



ISTITUTO NAZIONALE DI RICERCA METROLOGICA Repository Istituzionale

SMUTHI: A python package for the simulation of light scattering by multiple particles near or between planar interfaces

This is the author's submitted version of the contribution published as:

Original

SMUTHI: A python package for the simulation of light scattering by multiple particles near or between planar interfaces / Egel, Amos; Czajkowski, Krzysztof M.; Theobald, Dominik; Ladutenko, Konstantin; Kuznetsov, Alexey S.; Pattelli, Lorenzo. - In: JOURNAL OF QUANTITATIVE SPECTROSCOPY & RADIATIVE TRANSFER. - ISSN 0022-4073. - 273:(2021), p. 107846. [10.1016/j.jqsrt.2021.107846]

Availability:

This version is available at: 11696/71400 since: 2025-03-01T18:01:44Z

Publisher:

Elsevier

Published

DOI:10.1016/j.jqsrt.2021.107846

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)

SMUTHI: A python package for the simulation of light scattering by multiple particles near or between planar interfaces

Amos Egel^a, Krzysztof M. Czajkowski^{b,*}, Dominik Theobald^{c,*},
Konstantin Ladutenko^d, Alexey S. Kuznetsov^d, Lorenzo Pattelli^{e,f}

^aHembach Photonik GmbH, 91126 Rednitzhembach, Germany

^bFaculty of Physics, University of Warsaw, 02-093, Warsaw, Poland

^cLight Technology Institute (LTI), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany

^dDepartment of Physics and Engineering, ITMO University, 197101 St. Petersburg, Russia

^eIstituto Nazionale di Ricerca Metrologica (INRiM), 10135 Torino, Italy

^fEuropean Laboratory for Non-linear Spectroscopy (LENS), 50019 Sesto Fiorentino, Italy

arXiv:2105.04259v1 [physics.optics] 10 May 2021

Abstract

SMUTHI is a python package for the efficient and accurate simulation of electromagnetic scattering by one or multiple wavelength-scale objects in a planarly layered medium. The software combines the T-matrix method for individual particle scattering with the scattering matrix formalism for the propagation of the electromagnetic field through the planar interfaces. In this article, we briefly introduce the relevant theoretical concepts and present the main features of SMUTHI. Simulation results obtained for several benchmark configurations are validated against commercial software solutions. Owing to the generality of planarly layered geometries and the availability of different particle shapes and light sources, possible applications of SMUTHI include the study of discrete random media, meta-surfaces, photonic crystals and glasses, perforated membranes and plasmonic systems, to name a few relevant examples at visible and near-visible wavelengths.

Keywords: Scattering, Multiple scattering, T-Matrix, Layered media, Software

1. Introduction

The efficient collection, extraction or manipulation of light is often based on the interaction between particles and a supporting substrate or a host layered medium. Prominent examples of such applications can be found in the fields of metasurfaces, microscopy, plasmonics, illumination and energy harvesting [1, 2, 3, 4, 5, 6, 7], as well as in more fundamental research areas including cavity electrodynamics, tailored resonances, Anderson localization, topological photonics or bound states in the continuum [8, 9, 10, 11, 12, 13, 14].

In many cases, the numerical modeling of such systems poses huge computational challenges, especially when dealing with arrangements of many particles that cannot be described as one small unit cell repeated with infinite periodicity. Developing new tools to model these systems is therefore a key element to advance their functionalities, improve their design and deepen our understanding of their complex, collective physical behavior.

This paper presents a Python package that allows the optical simulation of light scattering by multiple wavelength-sized particles near or between planar interfaces. It implements the superposition T-matrix method [15, 16, 17]

*Krzysztof Czajkowski and Dominik Theobald contributed equally to this work.

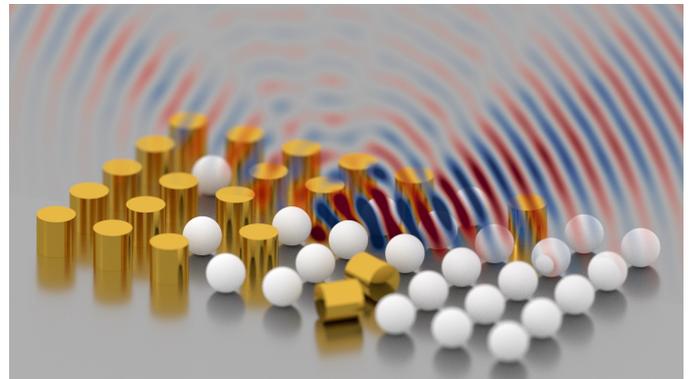


Figure 1: Artistic visualization of a Gaussian beam scattered by multiple particles on a substrate.

which, compared to mesh-based simulation approaches, is characterized by a superior computational efficiency and relatively small memory footprint, thereby enabling the simulation of large systems that would be impossible to model otherwise.

Several open source implementations dedicated to the simulation of electromagnetic scattering by multiple particles using the superposition T-matrix method have been made available in the last decade [18, 19, 20]. Despite their computational efficiency, however, these implementations are limited to the case of particles in a homogeneous back-

ground.

The here presented software lifts this limitation, by combining the advantages of the T-matrix method with the possibility to model systems involving planar interfaces.

The paper is organized as follows: Section 2 introduces the open source project behind the software. In section 3, a brief overview over the underlying theoretical concepts is given. A short user guide is provided in section 4. Section 5 illustrates some typical use case examples and establishes the validity of the code by comparison to accurate benchmark results from third-party software. Finally, we present conclusions and outlook in section 6.

2. The SMUTHI project

SMUTHI (“Scattering by MULTiple particles in THIn film systems”) is a Python package published under the MIT license. It is an open source project, initiated by the first author of this paper and further developed collaboratively by a community of scientists from different research groups.

The project is hosted on:

- <https://gitlab.com/AmosEgel/smuthi> (code repository)
- <https://pypi.org/project/SMUTHI> (Python package)
- <https://smuthi.readthedocs.io> (documentation)
- <https://groups.google.com/g/smuthi> (mailing list)

The source code follows an object oriented programming style. This allows easy access to individual modules of the code, which can be adapted to specific needs of the user or imported into other software projects.

