METRICA
METRology Institutional CAtalog

ISTITUTO NAZIONALE DI RICERCA METROLOGICA
Repository Istituzionale

T-matrix representation of optical scattering response: Suggestion for a data format

Original
T-matrix representation of optical scattering response: Suggestion for a data format / Asadova, N., Achouri, K., Arjas, K., Auguié, B., Aydin, R., Baron, A., Beutel, D., Bodermann, B., Boussaoud, K., Burger, S., Choi, M., Czajkowski, K.M., Evlyukhin, A.B., Fazel-Najafabadi, A., Fernandez-Corbaton, I., Garg, P., Globosits, D., Hohenester, U., Kim, H., Kim, S., et al.. - In: JOURNAL OF QUANTITATIVE SPECTROSCOPY & RADIATIVE TRANSFER. - ISSN 0022-4073. - (2024). [10.1016/j.jqsrt.2024.109310]
Availability:
This version is available at: 11696/82479 since: 2024-12-25T21:41:37Z

Publisher:
Elsevier

Published
DOI:10.1016/j.jqsrt.2024.109310

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

08 November 2026

Journal of Quantitative Spectroscopy & Radiative Transfer 333 (2025) 109310

Contents lists available at ScienceDirect

Journal of Quantitative Spectroscopy and Radiative Transfer

journal homepage: www.elsevier.com/locate/jqsrt

T-matrix representation of optical scattering response: Suggestion for a data format

Nigar Asadova [a], Karim Achouri [b], Kristian Arjas [c], Baptiste Auguié [d], Roland Aydin [e], Alexandre Baron [f], Dominik Beutel [a], Bernd Bodermann [g], Kaoutar Boussaoud [a], Sven Burger [h], Minseok Choi [i], Krzysztof M. Czajkowski [j], Andrey B. Evlyukhin [k], Atefeh Fazel-Najafabadi [d], Ivan Fernandez-Corbaton [a], Puneet Garg [a], David Globosits [l], Ulrich Hohenester [m], Hongyoon Kim [i], Seokwoo Kim [i], Philippe Lalanne [n], Eric C. Le Ru [d], Jörg Meyer [a], Jungho Mun [o], Lorenzo Pattelli [p], Lukas Pflug [q], Carsten Rockstuhl [a,*], Junsuk Rho [i], Stefan Rotter [l], Brian Stout [s], Päivi Törmä [c], Jorge Olmos Trigo [t], Frank Tristram [a], Nikolaos L. Tsitsas [r], Renaud Vallée [f], Kevin Vynck [u], Thomas Weiss [m], Peter Wiecha [v], Thomas Wriedt [w], Vassilios Yannopapas [x], Maxim A. Yurkin [y], Grigorios P. Zouros [x]

[a] Karlsruhe Institute of Technology, Kaiserstrasse 12, Karlsruhe, 76131, Germany
[b] École Polytechnique Fédérale de Lausanne, Route Cantonale, Lausanne, 1015, Switzerland
[c] Aalto University School of Science, PO Box 15100, Aalto, 00076, Finland
[d] 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
[e] Hamburg University of Technology, Eißendorfer Straße 42, Hamburg, 21073, Germany
[f] Univ. Bordeaux, CNRS, CRPP, UMR 5031, F-33600, Pessac, France
[g] Physikalisch-Technische Bundesanstalt, Bundesallee 100, Braunschweig, 38116, Germany
[h] Zuse Institute Berlin, Takustrasse 7, Berlin, 14195, Germany
[i] Pohang University of Science and Technology (POSTECH), Pohang, 37673, Republic of Korea
[j] Faculty of Physics, University of Warsaw, Pasteura 5, Warsaw, PL-02-093, Poland
[k] Leibniz University Hannover, Welfengarten 1, Hannover, 30167, Germany
[l] Vienna University of Technology (TU Wien), Wiedner Hauptstraße 8-10/136, Vienna, 1040, Austria
[m] Institute of Physics, University of Graz, Universitätsplatz 5, Graz, 8010, Austria
[n] Institut d'Optique d'Aquitaine, 1 Rue F. Mitterrand, Talence, 33400, France
[o] Purdue University, West Lafayette, IN 47904, USA
[p] Istituto Nazionale di Ricerca Metrologica, Str. delle Cacce 91, Turin, 10135, Italy
[q] Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 5a, Erlangen, 91058, Germany
[r] Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece
[s] Université d'Aix-Marseille, 52 Av. Escadrille Normandie Niemen, Marseille, 13013, France
[t] Universidad de La Laguna, Apdo. 456, San Cristóbal de La Laguna, 38200, Spain
[u] Institut Lumière Matière, CNRS, Université Claude Bernard Lyon 1, 10 Rue Ada Byron, 69622 Villeurbanne Cedex, France
[v] LAAS, Université de Toulouse, CNRS, Toulouse, France
[w] Universität Bremen, Badgasteiner Str.3, Bremen, 28359, Germany
[x] National Technical University of Athens, Iroon Polytechniou St. 9, Athens, 15780, Greece
[y] Université Rouen Normandie, INSA Rouen Normandie, CNRS, CORIA UMR 6614, Rouen, F-76000, France

ARTICLE INFO

Dataset link: https://github.com/tfp-photonics/tmatrix_data_format

Keywords:
Data format
Data Science
Data storage and management
Multipolar representations

ABSTRACT

The transition matrix, frequently abbreviated as T-matrix, contains the complete information in a linear approximation of how a spatially localized object scatters an incident field. The T-matrix is used to study the scattering response of an isolated object and describes the optical response of complex photonic materials made from ensembles of individual objects. T-matrices of certain common structures, potentially, have been repeatedly calculated all over the world again and again. This is not necessary and constitutes a major challenge for various reasons. First, the resources spent on their computation represent an unsustainable financial and ecological burden. Second, with the onset of machine learning, data is the gold of our era,

* Corresponding author.
E-mail address: carsten.rockstuhl@kit.edu (C. Rockstuhl).

https://doi.org/10.1016/j.jqsrt.2024.109310
Received 20 August 2024; Received in revised form 3 December 2024; Accepted 4 December 2024
Available online 19 December 2024
0022-4073/© 2024 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Scattering
T-matrix

and it should be freely available to everybody to address novel scientific challenges. Finally, the possibility of reproducing simulations could tremendously improve if the considered T-matrices could be shared. To address these challenges, we found it important to agree on a common data format for T-matrices and to enable their collection from different sources and distribution. This document aims to develop the specifications for storing T-matrices and associated metadata. The specifications should allow maximum freedom to accommodate as many use cases as possible without introducing any ambiguity in the stored data. The common format will assist in setting up a public database of T-matrices.

Part I

Introduction

1. Background information

The transition matrix, or T-matrix, constitutes a comprehensive representation of the optical properties of a scatterer in linear approximation [1,2]. In that context, the basic scattering problem can be expressed as follows: Given a scatterer in the surrounding medium and all its properties, i.e., shape and material composition, what is the scattered field for a given illumination? Due to the restriction to linear response, we can solve the problem in the frequency domain, i.e., we consider time-harmonic excitation. The response to a pulsed illumination can be reconstructed thanks to the superposition principle [3]. The advantage of the T-matrix approach resides in the fact that an algebraic expression, i.e., a matrix–vector-product, describes the light-matter interaction. For that, the incident and the scattered fields are expanded in a basis set, and their amplitudes are stored in a vector [4]. The T-matrix multiplied by the incident field vector gives the scattered field vector. The number of expansion coefficients is truncated in numerical calculations, which lends the T-matrix a finite dimension. For the expansion, vector spherical harmonics are usually used to reflect the three-dimensional localized character of the objects. The lowest-order expansion coefficients capture the dipolar, quadrupolar, and octupolar responses, and higher orders are of interest as well [5–7]. It remains to be mentioned that other matrices represent the optical response from a scattering object as well. A typical example would be the S-matrix that relates incoming and outgoing fields instead of incident and scattered fields. Moreover, the K-matrix, also called the reaction matrix, exists. It relates the total field outside the scatterer as a superposition of regular fields and singular fields [8]. The reaction matrix has the nice property that it is Hermitian for lossless systems. Still, the different matrices can be converted to each other. We concentrate, therefore, on one of them here, the T-matrix.

Besides being the basis for discussing the scattering response from a given object, the T-matrix allows the study of advanced photonic materials made from a larger number of objects on semi-analytical grounds [9,10]. Examples are coupled particles, ensembles of thousands, or even millions of identical or different scatterers that form amorphous photonic materials [11], or infinite arrangements of identical unit cells that contain one or multiple scatterers [12–14]. Scattering interaction with a substrate can also be considered [15]. Additionally, a computed T-matrix allows us to study the scattering by aggregates of particles or orientation-averaged scattering [16,17]. This is needed in optical particle characterization and in atmospheric radiative transfer. The spectral domain of interest and the application to be explored on the base of T-matrices are diverse. In electrodynamics, it spans multiple scientific disciplines, such as optics and photonics, nanotechnology, astronomy and astrophysics, remote sensing, atmospheric science, biophysics, and nanomedicine [18–21], and it can even be used for tasks such as measuring the size of air bubbles in water [22]. Beyond science, the description of scattering of light by particles also covers many important applications in metrology and technologies such as nanoelectronics and advanced material characterizations.

To perform all this research, the T-matrix of an object needs to be known. While it can be obtained experimentally [23], we discuss other approaches in the course of the paper. Unfortunately, analytical solutions are only available for basic shapes. For spheres, Gustav Mie finalized the solution of the problem in 1908 and gave us what we call nowadays the Lorentz-Mie coefficients [24]. They form the entries of a diagonal T-matrix. For all other particles, numerical methods are required to obtain their T-matrix. The null-field or extended boundary condition methods can compute T-matrices of gyroelectric spherical objects [25], as well as of dielectric and gyrotropic non-spherical objects [26–28]. Otherwise, any available Maxwell solver can be employed to compute the scattered field or the induced current in the scatterer for a set of different illumination conditions [29]. From the induced response, the T-matrix can be constructed. In the most direct approach, an individual vector spherical harmonic is used for the illumination, and the scattered response is expanded to obtain one column of the T-matrix. But also, plane wave illuminations are possible.

However, these computations consume quite some resources. Depending on the problem details, many full-wave simulations are necessary. To quantify the efforts, we may argue that the T-matrix might be of interest in dipolar or octupolar order, leading to six or 30 expansion coefficients for the field [30]. Generally, we need $2N(N+2)$ expansion coefficients for a T-matrix of order N. Given that the size of the T-matrix is $2N(N+2) \times 2N(N+2)$, we need $2N(N+2)$ simulations to retrieve the T-matrix for an object with no symmetry. Suppose we are interested in a dispersive T-matrix (e.g., for 200 wavelengths), one might wish to vary one or multiple geometrical parameters (e.g., a helix can be parameterized with at least four numerical parameters, and we might be interested in testing ten values for each parameter), and assume a computational time of our full wave solver of three minutes for one full wave problem. In that case, we may need something between three hours and three years to compute all these T-matrices. The upper limit is certainly an extreme example. Still, experience says that we need a few hours of computational time on a reasonable infrastructure to retrieve a T-matrix of interest with the necessary precision. Because a larger community is interested in the optical response from the same objects, this calculation and the possible re-calculation constitute a major challenge for different reasons.

2. Challenges

It is intellectually not satisfying to repeat the same tasks multiple times. Therefore, it seems wise to calculate T-matrices once, systematically store them, and make them available for later reuse according to the principles of findability, accessibility, interoperability, and reusability (FAIR principles) [31–33]. Beyond this general consideration, more practical reasons strongly suggest a common data format for T-matrices and their proper storage.

First, their computation consumes resources, both intellectual and scientific, as it requires dedicated scientists to handle the computation. Still, conventional resources are equally needed. Besides hardware, the energy used to perform the calculations should not be underestimated. With the onset of the current energy crisis, we, as a community, are asked to perform resource efficient computations. But even before, the increased energy expenses of computing facilities put the issue of reducing energy consumption on the agenda. To quantify the importance, data from the Scientific Computing Center at the Karlsruhe Institute of Technology suggests that the computational power per investment into hardware doubles every 1.5 years. In contrast, the computational

power per energy consumption doubles only every 2.7 years. Taking the ratio implies that the energy consumption per investment doubles every 3.3 years. That trend appears to be stable for the past ten years. As of today, the expenses for energy within five years correspond to the investment sum for a contemporary supercomputer. Following this trend of increasing importance of electricity costs suggests that in ten years, the expenses for energy will be eight times higher than for hardware. And in 20 years, the expenses for energy will be even 64 times higher. Consequently, financial support for energy, rather than hardware, will be the more crucial issue in the years to come.

Beyond that economic challenge, there is the associated environmental challenge. We shall strive to reduce our carbon footprint as much as possible to contribute to the solution of an essential problem for humanity, i.e., climate change. One answer could be to stop computations. But this does not make much sense when our purpose is to achieve progress and contribute to solving problems humanity faces. Photonic structures find use in devices for energy harvesting, photocatalysis, water purification, cancer treatment, and many more. We should not stop doing research. However, we should strive to perform research responsibly. Avoiding repetitive calculation of T-matrices on the base of which we study optical systems is one contribution that our community should aim at.

Moreover, performing science reproducibly is increasingly essential. Publishing computational codes open source is one step in this direction. Especially in the context of the T-matrix-based scattering formalism, we witnessed in the past years the publication of multiple codes for that purpose that all have their unique focus [5,34–41]. With the publication of these codes, we did not just give back to the public what had been supported by taxpayers' money. We also put others in the position to reproduce our results, develop them further, or contribute in new directions. This empowers an entire community and generates trust in published results.

However, a key ingredient in the multiple scattering formalism, the actual T-matrix, frequently cannot be generated within the framework of the semi-analytical scattering theory. Instead, it requires additional software that solves Maxwell's equations to generate T-matrices. This necessity of additional software might be a burden, and the cascading of research tools reduces the transparency of the numerical work. Relying on publicly available T-matrices would lower that dependency. In this context, the T-matrix is not only valuable as a subject of interest on its own, but serves as an interface between different computation tools.

Many computer codes have been developed over the years to compute T-matrices. Here, we focus on methods to compute scattering by non-axisymmetric scatterers. The T-matrix of a non-axisymmetric object (an ellipsoid) was first computed via the Null-field Method (NFM) (also referred to as the Extended Boundary Condition Method, EBCM) by Barber in his Ph.D. thesis [42]. Schneider and Peden used the NFM to compute scattering by an ellipsoid [43]. Later, Laitinen, and Lumme [44] used the method to compute scattering by a cube expanded into spherical functions. Wriedt [45] and Doicu et al. [34] used the NFM with discrete sources (NFM-DS) to compute scattering by arbitrarily shaped 3D particles described by a triangulated surface. Yurkin and Kahnert also computed scattering by cubes and compared results to that of the discrete dipole approximation (DDA) [46]. The T-matrix of a scattering object can also be computed via other surface and volume based computational electromagnetics methods. Nieminen et al. [3] used the point matching method. Das et al. [47] used the surface integral equation method (SIEM). Mackowski [48] as well as Loke et al. [49] used DDA. A similar method, the volume integral equation method (VIEM) is used by Markkanen and Yuffa [50]. Recently, there is much research in the invariant embedding method to compute the T-matrix of complex shaped particles. This method is used by Bi et al. [51] and Doicu et al. [52]. Generally, there is a larger number of general purpose solvers available to compute the T-matrix of a scatterer [53].

Finally, we want to emphasize the positive aspects of having an agreed T-matrix data format and a database where T-matrices are collected and archived. Hosting the template scripts in a dedicated repository and including source files for each datafile contributes to the comparability and interchangeability of the programs and promotes collaborations in the community. With that, it supports the validation and the identification of possible systematic errors of the different approaches.

New research questions emphasizing data-intensive methods can be tackled based on aggregated T-matrices. For example, training neural networks that solve direct or inverse problems in scattering theory would become more feasible [54,55]. This, in essence, could avoid one day even the necessity of an ordinary Maxwell solver to obtain the T-matrix of a given structure. Whether such networks can be trained for general purposes remains an open question, but this should be possible for important geometries frequently considered. Also, some basic science questions could be answered, like whether there is a particle for any possible T-matrix that copes with all physical constraints that can be imposed. Collecting these T-matrices stored in an agreed data format would be a stepping stone for entirely new directions our communities could pursue.

For all these reasons, the following document contains information on a data format we suggest using in the future. The document has been worked out by a larger number of groups working on the development of tools to study scattering problems. That ranges from atmospheric scattering problems, to nanophotonics, to metrology, to scattering in biological samples and beyond. It is the effort of a larger number of community members [56,57].

The document includes two parts after a brief introduction on the T-matrix formalism in the following. First, the data format specifications are described. Afterward, multiple aspects concerning the generation and validation of the T-matrices are discussed. Then, we outline some details related to the exploitation of the generated files in multiscattering problems and provide some summarizing statements in the end. Along with this documentation, codes that contain utilities to work with the data format are provided under the following link: https://github.com/tfp-photonics/tmatrix_data_format.

In perspective, a data repository and a dedicated web interface will be made available as a preferred portal for the archiving and exchanging of T-matrix datasets. However, this description would exceed the scope of the current manuscript, and we leave its documentation for a future article.

3. T-matrix formalism

Here, we support the concepts introduced above with mathematical formulations. The discussion is kept rather brief and only given with the purpose to set a common ground to discuss the data format. A recent general introduction into the method can be found in the book chapter by Mackowski [58]. The scattering problem is illustrated in Fig. 1. It is expressed with the following equation:

$$\mathbf{p} = \mathbf{T}\mathbf{a}\,, \tag{1}$$

where the vectors $\mathbf{p}$ and $\mathbf{a}$ represent the expansion coefficients of scattered and incident fields, respectively, and $\mathbf{T}$ is the T-matrix [59]. The expansion of the scattered field is not considered to be valid for nonspherical scatterers everywhere inside the smallest sphere circumscribing the scatterer, which is a topic of ongoing research [60–62]. The expansion can be performed with different basis functions, the most common one being the vector spherical wave functions (VSWF):

$$\mathbf{E}_{\mathrm{inc}}(\mathbf{r},\omega) = \sum_{l=1}^{\infty}\sum_{m=-l}^{l}\left[a^{\mathrm{e}}_{lm}(\omega)\mathbf{N}^{(1)}_{lm}(\mathbf{r},\omega) + a^{\mathrm{m}}_{lm}(\omega)\mathbf{M}^{(1)}_{lm}(\mathbf{r},\omega)\right] \tag{2}$$

$$\mathbf{E}_{\mathrm{sca}}(\mathbf{r},\omega) = \sum_{l=1}^{\infty}\sum_{m=-l}^{l}\left[p^{\mathrm{e}}_{lm}(\omega)\mathbf{N}^{(3)}_{lm}(\mathbf{r},\omega) + p^{\mathrm{m}}_{lm}(\omega)\mathbf{M}^{(3)}_{lm}(\mathbf{r},\omega)\right]. \tag{3}$$

In these expressions, VSWFs are defined in parity basis, $\mathbf{M}_{lm}(\mathbf{r},\omega)$ are the transverse electric (TE) or magnetic multipole fields, and

$\mathbf{N}_{lm}(\mathbf{r},\omega)$ are the transverse magnetic (TM) solutions or electric multipole fields [63]. Beware that different authors use different definitions of spherical vector wave functions. The superscripts (1) and (3) correspond to the choice of the spherical Bessel or Hankel functions in the solution, resulting in regular or singular solutions. The field inside the scatterer can also be expanded in VSWFs:

$$\mathbf{E}_{\text{int}}(\mathbf{r},\omega) = \sum_{l=1}^{\infty}\sum_{m=-l}^{l}\left[c^{\text{e}}_{lm}(\omega)\mathbf{N}^{(1)}_{lm}(\mathbf{r},\omega) + d^{\text{m}}_{lm}(\omega)\mathbf{M}^{(1)}_{lm}(\mathbf{r},\omega)\right]. \tag{4}$$

In the numerical calculation, the multipole order l is truncated to the largest order non-negligible for a given problem. A basis with well-defined helicity can be alternatively employed. The center of expansion can also be chosen in different ways [64]. Alternatively, a cylindrical wave expansion is beneficial for problems with a specific geometry and can be used to analytically solve the scattering problem of an infinite cylinder with arbitrary geometrical cross-section [4,65]. However, this basis set will not be addressed in the further discussion. The exact definitions and the normalization used in the data format specification are noted in Appendix C.

The VSWFs are complex-valued in the general case. Importantly, the time evolution is defined here by the factor $\exp(-i\omega t)$, which is more common in the optics community than its complex-conjugated counterpart. If required for some approaches, such as quasi-normal modes, complex frequencies can be considered [66,67]. Occasionally, the notion of "mode" here is used to refer to the vector spherical harmonics because they are simple solutions of the source-free Maxwell equations in homogeneous space in a spherical coordinate system, comparable to plane waves which are simple solutions to the Maxwell equations in homogeneous space in a Cartesian coordinate system. This is not to be confused with quasi-normal modes, which are the modes sustained by a scatterer.

