3.5.3, the group theoretical methods in *Tsym* require a file that contains the so-called irreducible characters of the symmetry group. The name of that file is assigned in the input file `params` to the string variable `chartabfile`. Several precomputed character tables are found in the subdirectory `CHARACTER_TABLES`. If the particle of interest belongs to a symmetry group of which the character table is not contained in that directory, then the user needs to compute the table prior to running *Tsym*.

The calculation of character tables requires methods of computational group theory. Such methods are implemented in the Groups, Algorithms, and Programming (*GAP*) system for computational discrete algebra. The source code can be obtained at the *GAP* website [40]. However, using *GAP* for computing character tables, especially for high-order symmetry groups, is far from trivial. (Some of the technical difficulties are discussed in [19]). For this reason, the newer versions of *Tsym* contain a subdirectory `GAP` with a shell script and two Fortran programs that essentially automatise the running of *GAP* and the post-processing of the output.

To compute character tables with the help of the toolbox provided by *Tsym* in the subdirectory `GAP`, the user should proceed as follows.

- Obtain and install *GAP*. The source code with installation instructions can be downloaded from the *GAP* website [40].
- Compile the two Fortran programs `generate_gap_input.f` and `reorder_character_table.f`, and name the executables `generate_gap_input.x` and `reorder_character_table.x`, respectively.
- Edit the shell script 'gap.bash'.
  - Edit the line
    `gap=/.../gap.sh`

and specify the complete path of your *GAP* installation, under which the script `gap.sh` can be found.

– Specify the names of the point groups for which you want to compute character tables, as well as the indices of the main rotational symmetry axes. Example: Suppose you want to compute character tables for the groups $\mathcal{C}_{100}$, $\mathcal{C}_{150}$, $\mathcal{C}_{200}$, $\mathcal{C}_{100v}$, $\mathcal{C}_{150v}$, $\mathcal{C}_{200v}$, $\mathcal{D}_{100h}$, $\mathcal{D}_{150h}$, and $\mathcal{D}_{200h}$. Then you make the following settings in the bash script.

```
groups="Cn Cnv Dnh"
Nsyms="100 150 200"
```

- Run the script `gap.bash`.
- Move the output files `*.char` to the directory `../CHARACTER_TABLES/`.

The script `gap.bash` loops over the point groups specified by the user and performs the following tasks.

- First it calls the program `generate_gap_input.x`, which generates the files `gap.in` and `gap2.in` in the *GAP* programming language.
- The first script `gap.in` is run by calling *GAP*. This generates a character table in *GAP* format, which is written to the temporary file `gap.out`.
- *GAP* is called again to execute the second script `gap2.in`. This generates a file `ConjClass.out`. That file contains the vital information of the ordering of the conjugacy classes of the character table in `gap.out`.
- Finally, the script calls the program `reorder_character_table.x`. This program analyses the conjugacy classes and reorders the character table according to the *Tsym* standard ordering of conjugacy classes. Now the table is ready for use in *Tsym*.

5.4. Numerical Recipes

For `Geom='PRISMS'`, it is recommended to perform the surface integrations in Waterman's T-matrix method with a quadrature method that has been tailored to this class of particles. It is invoked by setting `Cyl_Quad` equal to `.true.`. The method is described in [16]. It uses a double Gauss-Legendre

quadrature scheme on the side facets, but only a single quadrature in the polar direction on the top and bottom facets. The azimuthal integration on the top and bottom facets is performed analytically. To determine the range of the azimuthal integration for each polar quadrature angle, one needs to find the roots of a fourth order polynomial — see Eq. (37) in [16]. *Tsym* uses the Fortran routines `zroots` and `laguer` from Numerical Recipes [33] to determine the roots by use of Laguerre's method. However, these routines are not part of the publicly available distribution of *Tsym*, because they cannot be used without a license. Information on how to obtain a license for Numerical Recipes can be found in [33] on p. iv.

Licensed users can easily insert the Numerical Recipes Fortran routines `zroots` and `laguer` in the file `NumRec.F`. Further, one needs to edit the subroutine `setup` in the file `Tsym.F` and comment out the marked lines just after the comment heading "Integration scheme". These lines issue a warning if the user attempts to set `Cyl_Quad=.true.`, and they enforce the setting `Cyl_Quad=.false.`. After commenting out those lines and implementing the Numerical Recipes subroutines for Laguerre's method, the user can use the quadrature scheme for prisms by setting `Cyl_Quad` equal to `.true.` in the input file `params`.