For more demanding simulation tasks, SMUTHI allows parallelization of performance-critical operations on an NVIDIA graphics processing unit through the PyCUDA package [21].

The target group of users are scientists and engineers in the field of optics and electromagnetic scattering. Although SMUTHI offers some tools to assist the user in selecting appropriate numerical settings, a critical evaluation of the numerical results is always required to verify convergence and accuracy.

3. Simulation method

SMUTHI solves the 3D Maxwell equations in the frequency domain (i.e., one wavelength per simulation). It is based on the T-matrix method [15, 22] which was adapted to treat multiple particles (superposition T-matrix method, [16, 17]) located near or between planar interfaces [23, 24, 25].

One important advantage of the T-matrix method for multiple particles is its *modularity*: The evaluation of scattering

by a single particle is separated from the evaluation of the mutual interaction between the particles and from the evaluation of the interaction between a particle and a non-homogeneous environment, such as a particle on a substrate. Once a particle’s T-matrix is known, its scattering behavior can be evaluated under (almost) arbitrary illumination conditions.

For the evaluation of the T-matrix of an isolated particle, SMUTHI relies on the state-of-the-art NFM-DS (“Null-field method with discrete sources”) FORTRAN code by Adrian Doicu, Thomas Wriedt and Yuri Eremin [26, 27]. T-matrices computed by NFM-DS are then processed by SMUTHI to solve the problem of multiple scattering between the particles as well as between the particles and the layered medium.

In the following subsection, we sketch the basic concepts to provide a general outline of the method. The interested reader is referred to chapters 2-3 of [28] for a more complete description of the underlying theory.

3.1. The T-Matrix method for multiple particles between planar interfaces

In the context of scattering particles located near or inside planarly layered media, it is useful to present the total electric field as the sum of four constituent parts:

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}_{\text{init}}(\mathbf{r}) + \mathbf{E}_{\text{init}}^R(\mathbf{r}) + \sum_{i=1}^{N_S} \mathbf{E}_{\text{scat},i}(\mathbf{r}) + \sum_{i=1}^{N_S} \mathbf{E}_{\text{scat},i}^R(\mathbf{r}), \quad (1)$$

where \mathbf{E}_{init} denotes the *initial excitation*, $\mathbf{E}_{\text{init}}^R$ the response of the layer system to that field, $\mathbf{E}_{\text{scat},i}$ the scattered field from particle i , and $\mathbf{E}_{\text{scat},i}^R$ the response of the layer system to that field (the sums run over all N_S particles).

The central aspect of the T-matrix method is the expansion of the electromagnetic field using spherical vector wave functions. The incoming field at particle i (i.e., the initial field including layer response as well as the scattered field from all other particles including the layer response) is expanded in terms of regular spherical vector wave functions, $\Psi_n^{(1)}$ (see section B.2 of [27]),

$$\left. \begin{aligned} \mathbf{E}_{\text{init}}(\mathbf{r}) &= \sum_n a_{\text{init},n}^i \Psi_n^{(1)}(\mathbf{r} - \mathbf{r}_i) \\ \mathbf{E}_{\text{init}}^R(\mathbf{r}) &= \sum_n a_{\text{init},n}^{i,R} \Psi_n^{(1)}(\mathbf{r} - \mathbf{r}_i) \\ \mathbf{E}_{\text{scat},j}(\mathbf{r}) &= \sum_n a_{j,n}^i \Psi_n^{(1)}(\mathbf{r} - \mathbf{r}_i) \\ \mathbf{E}_{\text{scat},j}^R(\mathbf{r}) &= \sum_n a_{j,n}^{i,R} \Psi_n^{(1)}(\mathbf{r} - \mathbf{r}_i) \end{aligned} \right\} \text{for } |\mathbf{r} - \mathbf{r}_i| \leq R_i^{\text{in}}, \quad (2)$$

where R_i^{in} is radius of the largest sphere fully contained inside particle i .

On the other hand, the scattered electromagnetic field is expanded in terms of outgoing spherical vector wave functions, $\Psi_n^{(3)}$ (see section B.2 of [27]),

$$\mathbf{E}_{\text{scat},i}(\mathbf{r}) = \sum_n b_n^i \Psi_n^{(3)}(\mathbf{r} - \mathbf{r}_i) \quad \text{for } |\mathbf{r} - \mathbf{r}_i| \geq R_i^{\text{um}}, \quad (3)$$

where b_n^i are the expansion coefficients, \mathbf{r}_i is the center coordinate of particle i , and R_i^{um} is the radius of the smallest sphere that contains particle i . In the above summation, we have applied a multi-index notation, where n subsumes the spherical polarization $\tau = (\text{TE}, \text{TM})$, the multipole degree $l = 1, \dots, \infty$, and order $m = -l, \dots, l$. Note that the expansion (3) is strictly valid only outside the circumscribing sphere of particle i .

In total, we are thus dealing with five sets of expansion coefficients, two of which (the initial field coefficients and the initial field layer response coefficients) are known *a priori* and three of which are unknown. Correspondingly, there are three sets of linear equations that can be used to determine the unknown expansion coefficients. First, the T-matrix equation connects the incoming to the scattered field coefficients for each particle,

$$b_n^i = \sum_{n'} T_{nn'}^i \left(a_{\text{init},n'}^i + a_{\text{init},n'}^{i,R} + \sum_{j \neq i} a_{j,n'}^i + \sum_{j \neq i} a_{j,n'}^{i,R} \right), \quad (4)$$

where T^i denotes the T-matrix of particle i . Second and third, the coupling matrices connect the scattered field coefficients of particle j to the corresponding incoming field coefficients of particle i :

$$a_{j,n}^i = \sum_{n'} W_{j,nn'}^i b_{n'}^j \quad (5)$$

$$a_{j,n}^{i,R} = \sum_{n'} W_{j,nn'}^{i,R} b_{n'}^j \quad (6)$$

The above coupling equations define the direct (W_j^i) and layer system mediated ($W_j^{i,R}$) particle coupling operator.