An additional aspect worth mentioning is the connection of the T-matrix to the S-matrix, represented in a simple relation:

$$\mathbf{S} = \mathbb{1} + 2\mathbf{T}, \tag{5}$$

where $\mathbb{1}$ is the identity matrix. This relation follows from the fact that the S-matrix connects incoming with outgoing modes (in contrast to incident and scattered modes). While normally the incident fields in the T-matrix formalism are expanded in spherical Bessel functions (regular) and scattered fields in spherical Hankel functions (singular), the fields in S-matrix formalism are normally expanded using spherical Hankel functions, thus the additional factor of two coming from the relation between them. Please note, alternative formulations for the relation can be found, e.g., $\mathbf{S} = \mathbb{1} + \mathbf{T}$ [68]. But the validity of such expression requires some changes in the normalization of the basis function sets. The S-matrix is widely used with other basis functions as well, for example, eigenmodes of a waveguide [69] or of an optical fiber [70], and in the context of time-varying scattering processes [71]. Another example for an alternative matrix expressing the scattering problem is the previously mentioned reaction matrix, also called K-matrix, which can be obtained via $\mathbf{K} = \mathrm{i}\mathbf{T}(\mathbb{1}+\mathbf{T})^{-1}$ from the T-matrix [8].

Building upon these concepts, the T-matrix of an arrangement of scatterers can be formulated with the help of the translational addition theorem for VSWFs [72]. The following equation can be derived:

$$\mathbf{p}_{\text{local}} = (\mathbb{1} - \mathbf{C}^{(3)}_{\text{local}}\mathbf{T}_{\text{local}})^{-1}\mathbf{T}_{\text{local}}\mathbf{a}_{\text{local}}, \tag{6}$$

where "local" in the index implies that the T-matrices of all the objects comprising the total T-matrix are defined in their local coordinate systems, and $\mathbf{C}^{(3)}_{\text{local}}$ contains the translation coefficients. Moreover, other formulations to solve the same problem exist, and they all can be employed on the base of the T-matrices [73]. Finally, for many scatterers, it can be computationally more efficient to shift all the matrices to a common origin and diminish their size by doing so [74]. This is achieved by multiplying translation coefficients with the expansion coefficients as well:

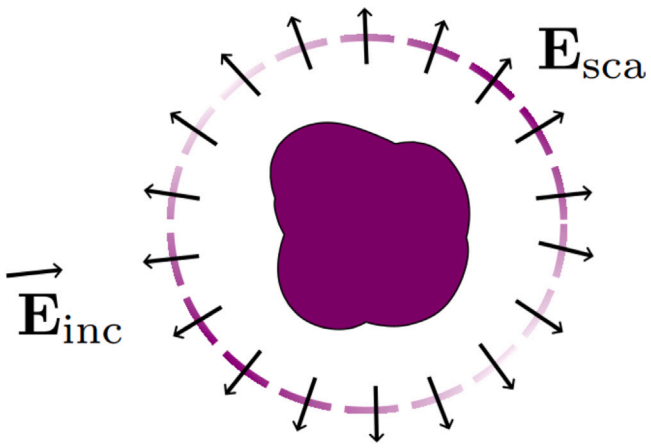

Fig. 1. Schematic representation of the scattering problem for an arbitrary scatterer in free space.

$$\mathbf{p}_{\text{global}} = \left(\mathbf{C}^{(1)}_{01}\cdots\mathbf{C}^{(1)}_{0N}\right)\mathbf{p}_{\text{local}} \tag{7}$$

and

$$\mathbf{a}_{\text{local}} = \begin{pmatrix}\mathbf{C}^{(1)}_{10}\\ \vdots \\ \mathbf{C}^{(1)}_{N0}\end{pmatrix}\mathbf{a}_{\text{global}}, \tag{8}$$

such that the global T-matrix connects the two newly defined vectors [75].

With the T-matrix at hand, useful characteristics of the scattering response can be derived directly [9], such as the orientation-averaged extinction cross-section:

$$\langle\sigma_{\text{ext}}\rangle = -\frac{2\pi}{k^2}\text{Re}\sum_{l=1}^{l_{\max}}\sum_{m=-l}^{l}\sum_{i=\pm1}T^{ii}_{l,m,l,m}, \tag{9}$$

and the orientation-averaged scattering cross-section:

$$\langle\sigma_{\text{sca}}\rangle = \frac{2\pi}{k^2}\sum_{l=1}^{l_{\max}}\sum_{l'=1}^{l_{\max}}\sum_{m=-l}^{l}\sum_{m'=-l'}^{l'}\sum_{i=\pm1}\sum_{j=\pm1}|T^{ij}_{l,m,l',m'}|^2, \tag{10}$$

where k is the wavenumber, $l \in \mathbb{N}_0$, $m \in \mathbb{Z}$ with $l \geq |m|$, and i,j indices indicate the polarization of the incident and scattered modes, respectively, and take the values of -1 and 1. These expressions were derived assuming illumination with plane waves. In the general case, the cross-sections are only proportional to the right-hand side of the equations. The cross-sections are always positive-defined.

The interaction between scatterers in an infinite periodic arrangement is also possible in this framework with the help of the Ewald summation technique [76], enabling the computation of metasurface parameters of interest, such as transmittance and reflectance [77]. The concise representation of the optical response is particularly advantageous for more complex geometries of interest that include scatterers with known T-matrices. It facilitates a considerably faster treatment than the calculation of scattering response for each particular arrangement and optical characteristic from scratch using a full-wave Maxwell's solver [78,79].

From all these discussions, we see that the T-matrix is central for the exploration of the optical properties of scattering systems. Therefore, we strive to develop a data format in the following that allows to systematically create, store, and share it.

Part II

Data format specifications

This part describes the specification of the data format for storing T-matrices. It is intended to give all the necessary details to generate valid files containing T-matrix data and to develop the necessary tools. We stress that many different formats could have been chosen. Still,

after intense discussions among the expert colleagues who authored the document at hand over an extended period of time and a dedicated workshop that took place in Bad Herrenalb, Germany in December 2023, an agreement was found on a suitable set of specifications. The following details are motivated by some basic requirements that we impose on the storage format. In particular, it needs to include:

1. Clear definitions of the T-matrix, especially regarding the vector spherical waves and their normalization,
2. Unique descriptions of the properties of a scatterer,
3. Comprehensive descriptions of the computation method for reproducibility, and
4. An accessible storage format to support different software.

Each file can have optional components along with the required components specified. We describe the different components of the data file, assuming a hierarchical data structure in the HDF5 format. We provide a brief overview of that format in Appendix D, as well as an introduction of tools to convert to this standard from some existing formats. It is recommended to store the files with the extension `.tmat.h5` (or `.tmat.hdf5`) to highlight the specific structure of the file. In the file naming scheme, we recommend rounding the numerical values to the second significant digit after the comma. In the following, we demonstrate an example of the file structure for visual assistance. The font is selected to emphasize the usage of **groups**, datasets, and *attributes*.

- hdf5 file
  - *name*, *description*, *keywords*, *storage_format_version*
  - tmatrix
  - **modes**
    - l
    - m
    - polarization
  - frequency
    - *unit*
  - **embedding**
    - *name*, *description*, *keywords*
    - relative_permittivity
    - relative_permeability
  - **scatterer**
    - **material**
      - *name*, *description*, *keywords*
      - relative_permittivity
      - relative_permeability
    - **geometry**
      - *name, description, keywords, shape, unit*
      - ...
  - **computation**
    - *software*, *method*, *name*, *description*, *keywords*
    - **files**
    - mesh.XYZ
    - **method_parameters**

The required entries given in the following subsections should not be omitted, unless it is mentioned explicitly, since they provide important information concerning the T-matrix and the scatterer of interest.

3.1. T-matrix

The `/tmatrix` is the main *dataset* of the file. It contains an array of complex values with the shape (a, b_s, b_i). The last two dimensions are the $b_s \times b_i$ entries of the T-matrices with b_s the number of scattered modes and b_i the number of incident modes. The wavelength-related sweep of length a is currently the only allowed third dimension for each T-matrix. Within an individual file, no other parameters describing the computation and the scatterer should be varied, unless it concerns the dispersive nature of material parameters. This is for the sake of simplicity of reading and searching the metadata. The last two dimensions should be related to the modes in the *group* `/modes`. In other words, the *attribute* `inner_dims` is set implicitly to 2. Any other explicit value is prohibited. An optional dataset to store is the T-matrix connecting the incident fields and the internal fields. It can be saved as a separate dataset `/rmatrix`. This matrix should be associated with the same modes as the main T-matrix and have the same normalization, with all the restriction in the previous paragraph applicable to its dimensions. The background medium used in the computation of this R-matrix is assumed to be the material of the scatterer.

3.2. Name, description, and keywords

Each T-matrix has a string `/name` that concisely describes the T-matrix and possibly its distinguishing feature. It is stored as an *attribute* of `/`. While a unique notation or wording for `/name` is not required, this attribute helps in the identification and retrieval of the T-matrix as well as for its usage for machine learning and other methods of analysis. Also, a unique identifier will be assigned automatically during the upload of data to the database, which will be discussed elsewhere.

The *attribute* `/description` is a string that comprehensively outlines the object, its shape, and materials. It can also contain additional information, e.g., optimization goals that were attempted to be reached with the specific geometry. If the data was computed with a particular application in mind, the application field can be separated into optional *attribute* `/application` to simplify search in the future database. After reading the name and description, one should have a precise idea of what the T-matrix describes. A comma-separated list of important keywords can be added to the file as well as the *attribute* `/keywords`. These three attributes are generally useful to be added in the materials, computation, or geometry definitions. The keywords can be used to provide information on special properties such as symmetries. It is helpful to follow consistent naming for the symmetries, the common ones being *czinfinity* (rotational symmetry), *mirrorxyz* and all possible combinations of symmetry planes, *reciprocal*, *passive*, *lossless*. The details of their definitions are provided in Section 5.3. We stress that carefully chosen names and an elaborative description will help at a later stage to retrieve and reuse a given entry of the dataset. To ensure optimal compatibility with machine learning frameworks, adherence to a standardized keyword and property scheme in the metadata can be supported by using machine learning methods such as large language models to identify unifying/reoccurring elements in the metadata.

3.3. (Angular) frequency, vacuum wavelength, or (angular) wavenumber

The frequency of the electromagnetic waves needs to be known to understand the T-matrix. In different communities, different ways of describing this information are common. For machine learning applications, it is essential that the frequency-related data are consistently formatted and includes clear units. This ensures that models can accurately interpret and utilize the data for training and prediction tasks. It is allowed to submit not only real values, but also complex values for methods that exploit quasi-normal modes [80]. Adding multiple descriptions, e.g., frequency and wavelength data, is discouraged. The relation between the datasets is defined by

$$\frac{2\pi\nu}{c_0} = \frac{\omega}{c_0} = \frac{2\pi}{\lambda_0} = 2\pi\tilde{\nu_0} = k_0$$

with the speed of light in vacuum $c_0 = 299\,792\,458\,\mathrm{m\,s^{-1}}$ (see Table 1).

The *dataset* has the required *attribute* `unit`, that defines the SI unit and prefix of the data as a string, e.g., “THz” for the (angular) frequency or “cm^{-1}” for the (angular) wavenumber. Frequencies can also be expressed in inverse seconds, for example, “s^{-1}”. The unit micrometers can be expressed as “µm” or “um”. There is no unit assumed by default. The full list of accepted units is given in Appendix E.

Table 1
Different ways to define the frequency.

Dataset name	Name	symbol
`/frequency`	Frequency	ν
`/angular_frequency`	Angular frequency	ω
`/vacuum_wavelength`	Vacuum wavelength	λ
`/vacuum_wavenumber`	Vacuum wavenumber	$\tilde{\nu}$
`/angular_vacuum_wavenumber`	Angular vacuum wavenumber	k

3.4. Modes

Since there are many ways to sort the entries of the T-matrix, the related mode of each row and column has to be explicitly given. This data is collected in the *group* `/modes`. The definition of the modes is given in Appendix C. These modes are indexed by a degree $l \in \mathbb{N}$ and by an order $m \in \mathbb{Z}$ with $|m| \leq l$. This data is given in the corresponding *datasets* `/modes/l` and `/modes/m`. Additionally, the polarization is given in `/modes/polarization`. To avoid any ambiguity, the polarization of each mode is given as a string "electric" (also known as "TM", for modes $\mathbf{N}_{lm}^{(n)}$), followed by "magnetic" ("TE", for modes $\mathbf{M}_{lm}^{(n)}$). If the helicity basis is used, the polarization is defined by alternating "positive" and "negative" for $\mathbf{A}_{lm+}^{(n)}$ and $\mathbf{A}_{lm-}^{(n)}$, respectively. The incident and scattered modes can be separated into `/modes/l_incident` and `/modes/l_scattered` (and likewise for the other items in the *group*). When present, they take precedence. This splitting is required if a different number of incident and scattered modes are used.

One must store the modes in a fixed order, and for visual assistance, a table can be found in Appendix C.2. The parameters sweep in the following sequence: the degree l ranges from 1 to $l_{\max}$, for each fixed l there are modes with order m traversing from $-l$ to l, and, lastly, for each set of (l, m) there are two alternating modes "electric" and "magnetic" (or "positive" and "negative"). As outlined above, the term "electric" multipole field is not to be confused with "transverse electric" field. The modes are required to be uniform for one file. Thus, they are – unless they are scalar – always given as one-dimensional arrays of length b_s and b_i.

Finally, for the special case of a cluster of scatterers when the T-matrix is defined in the local basis of each scatterer, the *datasets* `/modes/positions` and `/modes/index` should be added to provide the correspondence between the T-matrix entries and the local coordinate systems. The index of incident and scattered modes can also be separated into `/modes/index_incident` and `/modes/index_scattered`.

3.5. Storage format version

It is required to specify the version of the storage format in the *attribute* `/storage_format_version` as a string. With this document, we fix the standard format at "v1", which is also reflected as the first two characters of the version specification of the T-matrix data format repository [81] in `software` *attribute* of `/computation`.

3.6. Scatterer

The T-matrix can be computed for a cluster of scatterers. Information on each scatterer is then collected in separate groups. Each group comprises the information on both geometry and material. The names of these groups are distinguished by assigning a number to each name, e.g., `/scatterer_X`. For a single scatterer, `/scatterer` is accepted as the name of the group. It will be further referred to as `/NAME`.

3.6.1. Material

The material is a *subgroup* of the *group* `/NAME` describing the scatterer. In case of multiple scatterers, each `material` *subgroup* is inside the corresponding scatterer *group*. The surrounding medium is not included in the scatterer *group*. The description of the material is typically done by providing the permittivity and permeability or by defining the refractive index and the impedance. Each material *group* can be annotated with a `name`, `description`, and `keywords` as *attribute*. The recommended way to add the `name` *attribute* of the material is to specify both the chemical formula and the common name if available (e.g., "Au, Gold"). The definitions here assume linear, homogeneous materials. The specific case of a layered structure can be accommodated by entering the material parameters as arrays, and separating the attributes corresponding to each layer with a semicolon in the string. The materials can be dispersive. In each scatterer group, the wavelength dependence should occur along the first axis of the datasets describing the material parameters. The keywords can also contain the type of material, e.g., dielectric. Special information, such as nonlocality, can be added as a `keyword`, and the details can be included in the `description`. Then, the local permittivity contribution is included into the standard datasets introduced below for material parameters. If the values for the relative permittivity and permeability were used from an external source, this can be specified in the `reference` *attribute*, in the form of a reference paper or other link as a string. Alternatively, if the values were measured experimentally, the original datasets should be added under the *subgroup* `/NAME/material/experimental_data`, which includes the material parameters as described below and the corresponding frequencies/wavelengths with specified units. The method used for interpolation should be added as well in a string format as an *attribute* `interpolation`. We separate cases with isotropic and anisotropic media, as well as cases with and without magnetoelectric coupling. This leads to the four classes of isotropic, biisotropic, anisotropic, and bianisotropic materials, that we describe below (see also Appendix A).

Isotropic materials. The material parameters can either be a relative permittivity and permeability or a refractive index and relative impedance. Either of the pairs are to be specified fully, and the values are not set by default to vacuum. If any of the former two parameters are present, all occurrences of refractive index and relative impedance will be ignored. The relative permittivity ϵ is given in `/NAME/material/relative_permittivity` and the relative permeability μ is given in `/NAME/material/relative_permeability`. Alternatively, neither permittivity nor permeability is defined, and the refractive index $n = \sqrt{\epsilon\mu}$ in `/NAME/material/refractive_index` and the relative impedance $Z = \sqrt{\frac{\mu}{\epsilon}}$ in `/NAME/material/relative_impedance` are defined.

Anisotropic materials. For an anisotropic material, the above-mentioned *datasets* can be used, except the relative impedance. If any of the *datasets* is used for an anisotropic material, the *attribute* `inner_dims` has to be defined. Otherwise, by default, `inner_dims` is assumed to be 0. If it is set to 1, the values of the last dimension are taken as the diagonal values of the 3-by-3 tensor. If it is set to 2, the *dataset* contains the full tensor. Additionally, the *attribute* `coordinate_system` can be set to either "Cartesian", "cylindrical", or "spherical" (see Appendix B) to specify the chosen set of coordinates. The default is "Cartesian". In the spherical or cylindrical case, the z-axis takes a special role by default (axis of rotation of axis-symmetric objects). This default can be changed by appending "x" or "y" to the *attribute*, e.g., "cylindricalx".

Biisotropic materials. A biisotropic material has up to two additional parameters as an isotropic material. One of these two additional parameters is `/NAME/material/chirality`. Moreover, a second additional

parameter is `/NAME/material/nonreciprocity`. Both parameters have the default value 0. Biisotropic parameters can only be defined in conjunction with relative permittivity and relative permeability. Using the refractive index or relative impedance is prohibited in that case.

Bianisotropic materials. A bianisotropic material can be considered by using the parameters of the biisotropic materials in conjunction with the `inner_dims` *attribute*. Alternatively, it can be defined by giving the 6-by-6 bianisotropic tensor in the *dataset* `/NAME/material/bianisotropy`. Other material parameters are not permitted in that case. The *attribute* `inner_dims` has to be set to 1 or 2 to specify if the bianisotropy is given as full tensor or as diagonal values only. (With `inner_dims` set to 1 the material is, in fact, not bianisotropic but anisotropic.)

3.6.2. Geometry

The geometry of the objects described by the T-matrix can be defined as a *subgroup* `/geometry` of the *group* `/NAME`. There are various ways to describe the geometry. Therefore, this section can be adapted. Again, a `name`, `description`, and `keywords` can be added as *attributes* for this *group*. For an arrangement of scatterers, the *dataset* `position` is to be indicated explicitly for each scatterer. Position at the center of the coordinate system is assumed by default. The coordinates defined in `position` specifies the center of the smallest circumscribing sphere of the scatterer.

If the scatterer has a simple geometric shape, the parameters from Table 2 should be used to describe it. The shape should then be specified by adding the *attribute* `shape`. Note that by default for the rotationally symmetric objects, the symmetry axis is the z-axis. In the table, core–shell sphere is separated into a separate entry, but for a general case of a layered scatterer, it is admissible to set the final shape of the scatterer as the `shape` *attribute* and add an array of geometrical parameters together with an array of material properties corresponding to each layer. The accepted convention is to measure the radius of the shell from the center of the whole object [82,83]. If the same unit can be used to describe all geometrical parameters, it can be specified as an *attribute* `unit` of the *group*. The unit can also be an *attribute* of each individual parameter.

Clearly, the provided list of basic shapes cannot cover all scatterers that may be of interest, which limits the search capabilities within the future database. To address this, we strongly emphasize the importance of including the mesh in the data file, as detailed later in the section. The previously mentioned `name` attribute can be considered as a possible entry to specify names for shapes not included in the basic shapes list. Upon reasonable request, new shapes will be added to the current list, which will be published in the GitHub repository.

The *dataset* `/NAME/geometry/expansion_center` is expected to specify the center of expansion of VSWFs used in the computation of the T-matrix, and by default, it is assumed to be at the center of the coordinate system. For the listed objects with basic shapes, the coordinate system is fixed. For any rotated object with a basic shape, its final orientation can be specified in `/NAME/geometry/euler_angles` *dataset* using the standard Euler angles, following the extrinsic z-y-z notation. By default, these values are assumed to be zeros. For a freeform object, the orientation can only be inspected from the mesh file, since the standard orientation is not defined in the general case.

For a general representation of the scatterer, a mesh file has to be defined. The only possible reason for a mesh to be completely absent in the file is the choice of a semi-analytical method for computations (e.g., EBCM, Mie). In this case, *semi-analytical* should appear in `keywords` *attribute* of `/computation`, and the presence of the *attribute* `shape` and the corresponding *datasets* of geometrical parameters is obligatory. The mesh can be specified in `/NAME/geometry/mesh.XYZ` or `/computation/mesh.XYZ`, where XYZ stands for the file extension, depending on what is physically more reasonable. For example, for multiple scatterers and a single mesh file, it is sufficient to specify the mesh in `/computation`. The mesh should be ideally in a common format, e.g., msh or STL. However, STL can only define surface meshes and no volumetric ones, so for complicated structures, such as ones consisting of multiple materials, other formats can be used. To simplify access to the mesh for the user, a *softlink* `/mesh` to the mesh location can be provided at the top level, if possible. In exceptional cases, when there is no possibility to describe the mesh using a common format without losing information about the mesh or deforming it, the specific list of mesh parameters datasets can be stored inside a *group* `mesh`. A unit of length has to be added to the mesh definition as an *attribute* `unit`. The *attributes* `name`, `description`, and `keywords` can provide additional information.