6. Summary

The *Tsym* code described here is meant to be applied to non-axisymmetric particles with finite symmetries. *Tsym* exploits particle symmetries by use of group theory, which can profoundly improve the speed and stability of T-matrix computations. On the down side, it requires both skill and time to carefully check the convergence of the computations, which severely limits the user-friendliness of the code. Thus, the code should be used by program developers, experienced users, and those who are able and willing to spend time for thoroughly familiarising themselves with the theoretical basics and practical aspects of the program. The source code can be obtained via the *Tsym* homepage [20].

The public release of a research code always carries a certain risk of improper use of the program, resulting in the production and publication of incorrect result. This is particularly true for a code such as *Tsym* that places a high responsibility on the user for ensuring the correctness of the results. On the other hand, scientific progress critically depends on transparency, openness and, most importantly, independent analyses of scientific methods

and results. The public release of *Tsym* allows other users to carefully examine the strengths and limitation of the theoretical and numerical methods implemented in this code. The long-term benefits of an open-access policy are therefore expected to outweigh the potential risks.

[1] P. C. Waterman, Matrix formulation of electromagnetic scattering, Proc. IEEE 53 (1965) 805–812.

[2] B. R. Johnson, Invariant imbedding T matrix approach to electromagnetic scattering, Appl. Opt. 27 (1988) 4861–4873.

[3] D. W. Mackowski, Discrete dipole moment method for calculation of the T matrix for nonspherical particles, J. Opt. Soc. Am. A 19 (2002) 881–893.

[4] F. M. Schulz, K. Stamnes, J. J. Stamnes, Scattering of electromagnetic waves by spheroidal particles: A novel approach exploiting the T-matrix computed in spheroidal coordinates, Appl. Opt. 37 (1998) 7875–7896.

[5] T. A. Nieminen, H. Rubinsztein-Dunlop, N. R. Heckenberg, Calculation of the T-matrix: general considerations and application of the point-matching method, J. Quant. Spectrosc. Radiat. Transfer 79-80 (2003) 1019–1029.

[6] 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 (1996) 2266–2278.

[7] M. I. Mishchenko, L. D. Travis, Capabilities and limitations of a current FORTRAN implementation of the T-matrix method for randomly oriented, rotationally symmetric scatterers, J. Quant. Spectrosc. Radiat. Transfer 60 (1998) 309–324.

[8] T. Rother, Electromagnetic wave scattering on nonspherical particles, Springer, Berlin, 2009.

[9] A. Doicu, T. Wriedt, Y. A. Eremin, Light scattering by systems of particles. Null-field method with discrete sources: theory and programs, Springer, Berlin, 2006.

[10] ScattPort.
URL http://www.scattport.org

[11] T. Wriedt, J. Hellmers, New scattering information portal for the light-scattering community, J. Quant. Spectrosc. Radiat. Transfer 109 (2008) 1536–1542.

[12] V. V. Varadan, V. K. Varadan, Multiple scattering of electromagnetic waves by randomly distributed and oriented dielectric scatterers, Phys. Rev. D 21 (1980) 388–394.

[13] M. I. Mishchenko, Light scattering by randomly oriented axially symmetric particles, J. Opt. Soc. Am. A. 8 (1991) 871–882.

[14] N. G. Khlebtsov, Orientational averaging of light-scattering observables in the T-matrix approach, Appl. Opt. 31 (1992) 5359–5365.

[15] F. M. Schulz, K. Stamnes, J. J. Stamnes, Point group symmetries in electromagnetic scattering, J. Opt. Soc. Am. A 16 (1999) 853–865.

[16] F. M. Kahnert, J. J. Stamnes, K. Stamnes, Application of the extended boundary condition method to homogeneous particles with point group symmetries, Appl. Opt. 40 (2001) 3110–3123.

[17] M. Kahnert, T. Rother, Modeling optical properties of particles with small-scale surface roughness: combination of group theory with a perturbation approach, Opt. Express 19 (2011) 11138–11151.

[18] M. Kahnert, Irreducible representations of finite groups in the T matrix formulation of the electromagnetic scattering problem, J. Opt. Soc. Am. A 22 (2005) 1187–1199.

[19] M. Kahnert, T-matrix computations for particles with high-order finite symmetries, J. Quant. Spectrosc. Radiat. Transfer this issue.

[20] Tsym homepage - T matrix code for scattering by homogeneous particles with discrete symmetries (2012).
URL http://www.rss.chalmers.se/~kahnert/Tsym.html

[21] T. Rother, J. Wauer, Case study about the accuracy behavior of three different T-matrix methods, Appl. Opt. 49 (2010) 5746–5756.