The direct coupling operator describes how an outgoing spherical wave, emitted from position \mathbf{r}_j , will be perceived as a series of regular spherical waves at position \mathbf{r}_i . This translation relation is given by the spherical vector wave function addition theorem [29, 30], which can either be constructed from closed form expressions involving Wigner-3j functions [31, 32] or from an iterative scheme [27].

The layer system mediated coupling operator, on the other hand, states how the multiple transmissions and reflections of an outgoing spherical wave, emitted from position \mathbf{r}_j and propagated through the layer system, are then perceived as a series of regular spherical waves at position \mathbf{r}_i . Note that, due to the lateral translation symmetry of the planarly layered medium, the calculation of the layer system mediated coupling operator is most conveniently done using plane wave expansions rather than spherical waves. It is constructed by the following procedure:

1. Expand the outgoing spherical waves at \mathbf{r}_j into plane waves.
2. Propagate the plane waves through the layer system (using the transfer matrix method or the scattering matrix method, [25, 33]).
3. Expand each plane wave back into regular spherical waves at \mathbf{r}_i .

Further details are beyond the scope of this work and can be found in [33]. The resulting expression for $W_j^{i,R}$ (see equation (3.46) of [33]) involves an integral (the so-called *Sommerfeld integral*), which in general has to be solved numerically.

Finally, equations (5) to (6) can be inserted into (4) to yield the following linear system

$$\sum_{j \neq i} \sum_{n'} M_{j,nn'}^i b_{n'}^j = \sum_{n'} T_{nn'}^i \left(a_{\text{init},n'}^i + a_{\text{init},n'}^{i,R} \right) \quad (7)$$

with

$$M_{j,nn'}^i = \delta_{ij} \delta_{nn'} - \sum_{n''} T_{nn''}^i \left(W_{j,n''n'}^i + W_{j,n''n'}^{i,R} \right). \quad (8)$$

Solving (7) yields the scattered field coefficients, from which all quantities of interest can then be computed in a post-processing step.

It is worth noting that the dimension of the linear system (7) is proportional to the number of particles and to the number of partial waves used in the multipole expansions (3). For systems with many particles, solving (7) can quickly become a numerically substantial task.

4. User guide

This section provides an overview of the basic principles that govern a SMUTHI simulation and allows a new user to get started with the simulation of standard application scenarios.

The following instructions were tested for SMUTHI version 1.2.4. Possible future changes will be documented on the online documentation, to which the user is referred for a full description of SMUTHI application programming interface (API).

4.1. Installation

4.1.1. Hardware and software requirements

For simple simulations (e.g., a single particle on a substrate), SMUTHI does not have special hardware requirements. For heavier simulations comprising a large number of particles or involving demanding post processing steps, we recommend the use of a workstation computer with sufficient memory and with a CUDA-capable NVIDIA graphics card. Alternatively, a straightforward way to test SMUTHI

is offered by online Jupyter platforms such as Google Colab [34], which also provide free access to NVIDIA GPU hardware.

In order to run SMUTHI, Python 3.6 (or later) and the “Pip” Python package manager are required. Assuming a Linux (e.g., Ubuntu) operating system, the Foreign Function Interface library may be also needed, which can be installed with

```
sudo apt install libffi6 libffi-dev
```

In order to benefit from CUDA-accelerated calculations, a suitable NVIDIA graphics card, the NVIDIA CUDA toolkit and the PyCuda Python package are also needed. To install the latter, run

```
sudo python3 -m pip install pycuda
```

4.1.2. Installation

The following command installs the latest SMUTHI release from the Python Package Index (PyPi):

```
sudo python3 -m pip install smuthi
```

Alternatively, download the SMUTHI sources from the online Git repository (<https://gitlab.com/AmosEgel/smuthi>), browse into the project folder and install it manually by

```
sudo python3 -m pip install .
```

4.2. “Hello World”

The following code represents a minimal simulation example – a “Hello World” SMUTHI script. It simulates the extinction cross section for a glass sphere on a glass substrate:

```
import numpy as np
from smuthi.simulation import Simulation
from smuthi.initial_field import PlaneWave
from smuthi.layers import LayerSystem
from smuthi.particles import Sphere
from smuthi.postprocessing.far_field \
import extinction_cross_section

laysys = LayerSystem(thicknesses=[0,0],
                    refractive_indices=[1.52,1])

sph = Sphere(position=[0,0,100],
            refractive_index=1.52,
            radius=100,
            l_max=3)

plwv = PlaneWave(vacuum_wavelength=550,
                polar_angle=np.pi,
                azimuthal_angle=0,
                polarization=0)

simul = Simulation(layer_system=laysys,
                  particle_list=[sph],
                  initial_field=plwv)

simul.run()

ecs = extinction_cross_section(simulation=simul)
print("Extinction cross section:", ecs)
```

4.3. Building blocks of a simulation

In general, a SMUTHI simulation script contains the following building blocks (compare figure 2):

- Definition of the optical system: the initial field, the layer system and a list of scattering particles are defined
- Definition of the simulation object: the simulation object is initialized with the ingredients of the optical system. Further numerical settings can be applied.
- Simulation: The calculation is launched with the command `simulation.run()`
- Post processing: The results are processed into the desired output (for example: scattering cross section).

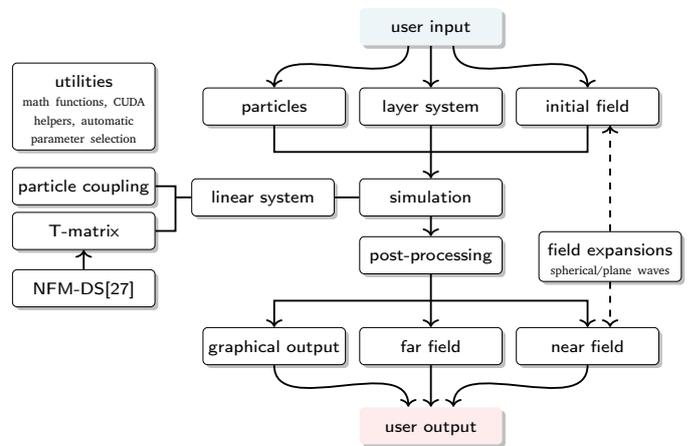


Figure 2: SMUTHI module structure and data flow

The following sections contain a description of each of these building blocks, together with some example code snippets that illustrate their usage in a SMUTHI simulation script.