3.7. Embedding

The *group* `/embedding` is a stand-alone *group* describing the embedding medium that has the same structure as the *group* `/material`. Anisotropic or non-reciprocal materials are not allowed for the embedding. Chiral materials are only allowed with the helicity basis for the T-matrix.

3.8. Computation

To reproduce the data of the T-matrix, this section should contain information on the way how it was computed. Because there are many ways of obtaining a T-matrix, this section should be adapted to different situations. The used software and its version are specified in the *attribute* `software`. It is required to add all the used software in a comma-separated string, including the version, and in particular, the repository [81], where the template scripts are located, is to be defined in the form "`tmatrix_data_format=vx.x.x`". Because differences between implementations of the HDF5 wrappers can occur, it is required to specify the program and its version used to create the HDF5 file in the same way as for other software, e.g., "`h5py=version`". In case several scripts realize different approaches to compute the T-matrix using the same external software, or there is an external repository hosting the original scripts, the specific links can be added in the `reference` *attribute* of `/computation`. The *attribute* `method` of `/computation` describes the computational technique that the software implements. The best approach is to include in a comma-separated string both the abbreviation and the full name of the method.

Additionally, `/computation` should contain the files needed to reproduce the data in a dedicated *group* `/computation/files`, e.g., full Python or other programming language script source codes. To simplify the search for parameters used in the specific computation, a *group* `/computation/method_parameters` includes as *datasets* all the specific numerical values, with the names of the *datasets* following the ones used by the software as closely as possible. It is important for consistency to use the parameter names from the template scripts of the repository. The parameters typically include information about mesh discretization and accuracy.

The subtle aspect, which can become significant for the analysis of the data, is the question of which entries of the T-matrix do not include any numerical inaccuracies. Some computational methods can leverage symmetries of the objects, such that, for example, the response of a rotationally symmetric 3D object can be reduced to solving a 2D problem. Then, the entries that are zero due to symmetry automatically are set in the final T-matrix and are not the result of a simulation. To store this information, we suggest to add a mask in a *dataset* `/computation/analytical_zeros` of the same shape as the T-matrix, where 0 stands for analytical zero entries, and 1 is set otherwise. This specification is different from the keywords of the T-matrix, where one can specify the symmetry of the object since the

Table 2
Basic three-dimensional shapes and their defining parameters.

Shape	Parameters
sphere	radius
cut_sphere [34]	radius, height
core_shell_sphere	radius_0 , radius_N
spheroid	radiusxy, radiusz
ellipsoid	radiusx, radiusy, radiusz
superellipsoid [45]	radiusx, radiusy, radiusz, n_parm, e_parm
cylinder[a]	radius, height
cone	radius_top, radius_bottom, height
ring	radius_major, radius_minor, height
torus	radius_major, radius_minor
cube	length
rectangular_cuboid	lengthx, lengthy, lengthz
helix[b,c,d].	radius_helix, radius_wire, pitch, number_turns, termination, handedness
pyramid	n_edges, radius, height, angle, apex_shift
regular_prism	n_edges, radius, height, shift
wedge	lengthx, lengthy, lengthz, deltax, deltay
convex_polyhedron	points

[a] Rotational axis is along z by default for rotationally symmetric scatterers. For different orientations, please specify the Euler angles additionally as described in the main text.
[b] The pitch is oriented along the z-axis by default. For different orientations, please specify the Euler angles additionally as described in the main text.
[c] Since various definitions are possible, we stress here that the normal plane cross-section is the default. If this is not the case, please specify in the description *attribute* of the geometry.
[d] termination can be "spherical" or "flat", handedness can be "right" or "left". The defaults are "flat" and "right".

rotationally symmetric object can still be computed with a method that performs full 3D computation and produces only approximate zeros. Finally, it is possible to add the mesh used in the computation in this group, as described in the geometry section.

Part III

Generation and validation

This part describes methods to generate files according to the standard and validate them.

4. Generating files

We stress at first that we provide files in a dedicated repository [81] that essentially consists of scripts that can be run to (a) compute T-matrices and (b) to assemble them in a way into files so that these files inherently agree with the specifications of the data format. So in a nutshell, if you use these files, you will obtain results that intrinsically agree with all the requirements. We describe a few of these methods in the following, but the developments are rapid and with time passing, we expect a larger number of tools to be adapted to compute and store these T-matrices in the required data format. Therefore, the list of current tools can only be considered as a snapshot.

Incorporating the T-matrix formalism with other numerical methods and different software implementations is of further great use to the community. We encourage everybody to consider adapting their tools so that they can be used for the same purpose and making these files available to the public via the mentioned repository [81].

A first sanity check for the correct implementation is to demonstrate that the computed T-matrix indeed is constructed such that it stores the response of the object for all illumination directions. It is known that the T-matrix of an object itself does not depend on the incident illumination, while the scattering coefficients do. Therefore, an additional check is recommended for the scattering cross-section of the object at a specific illumination using the full-wave simulation software and the computed T-matrix. This also will increasingly apply to machine learning tools for synthetic data generation. Over time, example use cases for synthetic data generation tools can be added. These tools may systematically vary parameters to create diverse and comprehensive datasets, enhancing the training of machine learning models and closing gaps in the available data.

To add a new method to the repository, the files to produce the T-matrix should be submitted as a pull request to the repository together with benchmarking results for some basic examples. They are manually inspected afterward. Once this verification has been done, the new codes to generate T-matrices will be made available to the public via the repository. Ideally, a small description is provided in agreement with the documentation for the already established methods, as outlined in the following.

In the following, we demonstrate the extraction of the T-matrix using various software.

4.1. JCMsuite

The program JCMsuite has the built-in capability to illuminate objects with vector spherical waves and calculate the decomposition of the scattered fields. Thus, it is well suited to compute T-matrix coefficients [84]. It uses the finite element method (FEM) to compute the scattering response and can be applied to arbitrary shapes. For illumination of a scattering object by multiple sources of the same frequency, it allows us to generate multiple independent solutions at the computational cost of a single solution by reusing the inverted system matrix [85]. A similar approach for T-matrix extraction from commercial software (Finite Element based HFSS) using different incident angles has been developed by Huang et al. [86]. It also provides the full definition of the bianisotropic tensor. However, the decomposition is done in a parity basis, so the embedding medium has to be achiral.

Generally, a simulation with JCMsuite is controlled by several files. These are, at minimum, a project, a source, a material, and a layout file. Combining these files with a script using MATLAB or Python to perform, e.g., automatic parameter sweeps is possible. Since all these files are text-based, including full documentation of the simulation setup in the HDF5 file is simple.

In our examples, the first three files, the project, sources, and material file, stay mostly the same. The project file typically looks like

```
Project {
  Electromagnetics {
    TimeHarmonic {
      Scattering {
        FourierModeRange = [0, %(degree_max)e]
        Accuracy {
          Precision = %(precision)e
          Refinement {
            MaxNumberSteps = %(max_refinements)e
            Strategy = PAdaptive
          }
          FiniteElementDegree = %(fem_degree)e
        }
```

```
      }
    }
  }
}

PostProcess {
  MultipoleExpansion {
    FieldBagPath = "project_results/fieldbag.jcm
    "
    OutputFileName = "project_results/vsh.jcm"
    MultipoleDegree = %(degree_max)e
  }
}
```

Listing 1: project.jcmpt

and includes a general setup of the type of calculation to perform, some settings of numerical parameters which control the solution accuracy, and the post-process to perform the decomposition. In this example, several parameters are taken from the Python script. In the configuration language of JCMsuite, they are indicated by the percent symbol, a variable name in brackets, and a variable type indicator letter. Besides parameters that define the accuracy of the FEM calculation, the maximal multipole order has to be set in the variable `degree_max`.

Next, the source file contains the definition

```
<?
for degree in range(1, keys['degree_max'] + 1):
    for order in range(-degree, degree + 1):
        for pol in 'MN':
            keys['degree'] = degree
            keys['order'] = order
            keys['pol'] = pol
            ?>

SourceBag {
  Source {
    ElectricFieldStrength {
      VectorSphericalWaveFunction {
        Coefficient = 1
        Lambda0 = %(lambda0)e
        MultipoleDegree = %(degree)i
        MultipoleOrder =  %(order)i
        Type = %(pol)s
      }
    }
  }
}

<?
?>
```

Listing 2: sources.jcmt

which uses simply a loop over vector spherical waves up to the maximum multipole order. The correct wavelength must also be set in the variable `lambda0`. The materials file includes a loop over all material parameters set in the Python script.

```
<?
for mat, er in enumerate(keys['epsilon']):
    keys['permittivity'] = er
    keys['mat_id'] = mat + 1
    keys['mat_name'] = f'Material_{mat}'
    ?>

Material {
  Name = "%(mat_name)s"
  DomainId = %(mat_id)e
  RelPermittivity = %(permittivity)e
}

<?
?>
```

Listing 3: materials.jcmt

At least the relative permittivity has to be set. Relative permeability and the chirality parameter are optional. The first material in the list is taken as the embedding material, which is associated with the domain ID 1 within the setup of JCMsuite used here.

Finally, the layout file contains the description of the geometry. In the example, we consider a single sphere. Due to the rotational symmetry, it is possible to restrict the FEM calculation to a two-dimensional domain. According to the domain IDs in the materials file, we set the background to domain ID 1 and then include the (semi-)circle for the object.

```
Layout2D {
  CoordinateSystem = Cylindrical
  UnitOfLength = %(uol)e
  MeshOptions {
    MaximumSideLength = %(maxsl)e
  }
  Objects {
    Parallelogram {
      Priority = -1
      DomainId = 1
      Width = %(domain_radius)e
      Height = %(domain_z)e
      Port = West
      Boundary {
        Class = Transparent
      }
    }
    Circle {
      DomainId = 2
      Priority = 1
      Radius = %(radius)e
      MeshOptions {
        MaximumSideLength = %(object_maxsl)e
      }
    }
  }
}
```

Listing 4: layout.sphere.jcmt

The whole computation is controlled from a Python script. While the script itself is a somewhat longer, for the setup and subsequent run of the calculation, the relevant sections are the definitions of the keys used to fill the open variables in the JCMsuite files and the loop to start all computations.

```
    keys = {
        "uol": 1e-9,  # unit of length (1 = m, 1
    e-9 = nm)
        "epsilon": [1, 4],  # First entry
    embedding
        "mu": [1, 1],
        "radius": 100,
        "degree_max": 5,
    }
    keys_method = {
        "domain_radius": 125,
        "domain_z": 250,
        "maximum_sidelength_domain": 10,
        "maximum_sidelength_object": 5,
        "precision": 1e-7,
        "max_refinements": 3,
        "fem_degree": 2,
    }
    keys.update(keys_method)
```

Listing 5: Snippet from Python script sphere_jcmsuite.py – defining the keys

```
    jobids = []
    for i, freq in enumerate(freqs):
        keys["lambda0"] = c0 * keys["uol"] / freqs[i]
        jobid = jcmwave.solve(
            "project.jcmp",
            keys=keys,
            working_dir=f"{working_dir}/job{i}",
            jcmt_pattern=jcmt_pattern,
        )
        jobids.append(jobid)
    results, logs = jcmwave.daemon.wait(jobids)
    jcmwave.daemon.shutdown()

    tmats, _, ls, ms, pols = jcm_tools.extract_tmatrix_jcm(results)
```

Listing 6: Snippet from Python script sphere_jcmsuite.py – loop

In the final line of the listing, we extract the T-matrices from the results using a custom function. In the last lines of the original script, the data is stored using the module `tmatrix_tools.py` that provides a collection of functions to create a standard conforming T-matrix file.

4.2. COMSOL

The same methodology can be implemented in COMSOL Multiphysics, one of the most used FEM software programs. To demonstrate the capabilities of COMSOL, we will concentrate on the case where symmetry considerations can cut down the calculation time. The simplest example would be a sphere, which has rotation symmetry. The full Java script version was made available to the public repository [81] as well. The compiled scripts, with commented-out last lines to avoid immediate running, can then be opened and modified manually in the GUI. For demonstration purposes, the screenshots of COMSOL's GUI are included in this section.

The first step is to define the parameters, and here, the additional circumscribing sphere is introduced, defined as the decomposition radius. This should fully include the object and it is the surface where the decomposition of the fields into VSWFs will be performed. The domain has to be larger than the decomposition radius and is followed by the perfectly matched layer (PML) (see Fig. 2). Rotationally symmetric examples require additional care in the choice of the domain size, such that it is comparable to the wavelength [87]. Under parameter decomposition, the maximum desired multipole order for the incident waves and scattered waves is defined. The customized part of the model is the function definitions for the incident waves. To obtain a T-matrix of an arbitrary object, it should be illuminated with VSWFs up to the desired multipole order. Therefore, definitions of Legendre polynomials, VSWFs, spherical Bessel, and spherical Hankel functions are necessary. As an additional note, the associated Legendre polynomials are implemented in versions starting from COMSOL 5.5., for the earlier versions one can use manually implemented functions up to degree 5, such that (3D) T-matrix computations are possible only up to degree 4.

In the definition of the geometry of the object, half of the circle is given, and the region of the circumscribing sphere is technically included in the domain, such that the domain has a few layers. The chirality parameter can be set to a non-zero value. The Variables Decomposition includes the integration formulas of VSWFs and scattered fields on the circumscribing sphere. In this model, the scattered field formulation is used, so the resulting field is a superposition of the background field and the scattered field. The definition of the background electric field, the solution of Maxwell's equations in the absence of the scatterer, is making use of the special functions defined previously (see Fig. 3). An alternative total field formulation could also be possible. Next, we choose axial symmetry for all boundaries. The equation view reveals the whole set of expressions. The mesh size can be controlled as well.

To finally run the computation, two studies are prepared. The first study computes the solution to the scattering problem. One can select a range of frequencies for the calculation, in addition to the nested parameter sweeps for the order and polarization of the vector spherical waves. The second study post-processes these results, evaluates the necessary integrals on the decomposition surface, and computes the T-matrix entries. The results obtained from the second study are in the form of a list of variables "ap", and "am", which stand for positive and negative helicity. In the final format used for the storage, one has to interleave these two matrices of coefficients. The Python script to convert the results to `.tmat.h5` format is provided as well in [81].

4.3. ONELAB

In this section, we demonstrate the retrieval of T-matrices with the open-source FEM solver ONELAB [88]. It is based on the mesh generator Gmsh [89] and the finite element solver GetDP [90]. The underlying theoretical treatment and the numerical implementation details are reported in [29]. In line with the previous sections, the VSWFs are implemented directly as the illumination fields. The simulations are conducted using a scattered field formulation of FEM, which makes it possible to locate the illumination sources within the scatterer. In this implementation, the final integration with VSWFs is performed on a spherical surface. As exemplified in the snippet 7 from a single sphere T-matrix computation code, after the initialization of the ONELAB client and definition of the parameters, the Gmsh and GetDP programs are run as subclients over a range of wavelengths. The results are stored in a particular ordering using the `_get_postprocess` function. Then, the `tmatrix_tools.py` utility module is used to store the data in HDF5 file.

```
c.setNumber("1Geometry/00Scatterer shape", value=0)
c.setNumber("1Geometry/01ellipsoid X-radius [nm]", value=radius)
c.setNumber("1Geometry/02ellipsoid Y-radius [nm]", value=radius)
c.setNumber("1Geometry/03ellipsoid Z-radius [nm]", value=radius)
c.setNumber(
    "3Electromagnetic parameters/00scatterer permittivity (real) []",
    value=np.real(eps_sph),
)
c.setNumber(
    "3Electromagnetic parameters/01scatterer permittivity (imag) []",
    value=np.imag(eps_sph),
)
c.setNumber("3Electromagnetic parameters/08n_max integer", value=lmax)
c.setNumber("4Mesh size and PMLs parameters/02 mesh size", value=ms)

tmats = []
for lambda0 in lambdas:
    c.setNumber("3Electromagnetic parameters/04 wavelength [nm]", value=int(lambda0))

    # run gmsh as a subclient
    c.runSubClient("myGmsh", mygmsh + " " + mymodel_geo + " -3 -o " + mymodel_msh)

    # run getdp as a subclient
    c.runSubClient(
        "myGetDP",
        mygetdp
        + " "
```

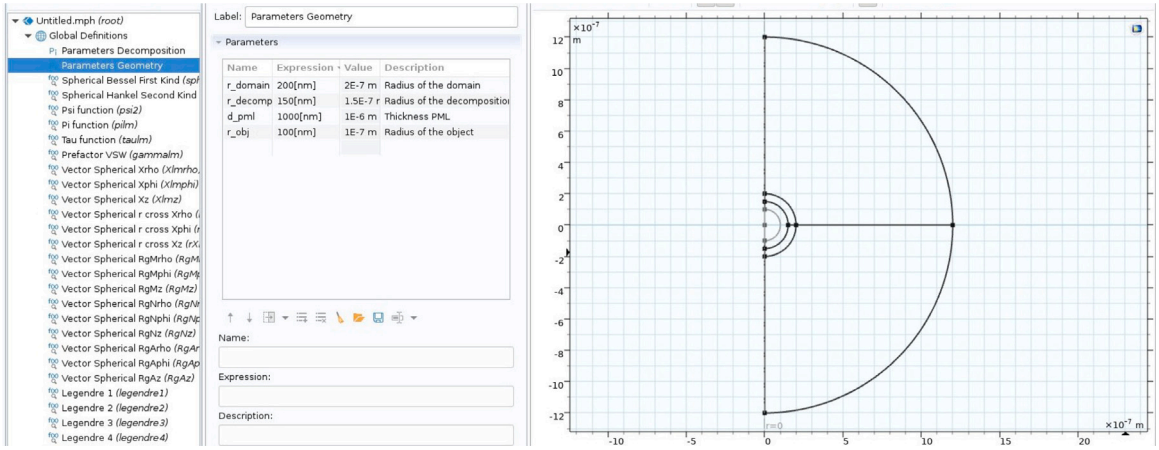

Fig. 2. COMSOL: Specification of the geometry of the problem in the GUI.

Fig. 3. COMSOL: Background electric field definition in the GUI.

```
        + mymodel_pro
        + " -pre res_VPWall_helmholtz_vector -msh "
        + mymodel_msh
        + " -cal -petsc_prealloc 200",
    )

    # build T-matrix
    single = treams.PhysicsArray(
        _get_postprocess(int(lambda0)),
        basis=treams.SphericalWaveBasis.default(lmax),
        k0=2 * np.pi / lambda0,
        material=1,
        modetype=("singular", "regular"),
        poltype="parity",
    )
    tmats.append(single)


# Store data in the standard format .tmat.h5
```

Listing 7: Snippet from sphere_onelab.py

The source files are provided in [81].

4.4. nanobem

nanobem is an open-source MATLAB toolbox, aimed among other functionalities at solving Maxwell's equations using the boundary element method (BEM). In this implementation, the method is based on a Galerkin scheme with Raviart–Thomas basis functions [91]. The material of the scatterers is assumed to be linear, homogeneous, and local, separated between different media via abrupt interfaces. For a detailed review of the toolbox, see [92,93]. In the following, we solely demonstrate how the construction of a T-matrix is realized using the toolbox.

Unlike in the FEM method, the computation of the field in the BEM method is performed at the surface of the object. For this reason, the problem is reformulated, and the previously defined coefficients include integration not on the circumscribing sphere, but on the surface of the object. The surface boundary is denoted as $\partial\Omega$, and the normal to the surface boundary is $\hat{\mathbf{n}}$. The representation formula allows us to compute the field at position $\mathbf{r}$ by surface integration:

$$\mathbf{E}_{\mathrm{sca}}(\mathbf{r}) = \oint_{\partial\Omega} \{\mathrm{i}\mu\omega G(\mathbf{r},\mathbf{r}') \cdot \hat{\mathbf{n}} \times \mathbf{H}(\mathbf{r}') - [\nabla \times G(\mathbf{r},\mathbf{r}')] \cdot \hat{\mathbf{n}} \times \mathbf{E}(\mathbf{r}')\} \mathrm{d}S' , \qquad (11)$$

where $G(\mathbf{r}, \mathbf{r}')$ is the dyadic Green's function expansion in the basis of transverse vector spherical waves (Eq. 7.3.40 of [94]):

$$G(\mathbf{r}, \mathbf{r}') = ik \sum_{l,m} \left(\mathbf{M}_{lm}(\mathbf{r}) \mathbf{M}^{\dagger}_{lm}(\mathbf{r}') + \mathbf{N}_{lm}(\mathbf{r}) \mathbf{N}^{\dagger}_{lm}(\mathbf{r}') \right). \tag{12}$$

For the T-matrix calculation we always have $r > r'$, therefore $\mathbf{M}_{lm}(\mathbf{r})$, $\mathbf{N}_{lm}(\mathbf{r})$ contain the spherical Hankel, while $\mathbf{M}^{\dagger}_{lm}(\mathbf{r}')$, $\mathbf{N}^{\dagger}_{lm}(\mathbf{r}')$ the spherical Bessel functions. From this equation, the T-matrix elements can be computed by (i) illuminating the scatterer with vector spherical waves, and (ii) by integrating the inner product between the induced tangential electromagnetic fields and $\mathbf{M}^{\dagger}_{lm}(\mathbf{r}')$, $\mathbf{N}^{\dagger}_{lm}(\mathbf{r}')$ over the boundary of the scatterer.