[22] M. Kahnert, T. Nousiainen, M. A. Thomas, J. Tyynelä, Light scattering by particles with small-scale surface roughness: comparison of four classes of model geometries, J. Quant. Spectrosc. Radiat. Transfer (2012) http://dx.doi.org/10.1016/j.jqsrt.2012.03.017.

[23] D. J. Wielaard, M. I. Mishchenko, A. Macke, B. E. Carlson, Improved T-matrix computations for large, nonabsorbing and weakly absorbing nonspherical particles and comparison with geometrical-optics approximation, Appl. Opt. 36 (1997) 4305–4313.

[24] F. M. Kahnert, Reproducing the optical properties of fine desert dust aerosols using ensembles of simple model particles, J. Quant. Spectrosc. Radiat. Transfer 85 (2004) 231–249.

[25] M. Kahnert, A. Kylling, Radiance and flux simulations for mineral dust aerosols: Assessing the error due to using spherical or spheroidal model particles, J. Geophys. Res. 109 (2004) D09203, doi:10.1029/2003JD004318, errata: doi:10.1029/2004JD005311.

[26] T. Nousiainen, M. Kahnert, B. Veihelmann, Light scattering modeling of small feldspar aerosol particles using polyhedral prisms and spheroids, J. Quant. Spectrosc. Radiat. Transfer 101 (2006) 471–487.

[27] M. Matricardi, The inclusion of aerosols and clouds in RTIASI, the ECMWF fast radiative transfer model for the infrared atmospheric sounding interferometer, Tech. Rep. 474, ECMWF Research Dept. Tech. Memo., Reading, UK (2005).

[28] K. Muinonen, T. Nousiainen, P. Fast, K. Lumme, J. I. Peltoniemi, Light scattering by gaussian random particles: ray optics approximation, J. Quant. Spectrosc. Radiat. Transfer 55 (1996) 577–601.

[29] LAPACK –Linear Algebra PACKage.
URL `http://www.netlib.org/lapack`

[30] R. M. Larsen, Lanczos bidiagonalization with partial reorthogonalization, Tech. Rep. DAIMI PB-357, Department of Computer Science, Aarhus University (1998).

[31] R. M. Larsen, PROPACK homepage (2012).
URL `http://soi.stanford.edu/ rmunk/PROPACK/`

[32] H. Laitinen, K. Lumme, T-matrix method for general star-shaped particles: first results, J. Quant. Spectrosc. Radiat. Transfer 60 (1998) 325–334.

[33] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in FORTRAN, Cambridge University Press, Cambridge, 1992.

[34] F. M. Schulz, K. Stamnes, F. Weng, VDISORT: An improved and generalized discrete ordinate radiative transfer model for polarized (vector) radiative transfer computations, J. Quant. Spectrosc. Radiat. Transfer 61 (1999) 105–122.

[35] F. M. Schulz, K. Stamnes, Angular distribution of the stokes vector in a plane parallel, vertically inhomogeneous medium in the vector discrete ordinate radiative transfer (vdisort) model, J. Quant. Spectrosc. Radiat. Transfer 65 (2000) 609–620.

[36] M. A. Yurkin, M. Kahnert, Light scattering by a cube: accuracy limits of the discrete dipole approximation and the t-matrix method, J. Quant. Spectrosc. Radiat. Transfer this issue.

[37] T. Rother, Green functions for plane wave scattering on single nonspherical particles, in: A. Kokhanovsky (Ed.), Light Scattering Reviews 4, Springer, Berlin, 2009, pp. 121–161.

[38] J. Hellmers, V. Schmidt, T. Wriedt, Improving the numerical stability of t-matrix light scattering calculations for extreme particle shapes using the nullfield method with discrete sources, J. Quant. Spectrosc. Radiat. Transfer 112 (2012) 1679–1686.

[39] K. Schmidt, M. Yurkin, M. Kahnert, A case study on the reciprocity in light scattering computations, Opt. Express 20 (2012) 23253–23274.

[40] The GAP Group, GAP – Groups, Algorithms, and Programming, Version 4.5.5 (2012).
URL `http://www.gap-system.org`

Acknowledgements

The author is indebted to Heikki Laitinen, Kari Lumme, Dan Mackowski, Michael Mishchenko, Karri Muinonen, and Timo Nousiainen for their kind permission to incorporate various Fortran routines from their respective programs into *Tsym* (as listed in Sect. 3.4). Michael Wolff and Sergei Volkov provided many helpful suggestions for improving this paper. Funding from the Swedish Research Council (project 621-2011-3346) is gratefully acknowledged.