4.3.1. Initial field

Currently, the following classes can be used to define the initial field:

Plane waves. An initial plane wave is specified by the vacuum wavelength, incident direction, polarization, complex amplitude and reference point. For example:

```
import numpy as np
from smuthi.initial_field import PlaneWave

plwv = PlaneWave(vacuum_wavelength=550,
                 polar_angle=np.pi,
                 azimuthal_angle=0,
                 polarization=0) # 0=TE 1=TM
```

Gaussian beams. A Gaussian beam is specified by the vacuum wavelength, incident direction, polarization, complex amplitude, beam waist and reference point. Note that for oblique incident directions, the Gaussian beam is in fact an elliptical beam, such that the electric field in the xy -plane,

i.e., parallel to the layer interfaces has a circular Gaussian footprint. For details, see [33].

A typical Gaussian beam can be implemented as follows.

```
import numpy as np
from smuthi.initial_field import GaussianBeam

focus = [0, 0, 500]
beam = GaussianBeam(vacuum_wavelength=550,
                    polar_angle=np.pi,
                    azimuthal_angle=0,
                    polarization=0,
                    reference_point=focus,
                    beam_waist=5000)
```

Further parameters can be set to control the Gaussian beam numerical precision. For details, see the API section of the online documentation.

Point dipoles. A single point dipole source is specified by the vacuum wavelength, dipole moment vector and position. The following code illustrates the definition of a dipole source:

```
from smuthi.initial_field import DipoleSource

# initialize dipole object
dip = DipoleSource(vacuum_wavelength=550,
                  dipole_moment=[1,0,0],
                  position=[0,0,300])
```

Further parameters can be set to control the dipole source numerical precision. For details, see the API section of the online documentation.

To define multiple dipole sources, create a dipole collection:

```
from smuthi.initial_field import DipoleSource, \
    DipoleCollection

# initialize dipole objects
dip1 = DipoleSource( ... )
dip2 = DipoleSource( ... )
dip3 = DipoleSource( ... )

# add them to a collection
collection = DipoleCollection(vacuum_wavelength=1d)
collection.append(dip1)
collection.append(dip2)
collection.append(dip3)
```

Note: a point dipole source must not be placed inside the circumscribing sphere of a scattering particle, except if the dipole is in a different layer than the particle.

4.3.2. Layer system

The layer system is specified by a list of layer thicknesses and a list of complex refractive indices.

Note that:

- The layer system is built from bottom to top, i.e., the first elements in the lists refer to the bottom layer.
- The interface between the bottom layer and the next layer in the layer system defines the $z = 0$ plane.
- Bottom and top layer are semi-infinite in size. You can specify a layer thickness of zero.



Figure 3: Illustration of possible particle shapes. From left to right: sphere, cylinder, spheroid, custom particle

- The minimal layer system consists of two layers (e.g., a substrate and an ambient medium). Homogeneous media without layer interfaces cannot be defined, but they can be mimicked by a trivial system of two identical layers. However, pure T-matrix implementations without the computational overhead due to planar interfaces should be preferred for this simple case.

A typical layer system definition could look like this (metal substrate with dielectric coating under vacuum):

```
from smuthi.layers import LayerSystem

# layer refractive indices
n0 = 1.0+6.1j # bottom, metal
n1 = 1.45     # middle, coating
n2 = 1       # top, vacuum

# layer thicknesses
d0 = 0       # bottom and
d1 = 120     # top must be
d2 = 0       # set to d=0

laysys = LayerSystem(thicknesses=[d0,d1,d2],
                    refractive_indices=[n0,n1,n2])
```

4.3.3. Particles

SMUTHI supports simulations comprising different types and combinations of scattering particles. A few typical geometries are shown in figure 3.

Spheres. A sphere is specified by its center position vector, complex refractive index, radius and multipole cutoff, e.g.:

```
from smuthi.particles import Sphere

sph = Sphere(position=[0,0,300],
             refractive_index=2.4,
             radius=110,
             l_max=3,
             m_max=3)
```

Spheroids. A spheroid is specified by its center position vector, Euler angles (to define the orientation in space), complex refractive index, two half axis parameters, and the multipole cutoff, e.g.:

```
from smuthi.particles import Spheroid

sphrd = Spheroid(position=[0,0,200],
                 euler_angles=[0.3,0.5,1.2],
                 refractive_index=2.4,
                 semi_axis_c=200,
                 semi_axis_a=100,
                 l_max=5,
                 m_max=5)
```

Cylinders. A cylinder is specified by its center position vector, Euler angles, complex refractive index, radius, height and multipole cutoff, e.g.:

```
from smuthi.particles import FiniteCylinder

cyl = FiniteCylinder(position=[0,0,200],
                    euler_angles=[0.3,0.5,1.2],
                    refractive_index=2.4,
                    cylinder_radius=80,
                    cylinder_height=140,
                    l_max=5,
                    m_max=5)
```

Custom particle. The custom particle class allows to model particles with arbitrary geometry. These are specified by their position vector, Euler angles, an STL (or, alternatively, FEM) file containing the particle surface mesh, a scale parameter to set the physical size of the particle (if it deviates from the size specified by the mesh file) and multipole cutoff.

```
from smuthi.particles import CustomParticle

cust = CustomParticle(position=[0,0,200],
                    euler_angles=[0.3,0.5,1.2],
                    refractive_index=2.4,
                    geometry_filename="custom.stl",
                    scale=100,
                    l_max=5,
                    m_max=5)
```

Two useful tools to generate an STL mesh file for a given geometry are GMSH [35] and trimesh [36].

When defining a scattering particle, you need to provide the parameters regarding geometry and material, as well as the parameters `l_max` and `m_max` which define the multipole expansion cutoff and can be specified for each particle independently, see section 4.4.1.