With the numerically implemented formulas, the computation of the T-matrix can be exemplified in the following code snippet.

```
mat1 = Material( 1, 1 );
mat2 = Material( 9, 1 );
%  material vector
mat = [ mat1, mat2 ];

%  discretized sphere boundary with 256 vertices
diameter = 160;
p = trisphere( 256, diameter );
%  boundary elements with linear shape functions
tau = BoundaryEdge( mat, p, [ 2, 1 ] );

%  wavenumber of light in vacuum
k0 = 2 * pi / 1000;
%  BEM solver and T-matrix solver
lmax = 4;
bem = galerkin.bemsolver( tau, 'order', [] );
tsolver = multipole.tsolver( mat, 1, lmax );
%  T-matrix
sol = bem \ tsolver( tau, k0 );
t1 = eval( tsolver, sol );

%  additional information for H5 file
info = multipole.h5info( tau );
info.name = 'Sphere';
info.description = 'Single sphere and single
    wavelength';
info.matname = [ 'Embedding medium',
'Dielectric medium of sphere' ];
%  save T-matrix
fout = 'tmatrix_sphere.tmat.h5';
```

Listing 8: Snippet from demomulti01.m: T-matrix computation for a sphere using BEM

In the first few lines, the parameters of the object are defined, including the material and geometry. The first material in the list corresponds to the background medium, and the second to the scatterer. The mesh triangulation is specified as the next step, here a triangular boundary element shape is selected. Next, for the desired wavelength and number of multipole orders indicated, the method `galerkin.bemsolver` is used to initialize the solver object, and `multipole.tsolver` initializes the T-matrix object. The variable `sol` contains the solution obtained by the BEM method, which includes the tangential electric and magnetic fields. The `eval` function uses the computed fields to calculate the T-matrix entries. The T-matrix is finally stored in the HDF5 file together with the required metadata. Further information can be obtained from the help pages of the toolbox.

4.5. ADDA

ADDA [95] is a robust open-source implementation of the DDA [96, 97]. It can handle particles of any shape and composition, including non-spherical and inhomogeneous particles, as well as particles with material anisotropy. It is optimized for high performance, supporting distributed memory clusters, multi-core processors, and GPU acceleration. This enables the analysis of large particles with high refractive indices compared to other DDA-based software tools. This acceleration is particularly important, as calculating the T-matrix requires simulations for multiple illuminations.

Although illumination with VSWFs is currently not available in ADDA, it is possible to retrieve the T-matrix by determining the scattered field from multiple plane wave illuminations. To that end, we adopt the approach proposed by Fruhnert et al. [98], which involves the decomposition of each incident field and the corresponding scattered fields into VSWFs. The T-matrix is then obtained by solving an inverse problem that relates the coefficients of the scattered and incident field expansions into adequate VSWFs for K different illuminations:

$$\left(\mathbf{p}^{(1)}\mathbf{p}^{(2)}\cdots\mathbf{p}^{(K)}\right) = \mathbf{T}_K \cdot \left(\mathbf{a}^{(1)}\mathbf{a}^{(2)}\cdots\mathbf{a}^{(K)}\right). \tag{13}$$

The initial method proposed in Ref. [98] expands the fields on a spherical shell enclosing the analyzed nanoparticle. In the following, we rely on the scattered far-field instead, as this quantity is more readily available in ADDA and is generally easier to compute. The core quantity in ADDA is the polarization of each dipole ($\mathbf{P}_i$) that is calculated self-consistently based on the dipole polarizability ($\overline{\boldsymbol{\alpha}}_i$) and Green's tensor ($\overline{\mathbf{G}}_{ij}$) for a given incident field on each dipole ($\mathbf{E}^{\text{inc}}_i$) [96]:

$$\overline{\boldsymbol{\alpha}}_i^{-1}\mathbf{P}_i - \sum_{j\neq i} \overline{\mathbf{G}}_{ij}\mathbf{P}_j = \mathbf{E}^{\text{inc}}_i. \tag{14}$$

Once the polarization is determined, the scattered far-field is obtained as:

$$\mathbf{E}_{\text{sca}}(\mathbf{r}) = \frac{\exp(\mathrm{i}kr)}{-\mathrm{i}kr}\mathbf{F}(\hat{\mathbf{r}}), \tag{15}$$

where the scattering amplitude $\mathbf{F}$ depends only on the scattering direction $\hat{\mathbf{r}} = \mathbf{r}/r$

$$\mathbf{F}(\hat{\mathbf{r}}) = -\mathrm{i}k^3(\bar{\mathbf{I}} - \hat{\mathbf{r}}\otimes\hat{\mathbf{r}}) \sum_i \mathbf{P}_i \exp\left(-\mathrm{i}k\mathbf{r}_i\cdot\hat{\mathbf{r}}\right). \tag{16}$$

Here, $\bar{\mathbf{I}}$ is the identity tensor and $\hat{\mathbf{r}}\otimes\hat{\mathbf{r}}$ projects any vector on $\hat{\mathbf{r}}$. Note that Eqs. (14)–(16) are based on the Gaussian-CGS system of units, as the one employed in ADDA. However, that is not relevant for the finally computed quantities, like the T-matrix and cross-sections.

To find the expansion coefficients, we leverage the far-field limit of the VSWFs (see Eq. (C.3)) and cancel the common dependence on r in these functions and $\mathbf{E}_{\text{sca}}(\mathbf{r})$. Then, Eq. (16) transforms into:

$$\mathbf{F}(\hat{\mathbf{r}}) = -k^2 \sum_{l=1}^{\infty}\sum_{m=-l}^{l} (-\mathrm{i})^l \left[p^{\text{e}}_{lm}\mathbf{Y}_{lm}(\hat{\mathbf{r}}) + p^{\text{m}}_{lm}\mathbf{X}_{lm}(\hat{\mathbf{r}})\right], \tag{17}$$

where the normalized vector spherical harmonic $\mathbf{X}_{lm}(\hat{\mathbf{r}})$ is defined in Appendix C and its counterpart $\mathbf{Y}_{lm}(\hat{\mathbf{r}})$ is defined as $\mathbf{Y}_{lm}(\hat{\mathbf{r}}) = \mathrm{i}\hat{\mathbf{r}}\times\mathbf{X}_{lm}(\hat{\mathbf{r}})$. This decomposition together with mutual orthogonality and normalization of the harmonics leads to the following calculation of the expansion coefficients:

$$\begin{aligned} p^{\text{e}}_{lm} &= -\frac{\mathrm{i}^l}{k^2}\int_0^{2\pi}\int_0^{\pi} \mathbf{F}(\theta,\phi)\mathbf{Y}^*_{lm}(\theta,\phi)\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi, \\ p^{\text{m}}_{lm} &= -\frac{\mathrm{i}^l}{k^2}\int_0^{2\pi}\int_0^{\pi} \mathbf{F}(\theta,\phi)\mathbf{X}^*_{lm}(\theta,\phi)\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi. \end{aligned} \tag{18}$$

To form an inverse problem of Eq. (13), one also needs the incident field coefficients. For the plane-wave illumination, those can be obtained analytically. To that end, we use the expressions from [34] with changes according to different definitions of VSWFs.

The code implementing the T-matrix calculation with ADDA is written in Python and hosted on GitLab [99]. The installation process is simplified by automatically setting up ADDA and designed to be cross-platform, having been tested on Windows and Linux environments, specifically under Python 3.11.

At its core, this implementation uses simple trapezoid integration for the numerical tasks, which, while straightforward, is effectively

enhanced with Numba to significantly boost computational speed. To solve the T-matrix, the code calculates the pseudoinverse of the matrix that represents the incident field coefficients.

Calculation of the T-matrix with the provided code is executed either using a simple Command-Line Interface (CLI) that mimics the standard ADDA CLI or by writing a custom Python script using the dedicated Python package. With the CLI, the approach is designed to be accessible for users familiar with ADDA's conventional interface. The main argument is, then, a string with a set of standard ADDA parameters. For example, here is a command executing the calculation of the T-matrix of a sphere with refractive index of 2, diameter 200 nm, at a wavelength of 500 nm:

```
./addatmatrix_cli.py tmatrix --adda_string "-
    shape sphere -m 2 0  -size 200 -lambda 500"
```

Listing 9: T-matrix generation using ADDA CLI

It is a good practice to also specify the number of illuminations K, size of the T-matrix, and number of scattering angles for quadratures in Eq. (18). However, some default values are present in the script.

Using the provided Python package simplifies scripting and integration into Python-based projects. For example, here is the same simulation as above:

```
from addatmatrix.data import write_t_matrix_data
from addatmatrix.t_matrix import t_matrix

output_path = "sphere.tmat.h5"
adda_string = "-shape sphere -m 2 0"
tmatrix_data, metadata, indices = t_matrix(
    adda_string,wavelength=500)
write_t_matrix_data(output_path, tmatrix_data,
    indices, metadata)
```

Listing 10: T-matrix generation using Python code

Looking towards future improvements, the code could benefit from FFT acceleration of far-field scattering calculation [100] and more advanced integration methods. Alternatively, the far-field scattering can be skipped altogether by direct computation of VSWF expansion coefficients from the dipole polarizations. The latter resembles the translation-addition of VSWFs, for which fast algorithms exist as well. Moreover, the incident illumination can also be changed to VSWFs - this will eliminate the calculation of pseudo-inverse, making the whole approach robust for particles larger than the wavelength. These enhancements would further solidify the code's utility and performance, making it an even more powerful tool for researchers and engineers working in nanophotonics and other fields.

4.6. MEEP

MEEP stands for "MIT Electromagnetic Equation Propagation" and solves Maxwell's equations via the finite difference time domain method (FDTD) [101]. Since the T-matrix connects fields in the frequency domain, the Fourier transform is part of the post-processing. A single simulation in time domain allows obtaining T-matrices at multiple frequencies. Similarly to the previous example, the complication arises from the fact that direct illumination with a VSWF is not an option in MEEP. Instead, plane wave illuminations are used, and a decomposition into spherical waves is performed afterward. Specifically, the plane waves in the following form are used:

$$\mathbf{E}_{\text{in}}(\mathbf{r},t) = \mathbf{E}_0 e^{i(\mathbf{k_0}\cdot\mathbf{r}-\omega t)} g(\mathbf{k_0}\cdot\mathbf{r}-\omega t), \tag{19}$$

with a suitable envelope function $g(\mathbf{k_0}\cdot\mathbf{r}-\omega t)$. The spectrum that is launched into the system is dictated by the temporal width of the envelope function. The total simulations have to cover all the angles of incidence and all the polarizations as fully as possible. The approach is to generate an equally distributed grid of points on a sphere following the "Fibonacci Sphere" algorithm and in this way define the incident angles, similarly to the approach described previously with ADDA. For each incident direction, fields with two orthogonal polarizations have to be defined. The intricate detail is the choice of sufficient illumination sources to retrieve the correct scattering response for any illumination just by multiplying the incident field with a computed T-matrix. A T-matrix of size $N \times N$ requires a minimum number of $N_{\text{min}} = 2(l_{\text{max}}+2)l_{\text{max}}$ equations. However, for a more stable solution, it is recommended to perform more simulations, typically $N_{\text{sim}} = 2N_{\text{min}}$. In the current implementation, the simulation is not performed with a rotated plane wave, but with a rotated object. Rotation of plane waves is not desired, since the oblique plane waves are not fully absorbed in perfectly matching layers, and the wavefront is thus slightly distorted. Before assembling the T-matrix, the fields are rotated back. After the transformation to the frequency domain is done, the scattering coefficients are calculated using the formula from [84]:

$$\{a,b\}_{lm} = -ik\int_{\Gamma} \mathbf{dS}\left[(\nabla\times\mathbf{E}_{\text{sc}})\times\{\mathbf{N}^{(1)*}_{lm},\mathbf{M}^{(1)*}_{lm}\} - k\{\mathbf{M}^{(1)*}_{lm},\mathbf{N}^{(1)*}_{lm}\}\times\mathbf{E}_{\text{sc}}\right], \tag{20}$$

here Γ is an arbitrary surface enclosing the scatterer. The surface of a cuboid in this case is beneficial, since the grid is rectangular.

In the following, the sequence of calls to different functions is shown in a few selected code snippets.

```
def sphere() -> None:
    c = get_test_config()
    t = calculate_T(c)
    h5save(c, t, "test_sphere", "Test single
    sphere file")
```

Listing 11: Snippet from main.py

The parameters are uploaded from a preset configuration. In the next code snippet, it can be seen that a few of them are set manually, while others are taking the default values.

```
def get_test_config() -> Config:
    c = Config(
        resolution=30,
        sim_amount_mult=2,
        l_max=2,
        material=1.15,
        params={"radius" : 0.4},
        shape="sphere",
        eps_embedding = 1.,
        cpu_cores_per_simulation=SIZE,
    )
    return c
```

Listing 12: Snippet from config_factory.py

Resolution is one of the typical FDTD parameters, and the minimum accepted resolution in the code is one tenth of the wavelength. However, to reach converged results, a very high resolution is often required. The parameter requiring some explanation is `sim_amount_mult`. This factor is multiplied by the minimum number of simulations to define the total number of simulations to perform. As discussed previously, 2 is the optimal value. The maximum multipole order is set considering the size of the object and the wavelength. Further, parallelization parameters depending on the available resources is defined.

With the parameters fully specified, the computation flow proceeds to the function which performs the actual calculations. The `calculate_T` function is called to start the (optionally) parallel execution of the simulations. Inside the inner function `core_calc_T`, the MEEP simulations are set up and run. The 6 monitors are placed at the locations where the scattering coefficients will be computed using

Eq. (20), enclosing the scatterer in a cube. The distance to the monitors must be sufficiently large, such that the transient high-frequency fields excited when the source turns off decay sufficiently at such distance. The runtime can be directly specified, however, by default the simulation continues until the fields at the monitors are not changing with some tolerance after the sources were turned off. The incident fields are derived from a simulation of field propagation without the scatterer. After this, for all the incident angles and polarizations, the actual permittivity grid of the scatterer is rotated by the corresponding angle and the simulation is performed. The resulting fields are rotated back and Fourier transformed. The incident and scattered fields are expanded in VSWFs. Eventually, to obtain the T-matrix, the inverse problem is solved using the least squares method.

```
    e_amp_inc = get_e_amp_from_e_in(c=c,
    source_data=source_data, sim_res=sim_res_inc
    )
    incident_matrix = []
    scatter_matrix = []
    rot_matrix = []
    for ii in sim_id_range:
        print(f"Simulation id = {ii}")
        rotation_data =
    get_persistent_sphere_angles(c=c, id=ii)
        rotated_epsgrid = rotate_eps_grid(
    eps_grid, rotation_data, c.eps_embedding)
        rot_array = np.array([rotation_data.
    theta, rotation_data.phi, rotation_data.
    alpha])
        rot_matrix.append(rot_array)
        sim_res_sca = do_scattered_sim(
            id=ii,
            c=c,
            source_data=source_data,
            eps_grid=rotated_epsgrid,
            sim_res_inc=sim_res_inc,
        )
        incident_coefs = get_inc_coefs(
            c=c, source_data=source_data,
    rotation_data=rotation_data, e_amp=e_amp_inc
        )
        scatter_coefs = get_sca_coefs(
            c=c, rotation_data=rotation_data,
    sim_res=sim_res_sca
        )
        incident_matrix.append(incident_coefs)
        scatter_matrix.append(scatter_coefs)
```

Listing 13: Snippet from the function `core_calc_T` in scat.py

As a final step, the function `h5save` stores the T-matrix and the required metadata in a HDF5 file. In MEEP units, the speed of light is set to 1, which is considered when filling the metadata.

4.7. SMARTIES

The original T-matrix method, devised by Waterman [1], introduced alongside a specific calculation scheme – the Extended Boundary Condition Method (EBCM). This technique has strong analytical roots, requiring no meshing of the particle's volume or surface and, instead, computes T-matrix elements via analytical formulas which reduce to Mie theory for spherical particles [102]. For axial-symmetric particles, the method is remarkably efficient, as the matrix elements are obtained via simple one-dimensional integrals along the generatrix. The EBCM method is particularly popular for simple geometrical shapes, where it typically provides the fastest and most accurate way to calculate a T-matrix [102].

SMARTIES [41] is a MATLAB implementation of the EBCM to simulate the optical properties of oblate and prolate spheroidal particles, with comparable speed and accuracy as Mie theory for spheres. SMARTIES uses an improved algorithm that overcomes some loss of precision faced by EBCM in the case of large and elongated particles, and can routinely achieve numerical accuracy better than 10 digits. Although restricted to spheroids, SMARTIES may be useful to researchers seeking a fast, accurate and reliable tool to simulate the near-field and far-field optical properties of nonspherical particles, and can also appeal to other developers of light-scattering software seeking a very accurate benchmark for nonspherical particles with a challenging aspect ratio and/or refractive index contrast.

We provide below an example script (Listing 14) to output the T-matrix of a gold spheroid, used in the calculation of Fig. 6. Further examples are available in the project's repository [103].

```
wavelength = (400:800)';
epsilon = epsAu(wavelength);
medium=1.33; % water

% spheroid semi-axes
% stParams.a=20; stParams.c=40;

 % simulation parameters
 stParams.N=3; stParams.nNbTheta=120;

 % additional options if required
 stOptions = {};

 % allocate 3D array for all results
 qmax = 2*(stParams.N*(stParams.N + 1) +
    stParams.N );
 tmatrix = zeros(length(wavelength), qmax, qmax)
    ;
 % loop over wavelengths
 for i=1:length(wavelength)
     stParams.k1=medium*2*pi/wavelength(i);
     stParams.s=sqrt(epsilon(i)) / medium;
     [stCoa, stT] = slvForT(stParams,stOptions);
     [T, u, up] = expand_tmat(stT, qmax); %
    tmatrix elements and indices
     ind = sub2ind(size(tmatrix),q*0+i, u, up);
     linear index in 3D array
     tmatrix(ind) =  T(:,7) + 1i*T(:,8);
 end
 analytical_zeros = true(qmax,qmax);
 analytical_zeros(sub2ind(size(analytical_zeros)
    , u, up)) = false;

 % data to export
 epsilon = struct(...
     'embedding', medium^2, 'particle', epsilon,
     'embedding_name', 'H2O, Water', ...
     'embedding_keywords', 'non-dispersive', '
    embedding_reference', 'constant', ...
     'material_name', 'Au, Gold', '
    material_reference', 'Au from Johnson and
    Christy 1972',...
     'material_keywords', 'dispersive, plasmonic
    ');

 geometry = struct('description',  'prolate
    spheroid', ...
     'shape', 'spheroid','radiusxy', stParams.a,
     'radiusz', stParams.c);

 computation = struct('description', '
    Computation using SMARTIES', ...
     'accuracy', 1e-10, 'Lmax', Lmax, 'Ntheta',
    Ntheta, ...
     'analytical_zeros', analytical_zeros);
```

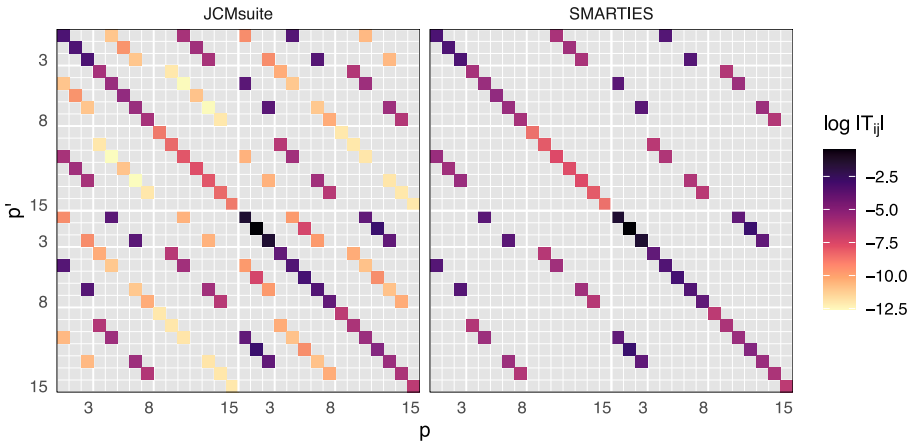

Fig. 4. Comparison of T-matrices for a prolate Au spheroid immersed in water, calculated with FEM (JCMsuite, left), and the EBCM (SMARTIES, right). Both computations use a wavelength of 630 nm, and the spheroid semi-axes are 20 nm and 40 nnm. The color scale presents the modulus of the T-matrix elements (in log scale for clarity). The same pattern is observed for the dominant terms, but the FEM introduces a few non-zero elements due to the mesh description of the particle shape. In this plot, the native SMARTIES convention for T-matrix as 2×2 blocks is used.

```
comments = struct('name', 'Au prolate spheroid
   in water',...
    'description', 'Computation using SMARTIES'
   ,...
    'keywords', 'gold, spheroid, ebcm', 'script
   ', [mfilename '.m']);

tmatrix_hdf5('smarties_example.tmat.h5',
   tmatrix, wavelength, epsilon,...
               geometry, computation, comments)
```

Listing 14: Example usage of SMARTIES

Internally, SMARTIES stores T-matrix elements in a custom structure, as the particle symmetries (axial-symmetric and plane of symmetry) result in many T-matrix elements being exactly zero, and therefore not calculated [41]. To convert to the `.tmat.h5` format, the function `expand_tmat()` expands the full matrix by filling the missing elements with zeros, and also reorders indices to match the conventions described herein, from their original ordering as a 2×2 block matrix (Fig. 4).

5. Validation

Before the final submission of the T-matrix file to the database, its compliance with the standard format developed and described in this document has to be proven. This includes technical aspects of the data representation and detecting physical inaccuracies pointing to faults in the computation method or discrepancies with the actual geometry setting. We stress that if the example files from the repository are used to generate the T-matrices, the requirements should be fulfilled, since they were inspected on individual basis. For the future database, the validation of the uploaded data will be performed using an automated script.

5.1. Formal validation

Several points to comply with the formatting guidelines are important:

- Nomenclature. The presence of all compulsory *groups*, *datasets*, and *attributes* is systematically checked. Commonly used versions of the naming for the inputs produce suggestions to the user for a suitable alternative in the correct naming convention.
- Shapes and types. There is a correspondence of the stored parameters between each other, the size of the array of the T-matrix, and the size of the array of the modes defining polarization and multipole orders being one example. The types of the entries are also inspected.

5.2. Normalization conventions

Different conventions regarding the vector spherical wave normalization are used in the literature. For the interchangeability of results calculated with different methods, it is important to adhere to one specific normalization, which we define in Appendix C. As a check, a reference structure can be calculated with a custom method, and its T-matrix compared with the provided reference output.

5.3. Physics constraints

T-matrices can manifest physical constraints imposed by the properties of the investigated object such as symmetries [9]. Combined with symmetries in lattices at which the scatterers are arranged, many physical effects can be controlled [104]. Adding checks on the symmetries should not be seen as an additional complication, since they can contribute largely to verifying the computation as physically correct.

Reciprocity

For the linear interaction of matter and light, one can derive the following expression (Eq. 36 in [9]):

$$T^{ij}_{l,m,l',m'} = (-1)^{m+m'} T^{ji}_{l',-m',l,-m} \tag{21}$$

for $l \in \mathbb{N}_0$, $m \in \mathbb{Z}$ with $l \geq |m|$, i and j the polarization indices, when the response of the object meets Lorentz reciprocity.

Rotational invariance

A rotationally symmetric object produces a scattering response that is also rotationally invariant. This sets constraints on the T-matrix of such an object, formulated in the following for the case of the symmetry axis aligned with the z-axis (Eqs. 30 and 31 in [9]):

$$T^{ij}_{l,m,l',m'} = \delta_{mm'} T^{ij}_{l,m,l',m'} , \tag{22}$$

$$T^{ij}_{l,m,l',m'} = ij T^{ij}_{l,-m,l',-m'} . \tag{23}$$

For a sphere, this condition implies a matrix with only diagonal entries in parity basis, which does not depend on azimuthal indices m, m'. Please note the difference in notation compared to the reference, as the polarization indices i, j take the values of -1, 1.

Mirror symmetries

The T-matrix of an object invariant with respect to mirror symmetry operations exhibits certain constraints. The specific formula depends on

the polarization and the plane of symmetry. As an example, for mirror reflection regarding the xy-plane, we can derive the formulas based on the properties of the spherical harmonics:

$$Y_{lm}(\pi-\theta,\phi) = (-1)^{l+m} Y_{lm}(\theta,\phi)\,. \tag{24}$$

For the parity basis, the following formula holds:

$$T^{ij}_{l,m,l',m'} = (-1)^{i+j+m+m'+l+l'}\, T^{ij}_{l,m,l',m'}\,, \tag{25}$$

In the helicity basis, the modified formula is:

$$T^{ij}_{l,m,l',m'} = (-1)^{m+m'+l+l'}\, T^{-i-j}_{l,m,l',m'}\,. \tag{26}$$

Lossless/non-absorptive

For a non-absorbing object, the condition for the corresponding T-matrix can be derived by applying the principle of conservation of energy (Eq. 45 in [9], Eq. 5.59 in [102]). The integral of the Poynting vector $\langle \mathbf{S}(\mathbf{r})\rangle \cdot \mathbf{r}$ over a spherical surface at infinity has to vanish (Eq. 5.55 in [102]). This leads to the unitarity of the S-matrix: $\mathbf{S}^\dagger\mathbf{S} = \mathbb{1}$. The corresponding T-matrix representation is then:

$$\mathbf{T}^\dagger\mathbf{T} = -\frac{1}{2}(\mathbf{T}^\dagger + \mathbf{T})\,, \tag{27}$$

where $(\mathbf{T}^\dagger)^{ij}_{l,m,l',m'} = T^{ji}_{l',m',l,m}$.

Passive

If loss is prevalent in the medium, the integral of the Poynting vector over a spherical surface at infinity is negative, as more energy comes in than goes out. From this inequality, it can be concluded that the following expression is Hermitian positive-semidefinite (Eq. 10 in [105]):

$$-2\mathbf{T}^\dagger\mathbf{T} - \mathbf{T}^\dagger - \mathbf{T}\,. \tag{28}$$

For the gain medium, the sign of the inequality is the opposite, thus the condition for positive-semidefineteness is violated.

Numerical accuracy metric

Since the T-matrix is obtained from the numerical solution of Maxwell's equations, the relations listed above are fulfilled with some degree of inaccuracy. It is not viable to demand these relations to hold exactly, but the discrepancy necessitates a single computable quantity. The following metric is introduced, however, this choice is not unique. For the metric to represent a relative deviation from the magnitude of the initial value, we take the sum of the squared absolute values of the difference between the transformed and initial matrix and divide it by the sum of the squared norm of all the initial and transformed matrix elements:

$$\sigma = \frac{1}{2}\frac{\sum_{l=1}^{l_{\max}}\sum_{l'=1}^{l_{\max}}\sum_{m=-l}^{l}\sum_{m'=-l'}^{l'}\sum_{i=\pm1}\sum_{j=\pm1}|T^{ij}_{l,m,l',m'} - (T')^{ij}_{l,m,l',m'}|^2}{\sum_{l=1}^{l_{\max}}\sum_{l'=1}^{l_{\max}}\sum_{m=-l}^{l}\sum_{m'=-l'}^{l'}\sum_{i=\pm1}\sum_{j=\pm1}|T^{ij}_{l,m,l',m'}|^2 + |(T')^{ij}_{l,m,l',m'}|^2}\,. \tag{29}$$

This can be regarded as normalization by the total interaction cross-section, which is proportional to the scattering cross-section of the object. It is straightforward to apply the metric for geometric transformations. In case of Eq. (27), the metric can be used with the transformed T-matrix reformulated as $\mathbf{T}' = 2\mathbf{T}^\dagger\mathbf{T} + \mathbf{T}^\dagger$. The Hermitian positive-semidefiniteness in Eq. (28) is not tested using the metric. The minimum tolerance in Cholesky decomposition which produces the eigenvalues with non-negative values gives insight into the degree to which the condition in Eq. (28) is fulfilled.

Convergence

Since the goal is to make previously computed results reusable, additional checks on reliability are recommended. A convergence check is particularly important since a coarse mesh would not correctly represent the symmetries of the object, however, the algorithm of creating the mesh might also contribute to the emerging asymmetries. It is recommended for the researcher to check the mesh separately for the presence of symmetries existing in the object. No direct checks on the mesh are incorporated by default.

An additional convergence study justifying the truncation at the chosen number of the multipole orders is highly valuable as well. For a single provided T-matrix, the average extinction cross-section can be calculated and compared with the same matrix truncated by one multipole order, while it is on the behalf of the researcher to decide whether a higher multipole order is needed. If the inclusion of the higher multipole order does not change the average extinction cross-section by more than one percent, this can be considered sufficient. One has, however, to keep in mind that for some more sensitive optical characteristics, relying on correct computation in the near-field region, the insufficiency of the number of multipole orders could be revealed.

The same approach is suggested for the other parameters. If data with different values of, e.g., resolution are calculated with admissible accuracy, it is still recommended to upload both cases and let the user decide if the accuracy is acceptable for their application. The provision of additional data ensuring convergence for all method parameters is not mandatory, and it is the responsibility of the researcher to ensure the trustworthiness of the data.

Examples of data usage

This part is intended to describe examples of how researchers in the community can benefit from using the existing `.tmat.h5` files. It highlights the features of some of the software developed and indicates how to load and use files in the proposed data format. This should serve as a practical demonstration on how to capitalize on the data format. We consider the retrieval of T-matrices of two different objects and computation of their optical quantities with two different multiscattering tools. The use of these programs is only exemplary, and we encourage the readers to assess whether it is possible to add similar functions to their software to simplify the exchange of T-matrices.

TiO_2 cylinder

We demonstrate a basic use case for the T-matrix data format. We have computed the T-matrix of a titanium dioxide cylinder, stored it in the suggested format, and finally we use a separate program to load the file and compute the ensemble-averaged scattering cross-section. The T-matrix is computed with JCMsuite as outlined in Section 4.1. For the scattering cross-section calculation, we use *treams*, an open-source T-matrix-based scattering program in Python [36]. A module to load and store files, which conforms to the format described here, is part of the program.

In this example, we consider a cylinder made from titanium dioxide. The cylinder has a radius of 250 nm and a height of 300 nm. We consider a non-dispersive relative permittivity of $\epsilon = 6.25$ in the frequency range between 240 THz and 400 THz. We compute the T-matrix up to $l_{\max} = 4$ using JCMsuite as outlined above. The main modification is the change of the layout file for the geometry of a cylinder. The full files can be found in Listing 17. The created T-matrix file can then be loaded in *treams* by `treams.io.load_hdf5` as shown in the snippet below. The stored T-matrix is three-dimensional with the outer dimension corresponding to the number of frequencies.

```
cyl = treams.io.load_hdf5("cylinder_tio2.tmat.h5")

positions = [[-300, 0, 0], [300, 0, 0]]
cyl_cluster = [
    treams.TMatrix.cluster([tm, tm], positions)
    for tm in cyl
]
cyl_cluster = [
    tm.interaction.solve().expand(
        treams.SphericalWaveBasis.default(4)
    )
    for tm in cyl_cluster
```

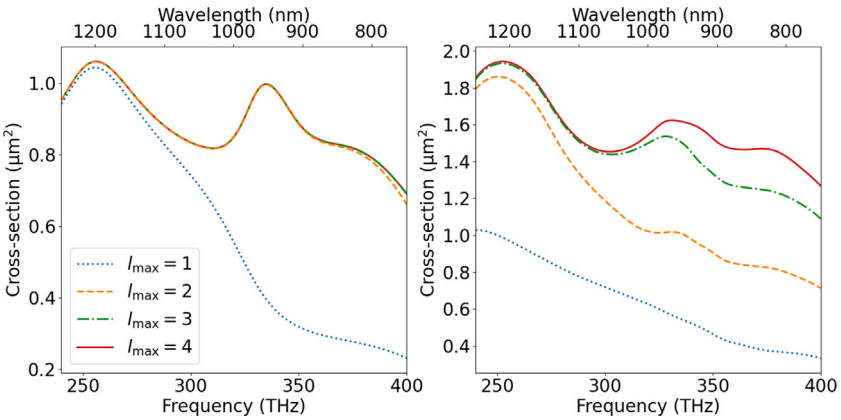

Fig. 5. Average extinction cross-section of a single titanium dioxide cylinder (left) with radius 250 nm and height 300 nm and two of those cylinders separated horizontally by 600 nm (right).

```
]

with h5py.File("two_cylinders.tmat.h5", "w") as
    fobj:
    treams.io.save_hdf5(fobj, cyl_cluster)
```

Listing 15: Data format usage in *treams*

The T-matrix can then be directly used in *treams*. In the example, we calculate the coupling between two disks separated horizontally by 600 nm. The `cluster` method stacks the matrices together, and the `solve` method calculates the interaction between objects and outputs a T-matrix in the local coordinate systems of each object. This is then expanded at a new single origin to obtain a global T-matrix. Without going into further details of the functions used to calculate the interaction, we can store the final result equally easily by `treams.io.store_hdf5`. The resulting average extinction cross-sections for the individual cylinder and the two cylinders are shown in Fig. 5.

The main demonstrated feature is that the loading function automatically reads all the files and returns them as a list of T-matrices. The order of the entries with the multipole description and polarization, the distinction of parity and helicity basis, and the frequency are recognized and stored in the T-matrix class of *treams*. Then, it can be conveniently used within *treams*. The storing function converts the class objects for the T-matrix to the correct entries in the data file. Further information can then be added afterward, e.g., the description. The original data file is also available at [81].

Gold spheroid

In this example, we are interested in the T-matrix of a gold spheroid in water and consider a different multiscattering tool to exploit it. TERMS [39,106] is a Fortran program based on the superposition T-matrix method, designed to simulate the near-field and far-field optical properties of collections of particles. It was developed primarily to model relatively compact clusters of resonant scatterers, such as plasmonic particles, often requiring large multipolar orders [60]. TERMS implements several independent algorithms, with complementary strengths and weaknesses, to describe the self-consistent electromagnetic interaction between multiple scatterers and compute far-field optical properties such as absorption, scattering, extinction, circular dichroism, as well as near-field intensities and the local 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 such as far-field cross-sections and near-field quantities, greatly reducing the computational time needed to simulate particles and systems of particles in random orientation [17]. Each scatterer is described by a T-matrix, which is computed internally for spherical particles (including layered spheres), or using external files computed with any other method.

The program's documentation and website [106] offer many examples of use; here we only illustrate the import of an external T-matrix in the `tmat.h5` format. The input file for the simulation reproduced below considers two gold spheroids in water, separated by 100 nm and rotated by 45 degrees to form a chiral structure.

```
ModeAndScheme 2 3
MultipoleCutoff 3 3 -3
Wavelength 400 800 200
Medium 1.7689
OutputFormat HDF5 smarties_dimer
TmatrixFiles 1
smarties.tmat.h5
Scatterers 2
TF1 0 -50 0 40 0 0 0 2
TF1 0  50 0 40 0 0.7853982 0 2
```

Listing 16: Script for TERMS

The simulation is run with the command `terms input`, and outputs cross-sections in the file `results.h5`. The results are displayed in Fig. 6. For comparison, the same simulation was run with a T-matrix produced by SMARTIES (Listing 14) for the same geometry.

Since Fortran is a low-level language, it is not very practical to implement a full support of all the options of the `tmat.h5` format. Instead, TERMS has currently implemented a basic import functionality with following expectations. The `vacuum_wavelength` is the only allowed field, and must be provided in nanometers. The wavelengths must match exactly the TERMS input file. Furthermore, no check is performed for the relative permittivity of the embedding medium, which should match the TERMS input file.

Hopefully, a Python interface to TERMS will be available in the future, which will add flexibility in the import of external T-matrices.

Summarizing statements

6. Summary

This manuscript establishes a unified data format for storing and distributing T-matrices, which represents a pivotal step in handling scattering data within the scientific community. T-matrices, in general, contain the complete information of the scattering properties of a scatterer in linear approximation. They tell us how light, or, more generally, electromagnetic waves, interact with a given object. However, they can also be used to express the properties of more advanced photonic materials made from many scatterers. Adopting

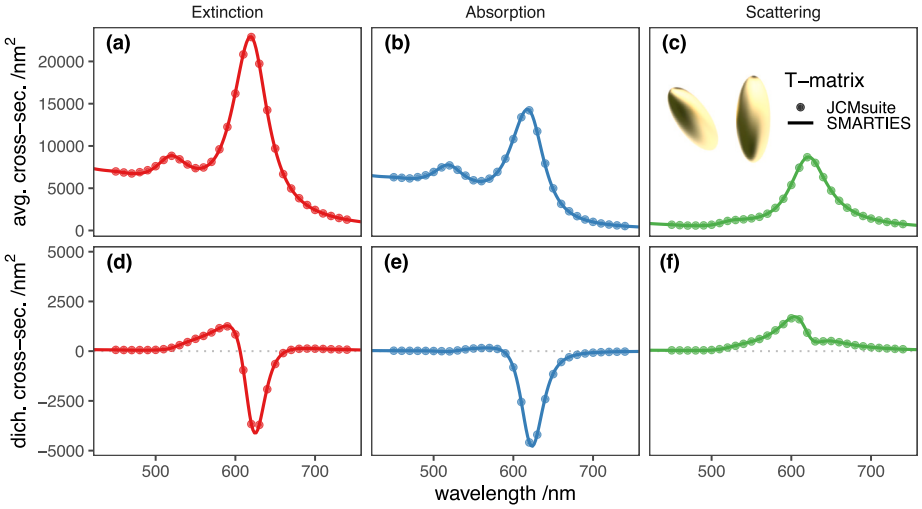

Fig. 6. Validation of results obtained using a T-matrix computed with JCMsuite (points), and SMARTIES (solid lines), both imported into TERMS for a multiple scattering calculation (Listing 16). The structure consists of a dimer of identical Au spheroids immersed in water, with center-to-center separation 100 nm, and a dihedral angle of $\pi/4$ making the dimer chiral. The top panels (a–c) present the orientation-averaged optical cross-sections for extinction, absorption, and scattering, respectively. The bottom panels (d–f) present the corresponding orientation-averaged circular dichroism cross-sections. Excellent agreement is obtained between the two methods.

this standardized format is a critical step towards addressing several prevalent challenges faced by researchers, including the repetitive computation of T-matrices across many laboratories worldwide. These computations frequently require considerable computational resources and pose significant financial and environmental burdens.

By providing a common framework for the systematic storage, accessibility, and sharing of T-matrices, we enhance the ability to reuse previously calculated data, thereby significantly reducing the need for duplicate computations. This approach conserves valuable resources and lessens the ecological impact of high-performance computing activities, which is increasingly important in light of current global energy concerns. Furthermore, the standardization of T-matrix data ensures that the scientific results are reproducible and verifiable, which is essential for maintaining the integrity and reliability of photonic research.

To permit wide use of this data format, we have described in this contribution multiple approaches to computing T-matrices and storing them in the required data format. In addition to the description provided here, the supporting files that can be used by others to compute T-matrices and store them in the expected data format are potentially of more practical use. These files are publicly available at [81].

We wish to motivate further scientists to adapt their tools so that T-matrices can be generated and stored in the desired format. The more tools are verified for this purpose, the larger the number of scientists that can benefit from the standardized manner of storing and archiving T-matrices. Moreover, we also demonstrated how to use T-matrices in a set of programs besides generating T-matrices. Along the same lines, we wish to motivate many more scientists to seek ways to integrate these T-matrices into their computational workflows.

The proposed data format is designed to be flexible yet precise enough to accommodate T-matrices of a wide range of different scattering structures, reflecting the needs of various spectral domains. This flexibility ensures broad applicability across multiple disciplines, including optics and photonics, nanotechnology, environmental sciences, microwave scattering, and biotechnology, among others. As such, researchers can now access a wealth of T-matrix data that may previously have been recalculated redundantly, accelerating the pace of innovation in fields reliant on scattering data.

Moreover, the uniform data format facilitates the integration of T-matrices with emerging technologies and methodologies, such as machine learning and big data analytics, which are becoming increasingly prevalent in scientific research. By enabling the efficient use of T-matrices in training machine learning-based technology to solve direct and inverse scattering problems, this format could potentially prevent the need for direct numerical simulations in some cases, streamlining the research process significantly [107–110]. This increasingly large number of simulations conducted solely with the purpose of generating training data, which is frequently computationally expensive, would no longer be necessary if systematically generated data from many labs worldwide could be used for training purposes.

The impact of this standardization extends beyond simplification and cost reduction in computational processes. It also promotes a collaborative scientific environment where researchers across the globe can contribute to and benefit from a future shared repository of T-matrices. This collaborative approach fosters innovation and enhances the educational value of scattering data, allowing for more comprehensive and advanced training of future scientists. To this end, it significantly supports research and development in optical nanometrology and advanced applications, ranging from nanotechnologies and advanced (nano-) materials to novel photonic devices.