Additional notes:

- At the moment, the simulation of non-spherical particles relies on the NFM-DS Fortran code by Adrian Doicu, Thomas Wriedt and Yuri Eremin [27].
- Particles must not overlap with each other or with layer interfaces.
- The circumscribing spheres of non-spherical particles may overlap with layer interfaces (e.g., a flat particle on a substrate), but care has to be taken with regard to the selection of the numerical parameters [37, 38]. Use of SMUTHI’s automatic parameter selection feature is recommended, see section 4.5.
- The circumscribing spheres of non-spherical particles must not overlap with each other. There is a SMUTHI sub-package which offers plane-wave mediated particle coupling and thus allows treating particles with overlapping circumscribing spheres [39], but this feature is still experimental and requires expert knowledge to be used.

4.3.4. The simulation class

The `Simulation` object is the central manager of a SMUTHI simulation. To define a `Simulation`, you need to at least specify the optical system, i.e., an initial field, a layer system and a list of scattering particles. In addition, you can provide a number of input parameters regarding numerical parameters or solver settings, see section 4.4.

The following code illustrates the definition of a simulation object, using the default settings for numerical parameters:

```
from smuthi.simulation import Simulation

# initialize simulation
simul = Simulation(layer_system=laysys,
                 particle_list=[sph, cyl],
                 initial_field=beam)

# run simulation
simul.run()
```

4.3.5. Post-processing

Once the `Simulation.run()` method has successfully terminated (i.e., all unknown expansion coefficients have been determined), we still need to process the results into the desired simulation output. SMUTHI offers data structures to obtain near and far field distributions as well as scattering cross sections. Below, we give a short overview on a couple of convenience functions that can be used to quickly generate some output.

Near fields. The near field¹ is an electric field distribution as a function of position, $\mathbf{E} = \mathbf{E}(\mathbf{r})$.

The following code snippets illustrates the generation of electric field plots and animations.

```
from smuthi.postprocessing.graphical_output \
    import show_near_field

qts_to_plot = ["norm(E)", "E_y"]

show_options = [{"label": "raw_data"},
               {"interpolation": "quadric"},]

show_near_field(simulation=simul,
               quantities_to_plot=qts_to_plot,
               show_opts=show_options,
               xmin=-600, xmax=600,
               zmin=-100, zmax=900,
               show_internal_field=True)
```

For a full list of possible settings, see the online documentation. Note: Spheres allow the evaluation of near fields everywhere (inside and outside the particles). Non-spherical particles allow the evaluation only outside the particles. The computed near fields inside the circumscribing sphere of non-spherical particles are in general not correct.

¹The term “near field” is opposed to “far field” which is an intensity distribution in direction space. It does not imply that the field is evaluated near the particles.

Far fields. A far field is an intensity distribution $I(\alpha, \beta)$ in direction space (i.e., power per solid angle, measured far away from the scattering centers), such that the total power is:

$$W = \int_0^{2\pi} \int_0^\pi I(\alpha, \beta) \sin \beta \, d\beta \, d\alpha \quad (9)$$

The following code snippet illustrates several convenience functions to visualize the far field:

```
from smuthi.postprocessing.graphical_output \
    import show_scattered_far_field, \
           show_total_far_field

show_scattered_far_field(simulation=simul)

show_total_far_field(simulation=simul)
```

Again, for a full list of possible settings, see the online documentation.

Cross sections. If the initial field is a plane wave, the total or differential scattering cross section as well as the extinction cross section can be evaluated (see section 4.7).

```
from smuthi.postprocessing.graphical_output \
    import show_scattering_cross_section
from smuthi.postprocessing.far_field \
    import total_scattering_cross_section, \
           extinction_cross_section

sc = total_scattering_cross_section(simulation=simul)

ec = extinction_cross_section(simulation=simul)

# differential cross section
show_scattering_cross_section(simulation=simul)
```

Purcell factor: For simulations with a dipole source, the total radiated power can be evaluated with the following commands:

```
# after the simulation has been run
p0 = dip.dissipated_power_homogeneous_background(
    layer_system=simul.layer_system)

p = dip.dissipated_power(
    particle_list=simul.particle_list,
    layer_system=simul.layer_system)

purcell = p / p0
```

The Purcell factor [40] is then defined as the power radiated into the system by the dipole, divided by the power that it would radiate in a homogeneous medium.

If the user needs post-processing that goes beyond the described functionality, we recommend to browse through the online API documentation of the `postprocessing` package or directly through the source code and construct your own post-processing routine from the provided data structure.

4.4. Numerical parameters

In addition to the parameters that define the optical system, several numerical parameters can be used to control the accuracy (and runtime) of a simulation.

4.4.1. Multipole cut-off

The scattering properties of each particle are represented by its T-matrix $T_{\tau lm, \tau' l' m'}$, where τlm and $\tau' l' m'$ are the multipole polarization, degree and order of the scattered and incoming field, respectively. In practice, the T-matrix is truncated at some multipole degree $l_{\max} \geq 1$ and order $0 \leq m_{\max} \leq l_{\max}$ to obtain a finite system of linear equations.

The user can specify the cut-off parameters for each particle like this:

```
large_sphere = Sphere( ...
    l_max=10,
    m_max=10,
    ...)

small_sphere = Sphere( ...
    l_max=3,
    m_max=3,
    ...)
```

In general, we can say:

- Large particles require higher multipole orders than small particles.
- Particles very close to each other, very close to an interface or very close to a point dipole source require higher multipole orders than those that stand freely.
- Larger multipole cutoff parameters imply better accuracy, but also a quickly growing numerical effort.
- Setting a too large multipole cutoff order can lead to numerical instabilities.
- When simulating flat particles near planar interfaces, the multipole truncation should be chosen with regard to the Sommerfeld integral truncation, compare [38].

Several rules of thumb can be found in the literature for the selection of the multipole truncation in the case of spherical particles, see for example [41, 42]. Their applicability to the case of particles near planar interfaces, however, is typically not guaranteed. For this reason, SMUTHI offers a built-in automatic parameter selection feature to estimate a suitable multipole truncation, see 4.5.