7. Outlook

This establishment of a standardized data format for T-matrices represents only the initial phase of a longer initiative aimed at enhancing the accessibility and utility of scattering data within the scientific community. The next crucial step in this endeavor is the development of an online database that facilitates not only the uploading but also the sharing of T-matrix data. This proposed database will be designed with advanced search capabilities, ideally leveraging large language models, allowing users to efficiently retrieve T-matrices that closely resemble a specific object or set of scattering properties. Additionally, it will enable users to input a desired T-matrix and search for corresponding entries in the database that match the input at a predefined operational frequency. This functionality is particularly valuable as it effectively addresses the inverse problem, offering a powerful tool for researchers seeking specific scattering responses without the need for direct computation. Such a resource will significantly expedite the process of scientific discovery and innovation, providing a robust platform for researchers worldwide to collaborate and advance the field of photonics and related disciplines.

Establishing an online and open access T-matrix database raises several research questions and opportunities, particularly in comparing

the accuracy and efficiency of various computational methods used to generate T-matrices. Researchers could systematically evaluate the performance of different scattering computation techniques, such as the finite element method, boundary element method, finite difference time domain method, or discrete dipole approximation, by comparing their output T-matrices stored within the database. This analysis would reveal discrepancies, validate the methods across various materials and geometries, and help refine these computational techniques for enhanced accuracy and efficiency.

Furthermore, the database opens the door to address interesting research questions. By mining the database for unique T-matrix patterns and associating them with physical structures, researchers could identify scatterers with predefined properties. These explorations could lead to the discovery of new materials with applications in advanced photonic technologies like lighting devices, enhanced sensing, or light management in solar cells.

Finally, this database democratizes access to high-level photonic research, allowing researchers without the computational expertise or resources for the computation of T-matrices to participate actively in the field. The database adds diversity to the research community by providing pre-computed T-matrices and enhances collaborative opportunities across different domains. This inclusive approach could lead to new perspectives and innovative uses of photonic systems.

8. Conclusions

In conclusion, establishing a standard data format for T-matrices is a significant step in our scientific community's efforts to optimize research efficiency and collaboration. It addresses economic and ecological issues associated with repetitive computations and paves the way for novel research opportunities in photonics and related fields. The scientific community can look forward to more sustainable, reproducible, and innovative research outcomes by adhering to this standardized approach. The future database, where the datasets can be shared, in contrast to a "data lake" with unstructured data, serves as an ideal platform for finding use cases and datasets for training and testing out machine learning methods in the field of photonics. This is an area which is up to now underserved in the classical data repositories for machine learning such as Kaggle, opening a unique opportunity to establish a go-to place for popularizing otherwise niche research data. The ongoing development and widespread adoption of this data format will undoubtedly catalyze further advancements in studying and applying complex photonic systems, with broad implications across multiple scientific disciplines.

CRediT authorship contribution statement

Nigar Asadova: Writing – review & editing, Writing – original draft, Validation, Software, Conceptualization, Investigation, Methodology, Visualization. **Karim Achouri:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Kristian Arjas:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Baptiste Auguié:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Roland Aydin:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Alexandre Baron:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Dominik Beutel:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Bernd Bodermann:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Kaoutar Boussaoud:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Sven Burger:** Writing – review & editing, Validation, Software, Methodology, Investigation, Conceptualization. **Minseok Choi:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Krzysztof M. Czajkowski:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization. **Andrey B. Evlyukhin:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Atefeh Fazel-Najafabadi:** Writing – review & editing, Validation, Software, Methodology, Investigation, Conceptualization. **Ivan Fernandez-Corbaton:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Puneet Garg:** Writing – review & editing, Methodology, Investigation, Conceptualization. **David Globosits:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Ulrich Hohenester:** Writing – review & editing, Validation, Software, Investigation, Conceptualization, Methodology. **Hongyoon Kim:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Seokwoo Kim:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Philippe Lalanne:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Eric C. Le Ru:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Jörg Meyer:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Jungho Mun:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Lorenzo Pattelli:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization. **Lukas Pflug:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Carsten Rockstuhl:** Writing – review & editing, Writing – original draft, Supervision, Investigation, Conceptualization, Methodology, Project administration. **Junsuk Rho:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Stefan Rotter:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Brian Stout:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Päivi Törmä:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Jorge Olmos Trigo:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Frank Tristram:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Nikolaos L. Tsitsas:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Renaud Vallée:** Writing – review & editing, Validation, Software, Methodology, Investigation, Conceptualization. **Kevin Vynck:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Thomas Weiss:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Peter Wiecha:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Thomas Wriedt:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Vassilios Yannopapas:** Writing – review & editing, Methodology, Investigation, Conceptualization. **Maxim A. Yurkin:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Conceptualization. **Grigorios P. Zouros:** Writing – review & editing, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: M.Y. is currently a handling associate editor of the Journal of Quantitative Spectroscopy and Radiative Transfer. T.W. is currently a handling associate editor of the Journal of Quantitative Spectroscopy and Radiative Transfer. S.B. is also affiliated with JCMwave GmbH which is developing and commercializing the JCMsuite software package. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

N.A., K.B., J.M., and C.R. acknowledge support by the Federal Ministry of Education and Research (BMBF) within the project DAPHONA (16DKWN039). K. Achouri acknowledges funding from the Swiss National Science Foundation (Grant No. TMSGI2_218392). B.A. and E.C.L.R. thank the Royal Society of New Zealand Te Apārangi for

support through Marsden grants MFP-VUW2204 and MFP-VUW2118. D.B., F.T., and C.R. acknowledge support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy via the Excellence Cluster 3D Matter Made to Order (Grant No. EXC - 2082/1 - 390761711) and from the Carl Zeiss Foundation via CZF-Focus@HEiKA. K.M.C. acknowledges support by the Polish National Science Center via the project 2020/37/N/ST3/03334. S.B. acknowledges support by BMBF (Forschungscampus MODAL, project number 05M20ZBM) and by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689). J.M. acknowledges the Presidential Sejong fellowship (RS-2023-00252778) funded by the Ministry of Science and ICT (MSIT) of the Korean government. H.K. acknowledges the Asan Biomedical Science fellowship, and the Presidential Science fellowship funded by the MSIT of the Korean government. J.R. acknowledges the POSCO-POSTECH-RIST Convergence Research Center program funded by POSCO, and the NRF grant (RS-2024-00356928) funded by the MSIT of the Korean government. A.B. acknowledges support from the French National Research Agency (ANR) in the frame of the project MELODIE (Grant ANR-22-CE09-0027) and by the Institut Universitaire de France (IUF). D.G. and S.R. acknowledge support by the Austrian Science Fund (FWF) under project P32300 (WAVELAND). M.Y. acknowledges support of the Normandy Region (project RADDAERO). L.P. acknowledges support by the European project 21GRD03 PaRaMetriC. The project 21GRD03 PaRaMetriC received funding from the European Partnership on Metrology, co-financed by the European Union's Horizon Europe Research and Innovation Programme and from the Participating States. K. Arjas and P.T. acknowledge support from the Academy of Finland under Project No. 349313 and from the Vilho, Yrjö and Kalle Väisälä Foundation. A.B.E. acknowledges the support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453). B.B. acknowledges support by the European project 20FUN02 "POLight". The work has been partly addressed also in the project 20FUN02 "POLight". The project 20FUN02 "POLight" has received funding from the EMPIR programme co-financed by the Participating States and from the European Union's Horizon 2020 research and innovation programme. L.P. and K.C. acknowledge fruitful collaboration with the main developer of SMUTHI and CELES, Amos Egel. L.P. acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)-Project-ID 416229255-SFB 1411. R.V. acknowledges Guillaume Demésy (Institut Fresnel, Marseille) for interactions on the use of the free ONELAB FEM software to calculate the T-matrix of peculiar structures. We thank Pascal Scherer that implemented the codes to use MEEP in the calculation of the T-matrix.

Appendix A. Constitutive relations

We consider causal materials invariant under translations of time, such that a description by time-harmonic fields with a fixed frequency can be used (please be reminded that we use the $\exp(-i\omega t)$ convention here). Thus, in principle, all quantities in this chapter are defined for dispersive media. However, the frequency argument will be omitted here. The most general case of constitutive relations for linear homogeneous materials are

$$\begin{pmatrix} \frac{1}{\epsilon_0}\mathbf{D} \\ c_0\mathbf{B} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\epsilon} & \boldsymbol{\xi} \\ \boldsymbol{\zeta} & \boldsymbol{\mu} \end{pmatrix} \begin{pmatrix} \mathbf{E} \\ Z_0\mathbf{H} \end{pmatrix} \tag{A.1}$$

with dimensionless 3-by-3 tensors $\boldsymbol{\epsilon}$, $\boldsymbol{\mu}$, $\boldsymbol{\xi}$, and $\boldsymbol{\zeta}$. We refer to the full 6-by-6 tensor as bianisotropic tensor. The quantities ϵ_0, c_0, and Z_0 are the vacuum values of the permittivity, speed of light, and the wave impedance, respectively. These prefactors are chosen to normalize all fields to the same unit, namely $\mathrm{V\,m^{-1}}$. Thus, the bianisotropic tensor contains dimensionless units. If all four 3-by-3 tensors are proportional to the unit matrix, then the material is called biisotropic. Then, the material parameters can be expressed as scalars ϵ, μ, ξ, and ζ. The magnetoelectric couplings can be expressed alternatively with the non-reciprocity parameter $\chi = \frac{\xi+\zeta}{2}$ and the chirality parameter $\kappa = \frac{\xi-\zeta}{2\mathrm{i}}$ as

$$\begin{pmatrix} \frac{1}{\epsilon_0}\mathbf{D} \\ c_0\mathbf{B} \end{pmatrix} = \begin{pmatrix} \epsilon & \xi \\ \zeta & \mu \end{pmatrix} \begin{pmatrix} \mathbf{E} \\ Z_0\mathbf{H} \end{pmatrix} \tag{A.2}$$

$$= \begin{pmatrix} \epsilon & \chi + \mathrm{i}\kappa \\ \chi - \mathrm{i}\kappa & \mu \end{pmatrix} \begin{pmatrix} \mathbf{E} \\ Z_0\mathbf{H} \end{pmatrix}. \tag{A.3}$$

If the non-reciprocity parameter vanishes ($\chi = 0$), then the material is referred to as chiral.

Starting from the bianisotropic case, but requiring vanishing magnetoelectric coupling, i.e., $\boldsymbol{\xi} = 0 = \boldsymbol{\zeta}$, instead of isotropy, leads to the constitutive relations

$$\begin{pmatrix} \frac{1}{\epsilon_0}\mathbf{D} \\ c_0\mathbf{B} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\epsilon} & 0 \\ 0 & \boldsymbol{\mu} \end{pmatrix} \begin{pmatrix} \mathbf{E} \\ Z_0\mathbf{H} \end{pmatrix} \tag{A.4}$$

of an anisotropic material. Finally, if the material is isotropic and has no magnetoelectric coupling, then we have an isotropic material with

$$\begin{pmatrix} \frac{1}{\epsilon_0}\mathbf{D} \\ c_0\mathbf{B} \end{pmatrix} = \begin{pmatrix} \epsilon & 0 \\ 0 & \mu \end{pmatrix} \begin{pmatrix} \mathbf{E} \\ Z_0\mathbf{H} \end{pmatrix} \tag{A.5}$$

as constitutive relations. In a general case, the material can be treated as nonlocal, which implies that the electric field at a point $\mathbf{r}$ is influenced not only by the electric field at that point, but also at all other points $\mathbf{r}'$ within a spatial domain surrounding $\mathbf{r}$. The following relation holds:

$$\mathbf{D}(\mathbf{r}) = \int \mathbf{R}(\mathbf{r}-\mathbf{r}')\mathbf{E}(\mathbf{r}')\mathrm{d}\mathbf{r}', \tag{A.6}$$

where $\mathbf{R}(\mathbf{r}-\mathbf{r}')$ is the nonlocal response kernel. In [111], the nonlocal response equation is derived in the following form:

$$\left(\frac{\beta^2}{\omega^2 + \mathrm{i}\gamma\omega} - \mathrm{i}\frac{D}{\omega}\right)\nabla\cdot(\nabla\cdot\mathbf{J}(\mathbf{r})) + \mathbf{J}(\mathbf{r}) = \sigma\mathbf{E}(\mathbf{r}) \tag{A.7}$$

where

$$\sigma = \mathrm{i}\epsilon_0\frac{\omega_\mathrm{p}^2}{\omega + \mathrm{i}\gamma} \tag{A.8}$$

is the Drude conductivity,

$$\omega_\mathrm{p}^2 = \frac{n_0 e^2}{\epsilon_0 m} \tag{A.9}$$

the plasma frequency of the metal, γ is the Drude damping rate, n_0 the equilibrium electron density, $\beta^2 = (3/5)v_\mathrm{F}^2$, here v_F is the Fermi velocity, D is the diffusion constant, and e and m are the electron charge and mass, respectively.

Appendix B. Coordinate systems

The Cartesian, cylindrical, and spherical coordinates are related by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \rho\cos\phi \\ \rho\sin\phi \\ z \end{pmatrix} = \begin{pmatrix} r\sin\theta\cos\phi \\ r\sin\theta\sin\phi \\ r\cos\theta \end{pmatrix} \tag{B.1}$$

and have the associated unit vectors

$$\begin{pmatrix} \hat{\boldsymbol{r}} \\ \hat{\boldsymbol{\theta}} \\ \hat{\boldsymbol{\phi}} \end{pmatrix} = \begin{pmatrix} \sin\theta & 0 & \cos\theta \\ \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\rho}} \\ \hat{\boldsymbol{\phi}} \\ \hat{\mathbf{z}} \end{pmatrix} \tag{B.2}$$

$$= \begin{pmatrix} \sin\theta\cos\phi & \sin\theta\sin\phi & \cos\theta \\ \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ -\sin\phi & \cos\phi & 0 \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \\ \hat{\mathbf{z}} \end{pmatrix}. \tag{B.3}$$

By default, the z-axis has a special role (symmetry axis of an axisymmetric object). If this is changed to either the x or y-axis, $(x, y, z)^T$ in Eq. (B.1) is replaced by $(y, z, x)^T$ or $(z, x, y)^T$, respectively. Eq. (B.3) is changed accordingly to $(\hat{\mathbf{y}}, \hat{\mathbf{z}}, \hat{\mathbf{x}})^T$ or $(\hat{\mathbf{z}}, \hat{\mathbf{x}}, \hat{\mathbf{y}})^T$.

Appendix C. Mode normalization

Here, we comprehensively define the normalization of the modes starting from elementary functions, which coincides with the normalization in [63]. This is necessary to have an unambiguous definition of the vector spherical waves we use. We also define a reference T-matrix of an object which can be used to verify the normalization. Please note, we work here with complex-valued VSWF. If real-valued expressions are needed, dedicated conversion tools can easily be set up.

C.1. Definition

We start with the definition of the associated Legendre polynomials (which are, in general, no polynomials) by

$$P_l^m(x) = \frac{(-1)^m}{2^l l!}(1-x^2)^{\frac{m}{2}}\frac{\mathrm{d}^{l+m}}{\mathrm{d}x^{l+m}}(x^2-1)^l \tag{C.1}$$

for $l \in \mathbb{N}_0$ and $m \in \mathbb{Z}$ with $l \geq |m|$. Please note especially the factor $(-1)^m$ in front. With the associated Legendre polynomials as above, we define the spherical harmonics by

$$Y_{lm}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)\mathrm{e}^{\mathrm{i}m\phi}\,. \tag{C.2}$$

We define the vector spherical waves by:

$$\mathbf{M}_{lm}^{(n)}(kr,\theta,\phi) = z_l^{(n)}(kr)\mathbf{X}_{l,m}(\theta,\phi)\,. \tag{C.3}$$

Here, $\mathbf{X}_{lm}$ is the normalized vector spherical harmonics:

$$\mathbf{L} = \frac{\hat{\mathbf{r}}\times\nabla}{\mathrm{i}} \tag{C.4}$$

$$\mathbf{X}_{l,m}(\theta,\phi) = \frac{1}{\sqrt{l(l+1)}}\mathbf{L}Y_{l,m}(\theta,\phi) \tag{C.5}$$

The superscript $n = 1$ refers to the incident modes and $n = 3$ is used for the scattered modes. These are the modes that matter in the context of the T-matrix definition. Thus, the functions $z_l^{(1)}(x) = j_l(x)$ are the spherical Bessel functions, and $z_l^{(3)}(x) = h_l^{(1)}(x)$ are the spherical Hankel functions of the first kind. For completeness, the functions $z_l^{(2)}(x) = N_l(x)$ are the spherical Neumann functions, and $z_l^{(4)}(x) = h_l^{(2)}(x)$ are the spherical Hankel functions of the second kind.

For the time evolution, we use $\exp(-\mathrm{i}\omega t)$ such that the spherical Hankel functions of the first kind are traveling outwards. k is the wavenumber in the medium. After applying the operator to the spherical harmonics, the vector spherical waves $\mathbf{M}_{lm}^{(n)}(kr,\theta,\phi)$ can be expressed as

$$\begin{aligned}\mathbf{M}_{lm}^{(n)}(kr,\theta,\phi) = z_l^{(n)}\Bigg[&\frac{1}{2}\left(\lambda_+ Y_{l,m+1}(\theta,\phi) + \lambda_- Y_{l,m-1}(\theta,\phi)\right)\hat{\mathbf{x}}\\ &+\frac{1}{2\mathrm{i}}\left(\lambda_+ Y_{l,m+1}(\theta,\phi) - \lambda_- Y_{l,m-1}(\theta,\phi)\right)\hat{\mathbf{y}}\\ &+mY_{lm}(\theta,\phi)\hat{\mathbf{z}}\Bigg]\end{aligned} \tag{C.6}$$

with $\lambda_\pm = \sqrt{(l\mp m)(l\pm m+1)}$ or equivalently as

$$\begin{aligned}&\mathbf{M}_{lm}^{(n)}(kr,\theta,\phi)\\ &= \mathrm{i}\sqrt{\frac{(2l+1)}{4\pi l(l+1)}\frac{(l-m)!}{(l+m)!}}\left[\mathrm{i}\frac{mP_l^m(\cos\theta)}{\sin\theta}\hat{\boldsymbol{\theta}} - \frac{\partial}{\partial\theta}P_l^m(\cos\theta)\hat{\boldsymbol{\phi}}\right]\mathrm{e}^{\mathrm{i}m\phi}z_l^{(n)}(kr)\,.\end{aligned} \tag{C.7}$$

This mode used for the electric field is called "TE" (transverse electric) mode or magnetic multipole. An orthogonal mode to this one can be defined by $\mathbf{N}_{lm}^{(n)}(kr,\theta,\phi) = \frac{\nabla}{k}\times\mathbf{M}_{lm}^{(n)}(kr,\theta,\phi)$ which results in

$$\begin{aligned}&\mathbf{N}_{lm}^{(n)}(kr,\theta,\phi)\\ &= \mathrm{i}\sqrt{\frac{(2l+1)}{4\pi l(l+1)}\frac{(l-m)!}{(l+m)!}}\left[\left(\frac{\partial}{\partial\theta}P_l^m(\cos\theta)\hat{\boldsymbol{\theta}} + \mathrm{i}\frac{mP_l^m(\cos\theta)}{\sin\theta}\hat{\boldsymbol{\phi}}\right)\frac{1}{k}\frac{\partial}{\partial r}(krz_l^{(n)}(kr))\right.\\ &\left. + l(l+1)P_l^m(\cos\theta)\hat{\mathbf{r}}z_l^{(n)}(kr)\right]\frac{\mathrm{e}^{\mathrm{i}m\phi}}{kr}\,,\end{aligned} \tag{C.8}$$

which is used for the mode named "TM" (transverse magnetic) or electric multipole.

Finally, we define the modes for positive and negative helicity by

$$\mathbf{A}_{lm\pm}^{(n)}(k_\pm r,\theta,\phi) = \frac{\mathbf{N}_{lm}^{(n)}(k_\pm r,\theta,\phi) \pm \mathbf{M}_{lm}^{(n)}(k_\pm r,\theta,\phi)}{\sqrt{2}}\,. \tag{C.9}$$

For biisotropic materials, only modes with well-defined helicity are solutions for Maxwell's equations with the constitutive relations from Eq. (A.3). The wavenumber

$$k_\pm = k_0\left(\sqrt{\epsilon\mu-\chi^2}\pm\kappa\right) \tag{C.10}$$

becomes then polarization dependent.

C.2. Reference T-matrix

We define a reference object as a verification aid for the T-matrix normalization. It consists of differently-sized spheres made from homogeneous materials. It is chosen such that it does not have vanishing entries in the T-matrix, so all spatial symmetries are broken. The symmetry breaking is achieved by using differently sized spheres at the corners of a tetrahedron. All spheres are made of a material with relative permittivity $\epsilon = 9$. The embedding medium is vacuum, and the wavelength is in the range from 300 to 700 nm. The spheres' radii and positions are:

- $r_1 = 50\,\mathrm{nm}$ and $\mathbf{a}_1 = a\left(-\frac{1}{2}, -\frac{\sqrt{3}}{6}, -\frac{\sqrt{6}}{12}\right)$
- $r_2 = 60\,\mathrm{nm}$ and $\mathbf{a}_2 = a\left(+\frac{1}{2}, -\frac{\sqrt{3}}{6}, -\frac{\sqrt{6}}{12}\right)$
- $r_3 = 70\,\mathrm{nm}$ and $\mathbf{a}_3 = a\left(0, \frac{\sqrt{3}}{3}, -\frac{\sqrt{6}}{12}\right)$
- $r_4 = 80\,\mathrm{nm}$ and $\mathbf{a}_4 = a\left(0, 0, \frac{\sqrt{6}}{4}\right)$

with the side length $a = 300\,\mathrm{nm}$ of the tetrahedron. These values were selected arbitrarily.