4.4.2. Integral contours

Several steps during a SMUTHI simulation involve the numerical evaluation of integrals over the in-plane wavenumber of a plane wave expansion

$$\int_0^\infty f(\kappa) \, d\kappa, \quad (10)$$

where $\kappa = \sqrt{k_x^2 + k_y^2}$. To manage these integrals, we often refer to the dimensionless *effective refractive index*

$$n_{\text{eff}} = \frac{\kappa \lambda}{2\pi}, \quad (11)$$

where λ is the vacuum wavelength.

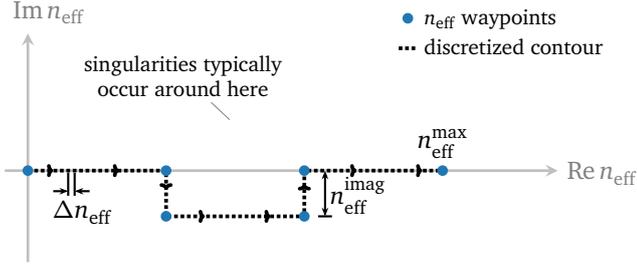


Figure 4: Parameters to control a deflected contour

In order to avoid numerical instability due to waveguide-mode or branchpoint singularities, it is favorable to replace the integral (10) along the real axis by a complex path $C : [0, 1] \rightarrow \mathbb{C}$ that is slightly deflected into the negative imaginary (see for example section 2.7.3 of [43]):

$$\int_0^\infty f(\kappa) d\kappa \longrightarrow \int_C f(\kappa) d\kappa, \quad (12)$$

The following parameters can be used to define a deflected contour (compare figure 4):

- The truncation point $n_{\text{eff}}^{\text{max}}$. In a layer with refractive index n , all partial plane waves with $n_{\text{eff}} < n$ are propagating and all waves with $n_{\text{eff}} > n$ are evanescent. The parameter $n_{\text{eff}}^{\text{max}}$ therefore defines how deep into the evanescent region the field expansion should be considered. As a rule of thumb, $n_{\text{eff}}^{\text{max}}$ should be chosen well above the largest occurring refractive index, e.g., $n_{\text{eff}}^{\text{max}} = n + 1.2$, which is the default. Note that a dipole source or a small particle very close to a layer interface might require a larger $n_{\text{eff}}^{\text{max}}$.
- The deflection into the imaginary $n_{\text{eff}}^{\text{imag}}$. If small, a fine discretization Δn_{eff} might be required because the integrand can then show rapid variations near waveguide mode or branchpoint singularities. However, a too large deflection into the imaginary can cause numerical problems, especially if particles are separated by a very large lateral distance. In that case, it is recommended to keep $n_{\text{eff}}^{\text{imag}} < 2/(k\rho_{\text{max}})$, where $k = 2\pi/\lambda$ is the vacuum wavenumber and ρ_{max} is the largest lateral separation between two particles, see section 4.4.3.
- The discretization along the path, Δn_{eff} . If a small value is chosen, the integral is more accurate, but the runtime grows. Note that for large lateral interparticle distances, Δn_{eff} must be chosen small in order to avoid aliasing. Again, it is recommended to keep $\Delta n_{\text{eff}} < 2/(k\rho_{\text{max}})$, see section 4.4.3.

Default contours. The simplest and recommended way to manage integral contours is through the globally defined *default contours*. Two default contours exist: a default Sommerfeld integral contour (which is by default used to

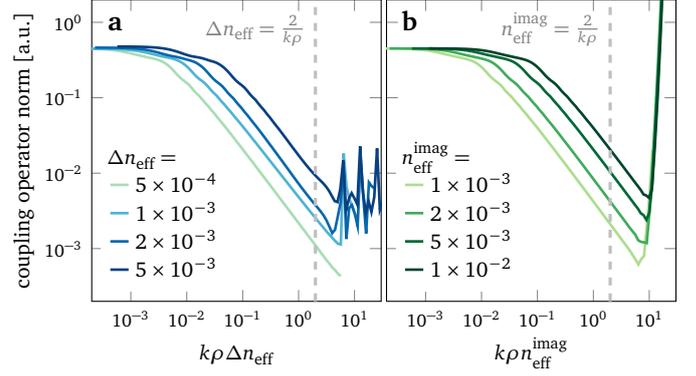


Figure 5: Aliasing and related issues for distant particles. a) Calculated layer system mediated particle coupling operator as a function of dimensionless lateral particle separation $k\rho$ scaled by Δn_{eff} for various values of Δn_{eff} . b) The same for $n_{\text{eff}}^{\text{imag}}$ instead of Δn_{eff} . See text for details.

compute the layer system mediated scattered field) and a default initial field integral contour (which is by default used to evaluate the initial field in case of dipole or Gaussian beam excitation). These contours are automatically set in the beginning of a simulation run, unless the `overwrite_default_contours` input argument of the simulation class is set to `False`. That way, the input arguments of the `Simulation` constructor can be used to set the contour parameters. Specifying, for instance:

```
from smuthi.simulation import Simulation

simul = Simulation( ...
    k_parallel="default",
    neff_max=2.5,
    neff_imag=5e-3,
    neff_resolution=2e-3,
    ... )
```

forces the setting of default contours using the specified parameters at the beginning of the simulation run. If you don't specify these arguments, the default settings of the `Simulation` class are applied. They are chosen such that for many application scenarios a reasonable integral contour is created.