The T-matrix of this structure was calculated semi-analytically using *treams* software. Multipole order up to 6 was considered. At 500 nm, the average extinction cross-section is $0.214\,177\,9\,\mu\mathrm{m}^2$. The first 6×6 entries of the global T-matrix in parity basis are presented in Fig. C.7.

The provided HDF5 file contains all the necessary information and can be accessed via [81].

Appendix D. HDF5 file format and conversion tools

First, this section gives a very brief introduction to the different components of the HDF5 file format and serves the purpose of explaining why it is a valuable framework for storing the considered data. We also emphasize particular aspects that need to be considered in setting up the data format. Additionally, it defines the general idea of how different pieces of data are related. Next, some particular options for conversion of files from other formats are specified.

D.1. HDF5 file format

HDF5 is a hierarchic file format. Its main components are *groups*, which work similarly to directories, and *datasets*, which contain the actual data similar to files. However, more information about the data type is provided compared to regular files. These *datasets* can be arrays of an arbitrary number of dimensions. It is possible to specify a dataset to have certain initial dimensions, and certain maximum size of the dataset, thus keeping an option to add more data to the dataset. Each item can have *attributes* associated, which are small pieces of data directly attached to a *group* or *dataset*. Usually, they are used to providing metadata. Different *groups* or *datasets* can be linked with *softlinks* or *hardlinks*.

		l	1					
		m	-1		0		1	
		pol	electric	magnetic	electric	magnetic	electric	magnetic
l	m	pol						
1	-1	electric	-0.418545 +0.083899i	0.031467 -0.024599i	0.016282 -0.002888i	-0.017304 +0.007180i	0.005978 +0.001293i	-0.000114 -0.008017i
		magnetic	-0.031400 +0.024139i	-0.297966 +0.358709i	-0.001788 +0.008202i	-0.008336 -0.011040i	-0.000114 -0.008017i	0.029737 +0.008557i
	0	electric	-0.013604 -0.007450i	-0.000721 +0.007129i	-0.161395 +0.297545i	-0.000048 +0.000413i	-0.016282 +0.002888i	0.001788 -0.008202i
		magnetic	-0.012473 -0.006370i	-0.0004392 +0.018173i	-0.000048 +0.000413i	-0.335920 +0.245274i	0.017304 -0.007180i	0.008336 +0.011040i
	1	electric	0.014327 -0.005093i	0.006451 +0.009110i	0.013604 -0.007450i	0.012473 -0.006370i	-0.418545 +0.083899i	-0.031400 +0.024139i
		magnetic	0.006451 +0.009110i	0.016684 -0.015666i	0.007721 -0.007129i	0.004392 -0.018173i	0.031467 -0.024599i	-0.297966 +0.358709i

Fig. C.7. First 6 × 6 entries of the global T-matrix for a reference arrangement of 4 spheres following the ordering described previously.

A complication in using HDF5 is the lack of a native complex number data type. Typically, complex numbers are stored as compound data types consisting of two floating-point numbers. The names of the fields are often "r" and "i", but other conventions also exist. It is highly recommended to use the mentioned field names if possible.

Another issue with HDF5 is the order of the array, namely if row-major or column-major order is used. HDF5 itself uses row-major order [112]. However, some programming languages use column-major order natively, like Fortran, MATLAB, or Julia. This can lead to issues when exchanging data between programs that use a different order, if the standard HDF5 Fortran wrapper is not used. To avoid ambiguity, the final formatting in row-major order is expected. For Python scripts, this is the default behavior, while for MATLAB, Julia, we refer to a supplementary link in [81].

Finally, it should be mentioned that in the description of data entries we are following the type definitions in Python. As such, by claiming that type of the attribute is a string, we refer to `str` text type of a variable, which corresponds to `char` type in MATLAB. By claiming that a dataset is a one-dimensional array, a Numpy array is referred to, which does not have any additional dimensions of length 1. Thus, it is equivalent to a vector in MATLAB definitions. In some languages, the identical representations are not easily attainable, thus known alternatives in other languages will be considered and converted correspondingly when reading into the database.

D.2. Tools for data format conversion

Several open-source programs are available to compute T-matrices, but many of them do not (yet) implement the output format presented herein. While we encourage the community to add this functionality in order to fully benefit from interoperability between programs, it can also be useful, as a short-term or one-time workaround, to *convert* T-matrix data stored in a different form. One example is the "long format" used to store T-matrix entries in earlier versions of SMARTIES [41], or Scuff-EM [113], or PXTAL [114], among others. We include example scripts at [81] to reshape such data and produce a standard `.h5` format.

Conversion to wide format, and export as `.tmat.h5`, can then be done by adding the required geometry and material information to make a complete entry. Basic export scripts are available in 4 different languages (R, Julia, MATLAB, Python) to serve as examples for similar conversion tasks.

Appendix E. Units

The accepted units are listed below, "Hz" can be substituted by "s^{-1}". For inverse length unit,"^{-1}" should be appended to the length unit.

Value	Frequencies	Lengths
1e−24	"yHz"	"ym"
1e−21	"zHz"	"zm"
1e−18	"aHz"	"am"
1e−15	"fHz"	"fm"
1e−12	"pHz"	"pm"
1e−9	"nHz"	"nm"
1e−6	"uHz"	"um"
1e−3	"mHz"	"mm"
1e−2	"cHz"	"cm"
1e−1	"dHz"	"dm"
1	"Hz"	"m"
1e1	"daHz"	"dam"
1e2	"hHz"	"hm"
1e3	"kHz"	"km"
1e6	"MHz"	"Mm"
1e9	"GHz"	"Gm"
1e12	"THz"	"Tm"
1e15	"PHz"	"Pm"
1e18	"EHz"	"Em"
1e21	"ZHz"	"Zm"
1e24	"YHz"	"Ym"

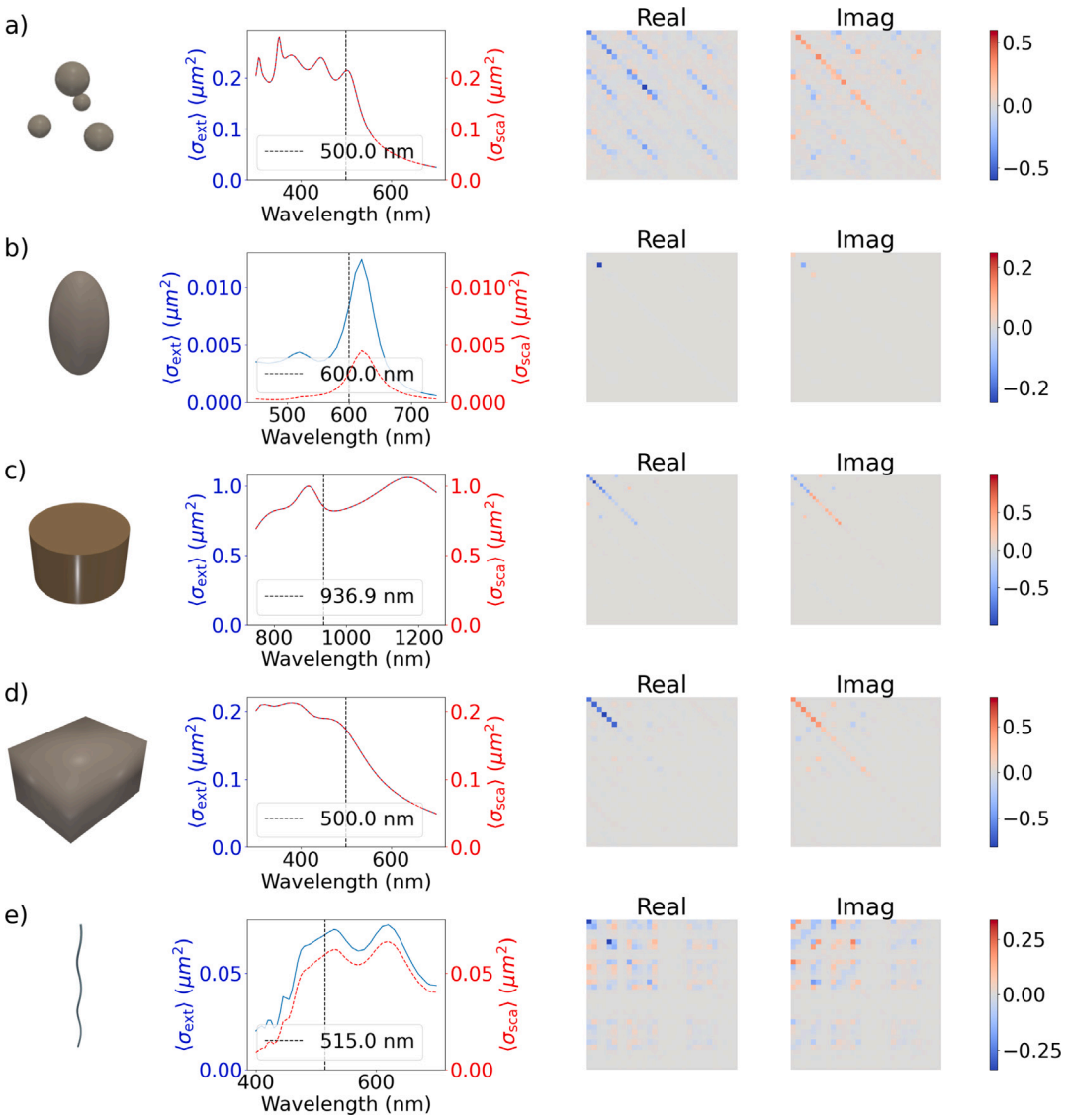

Fig. F.8. Left to right: shape, averaged cross-sections, T-matrix at single wavelength. a) 4 spheres (n = 3) in vacuum: radii: 50, 60, 70, 80 nm, positions: $a \cdot \left(\left(-\frac{1}{2}, -\frac{\sqrt{3}}{6}, -\frac{\sqrt{6}}{12}\right), \left(+\frac{1}{2}, -\frac{\sqrt{3}}{6}, -\frac{\sqrt{6}}{12}\right), \left(0, \frac{\sqrt{3}}{3}, -\frac{\sqrt{6}}{12}\right), \left(0, 0, \frac{\sqrt{6}}{4}\right)\right)$, $a = 300$ nm b) Gold [115] spheroid in water (n = 1.33): $r_{xy} = 20$ nm, $r_z = 40$ nm c) TiO_2 cylinder (n = 2.5) in vacuum: radius = 250 nm, height = 300 nm d) Anisotropic cuboid ($\epsilon_{xx} = 3.24$, $\epsilon_{yy} = 4$, $\epsilon_{zz} = 4.84$) in vacuum: $L_x = 250$ nm, $L_y = 200$ nm, $L_z = 150$ nm e) Silver [115] helix with right handedness and flat edges in vacuum: number of turns = 2.5, $r_{\text{wire}} = 21.25$ nm, pitch = 200 nm, $r_{\text{helix}} = 41.25$ nm.

Appendix F. Example structures

As a demonstration, we selected a few examples of T-matrices computed for different structures. The scripts necessary to generate these files are part of the datasets provided in [81]. These examples are computed using JCMsuite software. The purpose of these examples is to consider them in the future as additional reference examples. If you can reproduce these T-matrices at admissible accuracy with your codes, you likely have an implementation that agrees with the assumptions made throughout the manuscript here. The examples are the following and shown in Fig. F.8. The orientation-averaged extinction and scattering cross-sections plotted over a range of wavelengths are of interest here. a) shows an arrangement of four spheres. This example presents the analytical solution of the scattering problem at optical frequencies. The T-matrix is expanded at the common center of origin, thus a large number of multipoles ($l_{\max} = 6$) is considered. In b), the spectrum of a gold spheroid in water is demonstrated in the optical frequency range. Maximum mesh size used in the simulations is 1 nm for the spheroid and 3 nm for the embedding medium. This example exhibits rotational symmetry. The material is dispersive and typical plasmonic resonance is captured. Next, in c), the scattering response of a cylinder made of titanium dioxide is computed. Maximum mesh size used in the simulations is fixed to 6 and 15 nm for object and embedding medium, respectively. The relative permittivity value is taken as a constant. Two peaks at the resonant frequencies are observed. In d), the scatterer geometry does not have rotational symmetry, but mirror symmetry. Artificially set permittivity values were selected to demonstrate the capability of simulating objects made from anisotropic materials. Typical maximum mesh size for the object and the embedding medium were set to $\frac{\lambda_0}{44}$ and $\frac{\lambda_0}{20}$, where λ_0 is the vacuum wavelength. As a concluding example, in e) a silver helix is presented, with one dimension considerably larger than the others. Typical maximum mesh size is $\frac{\lambda_0}{80}$, $\frac{\lambda_0}{20}$ for the object and the embedding medium, correspondingly. No symmetry is present in this geometry.

Appendix G. Full code of the example in Section 5.3

```python
import h5py
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import treams
import treams.io

mpl.rcParams["lines.linewidth"] = 2.5
mpl.rcParams["font.size"] = 20

cyl = treams.io.load_hdf5("cylinder_tio2.tmat.h5")

positions = [[-300, 0, 0], [300, 0, 0]]
cyl_cluster = [
    treams.TMatrix.cluster([tm, tm], positions)
    for tm in cyl
]
cyl_cluster = [
    tm.interaction.solve().expand(
        treams.SphericalWaveBasis.default(4)
    )
    for tm in cyl_cluster
]

with h5py.File("two_cylinders.tmat.h5", "w") as fobj:
    treams.io.save_hdf5(fobj, cyl_cluster)

lmax = max(cyl[0].basis.l)
cyl_by_lmax = [
    [tm[treams.SphericalWaveBasis.default(i)]
    for tm in cyl]
    for i in range(1, lmax + 1)
]
cyl_cluster_by_lmax = [
    [tm[treams.SphericalWaveBasis.default(i)]
    for tm in cyl_cluster]
    for i in range(1, lmax + 1)
]

fig, axs = plt.subplots(1, 2, figsize=(16, 6))

freqs = np.array([tm.k0 * 299792.458 / (2 * np.pi) for tm in cyl])
lambdas = np.array([2 * np.pi / tm.k0 for tm in cyl])
linestyles = [":","--","-.", "-"]
for i in range(lmax):
    axs[0].plot(
        freqs,
        [tm.xs_ext_avg / 1e6 for tm in cyl_by_lmax[i]],
        linestyle=linestyles[i % lmax],
        zorder=-i,
    )

axs[0].set_xlabel("Frequency (THz)")
axs[0].set_ylabel("Cross-section ($\mu m^2$)")
axs[0].set_xlim([240, 400])

ax_top = axs[0].twiny()
ax_top.set_xlabel("Wavelength (nm)")
ax_top.set_xlim(lambdas[0], lambdas[-1])

axs[0].legend([f"$l_\\mathrm{{max}} = {i}$" for i in range(1, lmax + 1)])

for i in range(lmax):
    axs[1].plot(
        freqs,
        [tm.xs_ext_avg / 1e6 for tm in cyl_cluster_by_lmax[i]],
        linestyle=linestyles[i % lmax],
        zorder=-i,
    )

axs[1].set_xlabel("Frequency (THz)")
axs[1].set_ylabel("Cross-section ($\mu m^2$)")
axs[1].set_xlim([240, 400])

ax_top = axs[1].twiny()
ax_top.set_xlabel("Wavelength (nm)")
ax_top.set_xlim(lambdas[0], lambdas[-1])

fig.savefig("xs_cyl_tio2.png")
```

Listing 17: Full script for *treams*

Appendix H. Alphabetical list of pre-defined *groups*, *datasets* and *attributes*

The following table contains all reserved names. The color indicates the type: *groups* are black, *datasets* are blue, *attributes* are red and *softlinks* are green. Dummy *group* names are in uppercase. Here, the required entries are printed with an asterisk, the entries with required presence depending on the presence of other entries in the file are denoted with **, while the optional entries, i.e., entries that only provide additional information or have a default value when absent, are listed without an asterisk. If the parent is optional, but has a required child, the child is marked as required.

Parameter List	
Name of the group/dataset	Type
inner_dims for all datasets	int
/angular_frequency **	array: float, complex; scalar: float, complex
/angular_frequency/unit *	str
/angular_vacuum_wavenumber **	array: float, complex; scalar: float, complex
/angular_vacuum_wavenumber/unit *	str
/application	str
/computation/analytical_zeros	array: int
/computation/files	str
/computation/description	str
/computation/keywords	str
/computation/method *	str
/computation/method_parameters/	
/computation/name	str
/computation/software *	str
/computation/reference	str
/description	str
/embedding/bianisotropy	array: float, complex
/embedding/bianisotropy/coordinate_system	str
/embedding/chirality	array: float, complex; scalar: float, complex
/embedding/chirality/coordinate_system	str

/embedding/description	str
/embedding/experimental_data/	
/embedding/keywords	str
/embedding/name	str
/embedding/non-reciprocity	array: float, complex; scalar: float, complex
/embedding/non-reciprocity/coordinate_system	str
/embedding/reference	str
/embedding/refractive_index **	array: float, complex; scalar: float, complex
/embedding/refractive_index/coordinate_system	str
/embedding/relative_impedance **	array: float, complex; scalar: float, complex
/embedding/relative_permeability **	array: float, complex; scalar: float, complex
/embedding/refractive_index/coordinate_system	str
/embedding/relative_permeability/coordinate_system	str
/embedding/relative_permittivity **	array: float, complex; scalar: float, complex
/embedding/relative_permittivity/coordinate_system	str
/frequency **	array : float, complex
/frequency/unit *	str
/keywords	str
/mesh	.msh, .STL, etc
/modes/index **	array: int
/modes/index_incident **	array: int
/modes/index_scattered **	array: int
/modes/l_incident **	array: int
/modes/l_scattered **	array: int
/modes/l **	array: int
/modes/l_incident **	array: int
/modes/l_scattered **	array: int
/modes/m **	array: int
/modes/m_incident **	array: int
/modes/m_scattered **	array: int
/modes/polarization **	array: str
/modes/polarization_incident **	array: str
/modes/polarization_scattered **	array: str
/modes/positions **	array: float
/NAME/geometry/euler_angles	array: float
/NAME/geometry/expansion_center	array: float
/NAME/geometry/description	str
/NAME/geometry/keywords	str
/NAME/geometry/mesh/ or /NAME/geometry/mesh.XYZ ** (entries containing mesh: same with /computation/ possible)	.msh, .STL, etc.
/NAME/geometry/mesh.XYZ/file_extension	str
/NAME/geometry/mesh/unit or /NAME/geometry/mesh.XYZ/unit *	str
/NAME/geometry/name	str
/NAME/geometry/position	array: float
/NAME/geometry/shape **	str
/NAME/material/bianisotropy	array: float, complex
/NAME/material/bianisotropy/coordinate_system	str
/NAME/material/chirality	array: float, complex; scalar: float, complex
/NAME/material/chirality/coordinate_system	str
/NAME/material/description	str
/NAME/material/experimental_data/	
/NAME/material/interpolation	str
/NAME/material/keywords	str
/NAME/material/name	str
/NAME/material/non-reciprocity	array: float, complex; scalar: float, complex
/NAME/material/non-reciprocity/coordinate_system	str
/NAME/material/reference	str
/NAME/material/refractive_index **	array: float, complex; scalar: float, complex
/NAME/material/refractive_index/coordinate_system	str
/NAME/material/relative_impedance **	array: float, complex; scalar: float, complex
/NAME/material/relative_impedance/coordinate_system	str
/NAME/material/relative_permeability **	array: float, complex; scalar: float, complex
/NAME/material/relative_permeability/coordinate_system	str
/NAME/material/relative_permittivity **	array: float, complex; scalar: float, complex
/NAME/material/relative_permittivity/coordinate_system	str
/rmatrix	array: float, complex
/storage_format_version *	str
/tmatrix *	array: float, complex
/vacuum_wavelength **	array: float, complex; scalar: float, complex
/vacuum_wavelength/unit *	array: float, complex; scalar: float, complex
/vacuum_wavenumber **	array: float, complex; scalar: float, complex
/vacuum_wavenumber/unit *	str

Data availability

All data is available on a publically available GitHub page https://github.com/tfp-photonics/tmatrix_data_format.

References

[1] Waterman PC. Matrix formulation of electromagnetic scattering. Proc IEEE 1965;53(8):805–12.