Overriding the default contour. Every object in a SMUTHI simulation that defines integrals of type (10) has a `k_parallel` or `k_parallel_array` argument in its constructor. If an array of κ values is specified, that array is used instead of the default contour.

```
from smuthi.fields \
    import reasonable_Sommerfeld_kpar_contour
from smuthi.initial_field import DipoleSource

# define suitable k_parallel array
kp_ar = reasonable_Sommerfeld_kpar_contour(
    vacuum_wavelength=0.5,
    layer_refractive_indices=[1.52, 1],
    neff_imag=1e-3,
    neff_max=5,
    neff_resolution=1e-3)

# apply to specific dipole object
dip = DipoleSource( ...
    k_parallel_array=kp_ar
    ... )
```

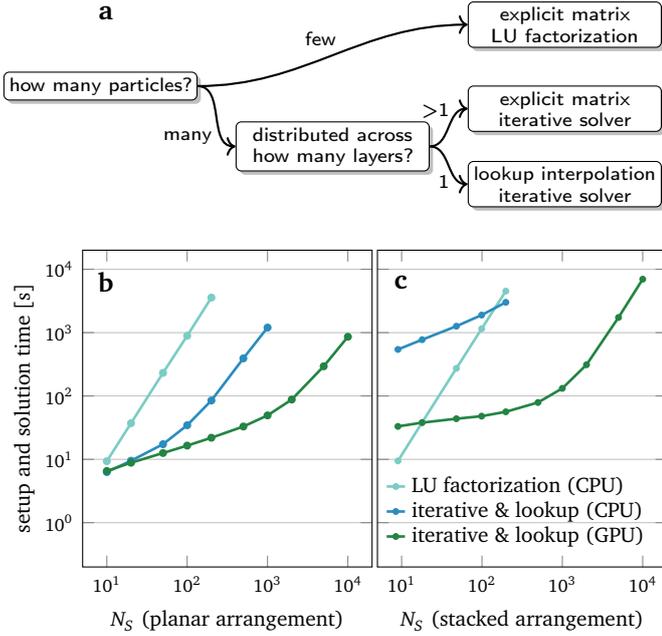


Figure 6: a) Suggested solver strategy among those available in SMUTHI. b-c) Linear system setup and solution time as a function of particle number for identical glass spheres on a glass substrate. We compare direct inversion on CPU with iterative solution on CPU and on GPU. The data for iterative solutions were generated in combination with lookup interpolation. Creating and querying the lookup table is faster if all particles share the same z coordinate (planar arrangement, panel b) as compared to arrangements where the particles are distributed over a span of heights (stacked arrangement, panel c). The actual runtime duration depends on the numerical and solver settings, as well as on the computer hardware. Plotted data refer to simulations with $l_{\max} = m_{\max} = 3$, run on a Google Colab [34] runtime equipped with an Intel Xeon CPU@2.30 GHz and a Tesla T4-16GB GPU.

It is advisable to override the default contour if different objects have a different convergence behavior with regard to the contour parameters. For example, if a dipole source is placed very close to an interface, a very large n_{eff}^{\max} might be required for convergence. However, when setting the default contours with such a large n_{eff}^{\max} value, all Sommerfeld integrals (in particular the ones that govern the particle multiple scattering) will be evaluated with that setting. Then it is better to set a moderate n_{eff}^{\max} for the default contour and override the contour in the dipole constructor separately.

4.4.3. Distant particles: Aliasing and related problems

Mathematically, Sommerfeld integrals are Hankel transforms, where the in-plane wavenumber $\kappa = 2\pi/\lambda n_{\text{eff}}$ and the lateral separation $\rho = \sqrt{\Delta x^2 + \Delta y^2}$ constitute a transform pair. As a consequence, a large lateral separation implies the need for a fine resolution of the integrand in numerical quadrature – otherwise aliasing errors are encountered.

This issue is illustrated in figure 5a, where the layer mediated particle coupling strength $W_j^{i,R}$ for two particles above a glass substrate is shown as a function of dimensionless

distance. Each curve refers to a different integrand sampling resolution Δn_{eff} , with the distance scaled by Δn_{eff} to compare the different curves directly. For this figure, the deflection into the imaginary was fixed to $n_{\text{eff}}^{\text{imag}} = 2 \times 10^{-4}$.

With growing distance, the coupling strength drops, because the electromagnetic interaction becomes weaker. However, for a certain large distance, the monotonous decay breaks and the onset of a noisy behavior marks the point where aliasing becomes relevant. We can identify $\Delta n_{\text{eff}} < 2/(k\rho)$ as a threshold, below which aliasing is not to be expected.

A similar issue affects the choice of the contour deflection into the imaginary, $n_{\text{eff}}^{\text{imag}}$, see 5b. Here, Δn_{eff} is fixed to 10^{-4} . The reason for the numerical errors at large distances is that the quadrature of the Sommerfeld integrals becomes numerically unstable if the argument $k\rho n_{\text{eff}}$ of the involved Bessel functions has a too large negative imaginary part. Again, a threshold of $n_{\text{eff}}^{\text{imag}} < 2/(k\rho)$ can be used to prevent aliasing.

4.4.4. Angular resolution

When the far field quantities are evaluated in direction space, the angular resolution determines the accuracy of integrated power quantities, like the total power flux or the total scattering cross section, compare (9). Depending on the size and configuration of scattering particles, the scattered intensity as a function of direction can have narrow features (e.g., a sharp peak in forward direction for large particles). Then, a fine angular resolution might be necessary to obtain correct integral results.

The default angular resolution can be set as an input argument of the Simulation object:

```

from smuthi.simulation import Simulation
import numpy as np

simul = Simulation( ...
                  angular_resolution=np.pi / 360
                  ...)
  
```

Note that the angular resolution only impacts the post processing, but has no influence on the steps performed during the Simulation.run() command.

4.4.5. Solver settings

In order to allow runtime-efficient simulations, SMUTHI currently offers two numerical strategies for the solution of the scattering problem:

1. LU factorization: To this end, the interaction matrix $M_{j,nn'}^i$ (see (8)) is fully stored in memory.
2. Iterative solution with the GMRES, LGMRES or GCROTMK method. In this case, you can either store the full interaction matrix in memory, or use a lookup from which the matrix entries are approximated by interpolation, see section 3.10.1 of [33] or [44].