[2] Mishchenko M, Martin P. Peter Waterman and T-matrix methods. J Quant Spectrosc Radiat Transfer 2013;123:2–7.
[3] Nieminen TA, Rubinsztein-Dunlop H, Heckenberg N. Calculation of the T-matrix: general considerations and application of the point-matching method. J Quant Spectrosc Radiat Transfer 2003;79:1019–29.
[4] Bohren CF, Huffman DR. Absorption and scattering of light by small particles. John Wiley & Sons; 2008.
[5] Nečada M, Törmä P. Multiple-scattering T-matrix simulations for nanophotonics: Symmetries and periodic lattices. Commun Comput Phys 2021;30(2):357–95.
[6] Liu W, Kivshar YS. Multipolar interference effects in nanophotonics. Phil Trans R Soc A 2017;375(2090):20160317.
[7] Mun J, Rho J. Importance of higher-order multipole transitions on chiral nearfield interactions. Nanophotonics 2019;8(5):941–8.
[8] Colom R, Devilez A, Bonod N, Stout B. Optimal interactions of light with magnetic and electric resonant particles. Phys Rev B 2016;93(4):045427.
[9] Mishchenko MI, Travis LD, Mackowski DW. T-matrix computations of light scattering by nonspherical particles: A review. J Quant Spectrosc Radiat Transfer 1996;55(5):535–75.
[10] Mackowski DW, Mishchenko MI. Calculation of the T matrix and the scattering matrix for ensembles of spheres. J Opt Soc Amer A 1996;13(11):2266–78.
[11] Pattelli L, Egel A, Lemmer U, Wiersma DS. Role of packing density and spatial correlations in strongly scattering 3D systems. Optica 2018;5(9):1037–45.
[12] Khlebtsov NG. T-matrix method in plasmonics: An overview. J Quant Spectrosc Radiat Transfer 2013;123:184–217.
[13] Theobald D, Beutel D, Borgmann L, Mescher H, Gomard G, Rockstuhl C, et al. Simulation of light scattering in large, disordered nanostructures using a periodic T-matrix method. J Quant Spectrosc Radiat Transfer 2021;272:107802.
[14] Mackowski D. The extension of the Multiple Sphere T Matrix code to include multiple plane boundaries and 2-D periodic systems. J Quant Spectrosc Radiat Transfer 2022;290:108292.
[15] Czajkowski KM, Bancerek M, Antosiewicz TJ. Multipole analysis of substrate-supported dielectric nanoresonator metasurfaces via the T-matrix method. Phys Rev B 2020;102(8):085431.
[16] Ustimenko N, Baryshnikova KV, Melnikov R, Kornovan D, Ulyantsev V, Chichkov BN, et al. Multipole optimization of light focusing by silicon nanosphere structures. J Opt Soc Am B 2021;38(10):3009–19.
[17] Fazel-Najafabadi A, Auguié B. Orientation-averaged light scattering by nanoparticle clusters: far-field and near-field benchmarks of numerical cubature methods. J Quant Spectrosc Radiat Transfer 2022;286:108197.
[18] Vandenbroucke B, Baes M, Camps P. CosTuuM: polarized thermal dust emission by magnetically oriented spheroidal grains. Astron J 2020;160(1):55.
[19] Olmos-Trigo J, Lasa-Alonso J, Gómez-Viloria I, Molina-Terriza G, García-Etxarri A. Capturing near-field circular dichroism enhancements from far-field measurements. Phys Rev Res 2024;6(1):013151.
[20] Wriedt T, Hellmers J, Eremina E, Schuh R. Light scattering by single erythrocyte: comparison of different methods. J Quant Spectrosc Radiat Transfer 2006;100(1–3):444–56.
[21] Tsang L, Liao T-H, Gao R, Xu H, Gu W, Zhu J. Theory of microwave remote sensing of vegetation effects, SoOp and rough soil surface backscattering. Remote Sens 2022;14(15):3640.
[22] Hansen GM. Mie scattering as a technique for the sizing of air bubbles. Appl Opt 1985;24(19):3214–20.
[23] Olmos-Trigo J. Solving maxwell's equations using polarimetry alone. Nano Lett 2024;24(28):8658–63.
[24] Mishchenko MI. Gustav Mie and the fundamental concept of electromagnetic scattering by particles: a perspective. J Quant Spectrosc Radiat Transfer 2009;110(14–16):1210–22.
[25] Almpanis E, Zouros GP, Papanikolaou N. Spherical optomagnonic resonators. In: Optomagnonic structures: novel architectures for simultaneous control of light and spin waves. World Scientific; 2021, p. 243–97.
[26] Zouros GP, Kolezas GD, Stefanou N, Wriedt T. EBCM for Electromagnetic Modeling of Gyrotropic BoRs. IEEE Trans Antennas and Propagation 2021;69(9):6134–9.
[27] Barber P, Yeh C. Scattering of electromagnetic waves by arbitrarily shaped dielectric bodies. Appl Opt 1975;14(12):2864–72.
[28] Zouros GP, Kolezas GD, Katsinos K. EM Scattering by Core-Shell Gyroelectric-Isotropic and Isotropic-Gyroelectric BoRs Using the EBCM. IEEE J Multiscale Multiphys Comput Tech 2022;7:117–25.
[29] Demésy G, Auger J-C, Stout B. Scattering matrix of arbitrarily shaped objects: combining finite elements and vector partial waves. J Opt Soc Amer A 2018;35(8):1401–9.
[30] Somerville W, Auguié B, Le Ru E. Accurate and convergent T-matrix calculations of light scattering by spheroids. J Quant Spectrosc Radiat Transfer 2015;160:29–35.
[31] Scheffler M, Aeschlimann M, Albrecht M, Bereau T, Bungartz H-J, Felser C, et al. FAIR data enabling new horizons for materials research. Nature 2022;604(7907):635–42.
[32] Ryabchykov O, Guo S, Bocklitz T. Photonic data analysis in 2050. Vib Spectrosc 2024;132:103685.
[33] Heidenreich S, Bär M, Klauenberg K, Elster C, Harris P, Lines K, et al. Strategic agenda, European metrology network for mathematics and statistics. EURAMET eV; 2023.
[34] Doicu A, Wriedt T, Eremin YA. Light scattering by systems of particles: null-field method with discrete sources: theory and programs, vol. 124, Springer; 2006.
[35] Martin T. T-matrix method for closely adjacent obstacles. J Quant Spectrosc Radiat Transfer 2019;234:40–6.
[36] Beutel D, Fernandez-Corbaton I, Rockstuhl C. treams–a T-matrix-based scattering code for nanophotonics. Comput Phys Comm 2024;297:109076.
[37] Egel A, Czajkowski KM, Theobald D, Ladutenko K, Kuznetsov AS, Pattelli L. SMUTHI: A python package for the simulation of light scattering by multiple particles near or between planar interfaces. J Quant Spectrosc Radiat Transfer 2021;273:107846.
[38] Egel A, Pattelli L, Mazzamuto G, Wiersma DS, Lemmer U. CELES: CUDA-accelerated simulation of electromagnetic scattering by large ensembles of spheres. J Quant Spectrosc Radiat Transfer 2017;199:103–10.
[39] Schebarchov D, Fazel-Najafabadi A, Le Ru EC, Auguié B. Multiple scattering of light in nanoparticle assemblies: user guide for the terms program. J Quant Spectrosc Radiat Transfer 2022;284:108131.
[40] Stefanou N, Yannopapas V, Modinos A. MULTEM 2: A new version of the program for transmission and band-structure calculations of photonic crystals. Comput Phys Commun 2000;132(1–2):189–96.
[41] Somerville W, Auguié B, Le Ru EC. SMARTIES: User-friendly codes for fast and accurate calculations of light scattering by spheroids. J Quant Spectrosc Radiat 2016;174:39–55.
[42] Barber PW. Differential-scattering of electromagnetic waves by homogeneous isotrobic dielectric bodies. University of California, Los Angeles; 1973.
[43] Schneider JB, Peden IC. Differential cross section of a dielectric ellipsoid by the T-matrix extended boundary condition method. IEEE Trans Antennas and Propagation 1988;36(9):1317–21.
[44] Laitinen H, Lumme K. T-matrix method for general star-shaped particles: first results. J Quant Spectrosc Radiat Transfer 1998;60(3):325–34.
[45] Wriedt T. Using the T-Matrix method for light scattering computations by non-axisymmetric particles: Superellipsoids and realistically shaped particles. Part Part Syst Charact 2002;19(4):256–68.
[46] Yurkin MA, Kahnert M. Light scattering by a cube: Accuracy limits of the discrete dipole approximation and the T-matrix method. J Quant Spectrosc Radiat Transfer 2013;123:176–83.
[47] Das A, Balasubramanian M, Campbell SD, Werner PL, Werner DH. A Surface Integral Equation Based T-Matrix Formulation for Penetrable Obstacles and Scatterers. In: IEEE international symposium on antennas and propagation and USNC-URSI radio science meeting. IEEE; 2023, p. 1829–30.
[48] Mackowski DW. Discrete dipole moment method for calculation of the T matrix for nonspherical particles. J Opt Soc Amer A 2002;19(5):881–93.
[49] Loke VLY, Nieminen TA, Heckenberg NR, Rubinsztein-Dunlop H. T-matrix calculation via discrete dipole approximation, point matching and exploiting symmetry. J Quant Spectrosc Radiat Transfer 2009;110(14–16):1460–71.
[50] Markkanen J, Yuffa AJ. Fast superposition T-matrix solution for clusters with arbitrarily-shaped constituent particles. J Quant Spectrosc Radiat Transfer 2017;189:181–8.
[51] Bi L, Yang P, Kattawar GW, Mishchenko MI. Efficient implementation of the invariant imbedding T-matrix method and the separation of variables method applied to large nonspherical inhomogeneous particles. J Quant Spectrosc Radiat Transfer 2013;116:169–83.
[52] Doicu A, Wriedt T, Khebbache N. An overview of the methods for deriving recurrence relations for T-matrix calculation. J Quant Spectrosc Radiat Transfer 2019;224:289–302.
[53] https://scattport.org/.
[54] So S, Mun J, Rho J. Simultaneous inverse design of materials and structures via deep learning: demonstration of dipole resonance engineering using core–shell nanoparticles. ACS Appl Mater Interfaces 2019;11(27):24264–8.
[55] Talebi-Moghaddam S, Bauer F, Huber F, Will S, Daun K. Inferring soot morphology through multi-angle light scattering using an artificial neural network. J Quant Spectrosc Radiat Transfer 2020;251:106957.
[56] Antosiewicz TJ, Apell SP. Plasmonic glasses: optical properties of amorphous metal-dielectric composites. Opt Express 2014;22(2):2031–42.
[57] Agocs E, Bodermann B, Burger S, Dai G, Endres J, Hansen P-E, et al. Scatterometry reference standards to improve tool matching and traceability in lithographical nanomanufacturing. In: Nanoengineering: fabrication, properties, optics, and devices XII, vol. 9556, SPIE; 2015, p. 153–64.
[58] Mackowski DW. Chapter 6 - T-matrix method for particles of arbitrary shape and composition. In: Mengüç MP, Francoeur M, editors. Light, plasmonics and particles. Nanophotonics, Elsevier; 2023, p. 113–31.
[59] Alaee R, Rockstuhl C, Fernandez-Corbaton I. Exact multipolar decompositions with applications in nanophotonics. Adv Opt Mater 2019;7(1):1800783.
[60] Schebarchov D, Le Ru EC, Grand J, Auguié B. Mind the gap: testing the Rayleigh hypothesis in T-matrix calculations with adjacent spheroids. Opt Express 2019;27(24):35750–60.
[61] Lamprianidis AG, Rockstuhl C, Fernandez-Corbaton I. Transcending the Rayleigh Hypothesis with multipolar sources distributed across the topological skeleton of a scatterer. J Quant Spectrosc Radiat Transfer 2023;296:108455.

[62] Bertrand M, Devilez A, Hugonin J-P, Lalanne P, Vynck K. Global polarizability matrix method for efficient modeling of light scattering by dense ensembles of non-spherical particles in stratified media. J Opt Soc Amer A 2020;37(1):70–83.
[63] Jackson JD. Classical electrodynamics. New York: John Wiley & Sons, Inc.; 1998.
[64] Kildishev AV, Achouri K, Smirnova D. The art of finding the optimal scattering center (s). Adv Opt Mater 2023;2402787.
[65] Katsinos K, Zouros GP, Roumeliotis JA. An entire domain CFVIE-CDSE method for EM scattering on electrically large highly inhomogeneous gyrotropic circular cylinders. IEEE Trans Antennas and Propagation 2021;69(4):2256–66.
[66] Lalanne P, Yan W, Vynck K, Sauvan C, Hugonin J-P. Light interaction with photonic and plasmonic resonances. Laser Photonics Rev 2018;12(5):1700113.
[67] Lalanne P, Yan W, Gras A, Sauvan C, Hugonin J-P, Besbes M, et al. Quasinormal mode solvers for resonators with dispersive materials. J Opt Soc Amer A 2019;36(4):686–704.
[68] Vavilin M, Fernandez-Corbaton I. The polychromatic T-matrix. J Quant Spectrosc Radiat Transfer 2024;314:108853.
[69] Rotter S, Gigan S. Light fields in complex media: Mesoscopic scattering meets wave control. Rev Modern Phys 2017;89(1):015005.
[70] Cao H, Čižmár T, Turtaev S, Tyc T, Rotter S. Controlling light propagation in multimode fibers for imaging, spectroscopy, and beyond. Adv Opt Photonics 2023;15(2):524–612.
[71] Globosits D, Hüpfl J, Rotter S. Pseudounitary Floquet scattering matrix for wave-front shaping in time-periodic photonic media. Phys Rev A 2024;110(5):053515.
[72] Cruzan OR. Translational addition theorems for spherical vector wave functions. Quart Appl Math 1962;20(1):33–40.
[73] Stout B, Auger J-C, Devilez A. Recursive T matrix algorithm for resonant multiple scattering: applications to localized plasmon excitations. J Opt Soc Amer A 2008;25(10):2549–57.
[74] Dwivedi R, Aradian A, Ponsinet V, Vynck K, Baron A. Effective-medium description of dense clusters of plasmonic nanoparticles with spatial dispersion. Phys Rev A 2024;109(2):023507.
[75] Suryadharma RN, Fruhnert M, Fernandez-Corbaton I, Rockstuhl C. Studying plasmonic resonance modes of hierarchical self-assembled meta-atoms based on their transfer matrix. Phys Rev B 2017;96(4):045406.
[76] Linton CM. Lattice sums for the Helmholtz equation. SIAM Rev 2010;52(4):630–74.
[77] Schulz SA, Oulton R, Kenney M, Alù A, Staude I, Bashiri A, et al. Roadmap on photonic metasurfaces. Appl Phys Lett 2024;124(26):260701.
[78] Yannopapas V. Negative refraction in random photonic alloys of polaritonic and plasmonic microspheres. Phys Rev B 2007;75(3):035112.
[79] Achouri K, Tiukuvaara V, Martin OJ. Multipolar modeling of spatially dispersive metasurfaces. IEEE Trans Antennas and Propagation 2022;70(12):11946–56.
[80] Wu T, Baron A, Lalanne P, Vynck K. Intrinsic multipolar contents of nanoresonators for tailored scattering. Phys Rev A 2020;101(1):011803.
[81] https://github.com/tfp-photonics/tmatrix_data_format/.
[82] Tsitsas NL, Athanasiadis C. On the scattering of spherical electromagnetic waves by a layered sphere. Quart J Mech Appl Math 2006;59(1):55–74.
[83] Langevin D, Bennet P, Khaireh-Walieh A, Wiecha P, Teytaud O, Moreau A. PyMoosh: a comprehensive numerical toolkit for computing the optical properties of multilayered structures. J Opt Soc Am B 2024;41(2):A67–78.
[84] Santiago XG, Hammerschmidt M, Burger S, Rockstuhl C, Fernandez-Corbaton I, Zschiedrich L. Decomposition of scattered electromagnetic fields into vector spherical wave functions on surfaces with general shapes. Phys Rev B 2019;99(4):045406.
[85] Burger S, Zschiedrich L, Pomplun J, Schmidt F, Bodermann B. Fast simulation method for parameter reconstruction in optical metrology. Proc SPIE 2013;8681:868119.
[86] Huang H, Tsang L, Njoku EG, Colliander A, Ding K-H, Liao T-H. Hybrid method combining generalized T matrix of single objects and Foldy-Lax equations in NMM3D microwave scattering in vegetation. In: 2017 progress in electromagnetics research symposium-fall. IEEE; 2017, p. 3016–23.
[87] Gladyshev S, Pashina O, Proskurin A, Nikolaeva A, Sadrieva Z, Petrov M, et al. Fast simulation of light scattering and harmonic generation in axially symmetric structures in COMSOL. ACS Photonics 2024;11(2):404–18.
[88] Geuzaine C, Henrotte F, Remacle J-F, Marchandise E, Sabariego R. Onelab: open numerical engineering laboratory. In: Colloque National en Calcul des Structures. 2013.
[89] Geuzaine C. GetDP: A general finite-element solver for the de Rham complex. Pamm 2007;7:1010603–4.
[90] Geuzaine C, Remacle J-F. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. Internat J Numer Methods Engrg 2009;79(11):1309–31.
[91] Hohenester U. Nano and quantum optics. Cham: Springer; 2020.
[92] Hohenester U, Reichelt N, Unger G. Nanophotonic resonance modes with the nanobem toolbox. Comput Phys Comm 2022;276:108337.
[93] Hohenester U. Nanophotonic resonators in stratified media with the nanobem toolbox. Comput Phys Comm 2024;294:108949.
[94] Chew WC. Waves and fields in inhomogenous media, vol. 16, John Wiley & Sons; 1999.
[95] Yurkin MA, Hoekstra AG. The discrete-dipole-approximation code adda: Capabilities and known limitations. J Quant Spectrosc Radiat Transfer 2011;112(13):2234–47.
[96] Yurkin MA, Hoekstra AG. The discrete dipole approximation: an overview and recent developments. J Quant Spectrosc Radiat Transfer 2007;106(1–3):558–89.
[97] Chaumet PC. The discrete dipole approximation: A review. Mathematics 2022;10(17):3049.
[98] Fruhnert M, Fernandez-Corbaton I, Yannopapas V, Rockstuhl C. Computing the T-matrix of a scattering object with multiple plane wave illuminations. Beilstein J Nanotechnol 2017;8(1):614–26.
[99] https://gitlab.com/k.czajkowski/addatmatrix.
[100] Chaumet PC, Zhang T, Sentenac A. Fast far-field calculation in the discrete dipole approximation. J Quant Spectrosc Radiat Transfer 2015;165:88–92.
[101] Oskooi AF, Roundy D, Ibanescu M, Bermel P, Joannopoulos JD, Johnson SG. MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method. Comput Phys Comm 2010;181(3):687–702.
[102] Mishchenko MI, Travis LD, Lacis AA. Scattering, absorption, and emission of light by small particles. Cambridge University Press; 2002.
[103] Somerville W, Auguié B, Le Ru E. SMARTIES. 2024, http://dx.doi.org/10.5281/zenodo.1234, URL https://github.com/nano-optics/smarties/.
[104] Salerno G, Heilmann R, Arjas K, Aronen K, Martikainen J-P, Törmä P. Loss-driven topological transitions in lasing. Phys Rev Lett 2022;129(17):173901.
[105] Le Ru EC, Somerville WR, Auguié B. Radiative correction in approximate treatments of electromagnetic scattering by point and body scatterers. Phys Rev A 2013;87(1):012504.
[106] Schebarchov D, Fazel-Najafabadi A, Le Ru EC, Auguié B. TERMS website. 2021, http://dx.doi.org/10.5281/zenodo.5703291, URL http://nano-optics.ac.nz/terms.
[107] Wiecha PR, Arbouet A, Girard C, Muskens OL. Deep learning in nano-photonics: inverse design and beyond. Photonics Res. 2021;9(5):B182–200.
[108] Nees N, Pflug L, Mann B, Stingl M. Multi-material design optimization of optical properties of particulate products by discrete dipole approximation and sequential global programming. Struct Multidiscip Optim 2023;66(1):5.
[109] Gladyshev S, Karamanos TD, Kuhn L, Beutel D, Weiss T, Rockstuhl C, et al. Inverse design of all-dielectric metasurfaces with accidental bound states in the continuum. Nanophotonics 2023;12(19):3767–79.
[110] Khaireh-Walieh A, Langevin D, Bennet P, Teytaud O, Moreau A, Wiecha PR. A newcomer's guide to deep learning for inverse design in nano-photonics. Nanophotonics 2023;12(24):4387–414.
[111] Doicu A, Eremin Y, Wriedt T. Transition matrix of a nonspherical particle in the non-local optical response theory. J Quant Spectrosc Radiat Transfer 2020;242:106756.
[112] https://support.hdfgroup.org/HDF5/doc1.6/UG/12_Dataspaces.html.
[113] Homer Reid. BUFF-EM: A Volume-Integral Solver Suite for Classical Scattering and Fluctuational Electrodynamics, URL https://github.com/HomerReid/buff-em.
[114] García de Abajo FJ. Multiple scattering of radiation in clusters of dielectrics. Phys Rev B 1999;60:6086–102.
[115] Johnson PB, Christy R-W. Optical constants of the noble metals. Phys Rev B 1972;6(12):4370.