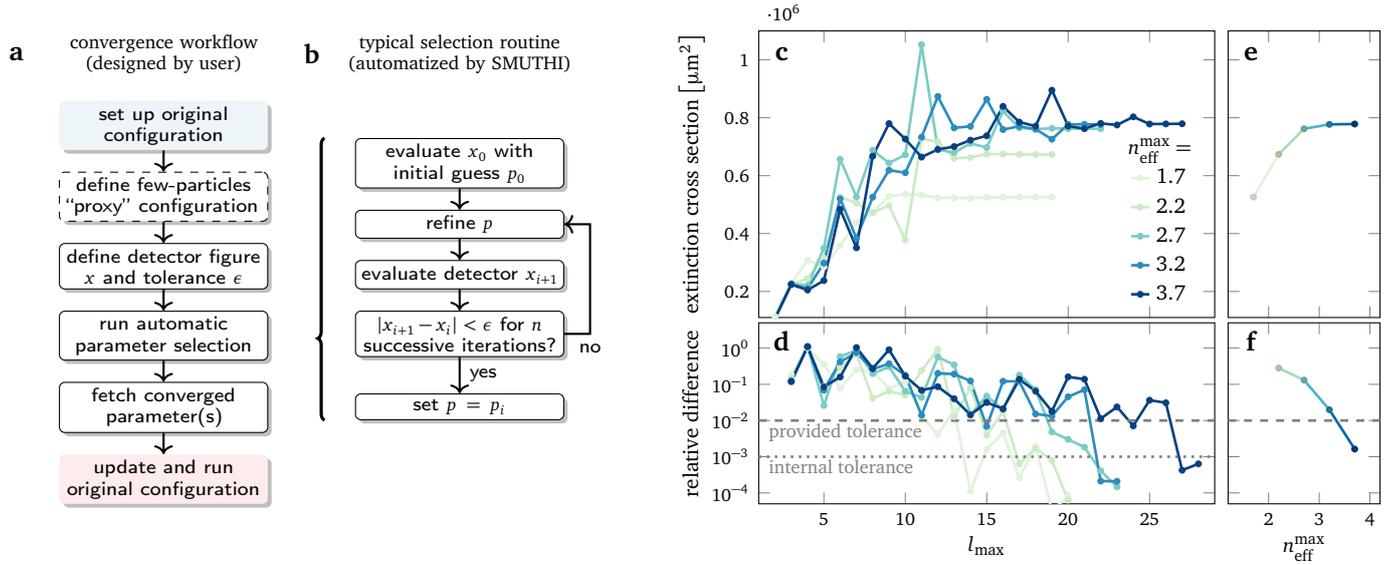


Figure 7: Overview of the automatic parameter selection process. a) Flowchart representation of the steps that the user takes in order to use the automatic parameter selection. The second step (definition of proxy-simulation) is optional, and only required if the original configuration involves a long run time, see section 4.5.1. b) Flowchart of the typical algorithm followed to select a single numerical parameter. For further details, see the online documentation. c-f) Illustrative data generated during the selection of the appropriate $n_{\text{eff}}^{\text{max}}$ value for a particle on a substrate using the extinction cross section as the detector figure x and a tolerance of $\epsilon = 10^{-2}$.

With growing particle number, all involved operations get more expensive, but the cost of LU factorization grows faster than the cost of iterative solution. Similarly, the calculation cost of the full interaction matrix grows faster than the cost of computing a lookup table. For this reason, we recommend the decision scheme depicted in figure 6a. Figure 6b and c illustrates the typical scaling of solver time as a function of particle number for different solver settings and hardware setups. It is worth noting that the lookup interpolation approach is even more efficient for configurations where all particles are located at the same height, i.e., they share the same z -coordinate (figure 6b) as compared to the general case where particles are distributed over a span of heights (6c). The plotted data also illustrates that for small particle numbers, the preparation of the lookup tables limits the runtime, whereas for large particle numbers, the iterative solution becomes the computational bottleneck. The cross-over between these regimes is marked by a change in the double-logarithmic slope of the corresponding graphs.

In a SMUTHI script, the numerical strategy for solving the linear system is defined through the input parameters of the simulation constructor. The relevant parameters are:

1. `solver_type`: At the moment, "LU" (default), "GMRES", "LGMRES" or "GCROTMK" are available.
2. `solver_tolerance`: Convergence criterion (for iterative solver only, default 10^{-4}).
3. `store_coupling_matrix`: If true (default), the coupling matrix is explicitly calculated and stored in memory. Otherwise, a lookup table is prepared and the matrix-vector multiplications are run on the fly, where the matrix entries are computed using the

lookup table. Note that lookup table interpolation is currently only available if all particles are in the same layer. The parameter is ignored in case of LU solver type.

4. `coupling_matrix_lookup_resolution`: In case lookup tables are used, this sets the sampling step distance (in length units). The parameter is ignored when the coupling matrix is explicitly calculated.
5. `coupling_matrix_interpolator_kind`: In case lookup tables are used, switch between "linear" or "cubic" (default) interpolation. "linear" is faster and "cubic" is more precise on equal resolution. The parameter is ignored when the coupling matrix is explicitly calculated.

This would be a typical setting for a small number of particles:

```
from smuthi.simulation import Simulation

simul = Simulation( ...
    solver_type="LU",
    store_coupling_matrix=True,
    ... )
```

This would be a typical setting for a large number of particles:

```
from smuthi.simulation import Simulation

simul = Simulation( ...
    solver_type="GMRES",
    solver_tolerance=1e-3,
    store_coupling_matrix=False,
    coupling_matrix_lookup_resolution=5,
    coupling_matrix_interpolator_kind="linear",
    ... )
```

Note that GPU acceleration is currently only available for particle coupling through lookup interpolation.

4.5. Automatic parameter selection

Smuthi offers a module to run an automated convergence test for the following parameters:

- Multipole truncation parameters l_{\max} and m_{\max} for each particle
- Integral contour parameters n_{eff}^{\max} and Δn_{eff}
- The angular resolution $\Delta\beta$, $\Delta\alpha$ of far field data

The user provides: a simulation object, a detector function and a relative tolerance. The detector function maps a simulation object (that has already been run) to one numeric figure (the detector quantity). In other words, the detector function does some post-processing to yield a scalar value that we use to monitor convergence. The following pre-defined strings can be passed as arguments for convenience: "extinction cross section", "total scattering cross section" or "integrated scattered far field" to set the corresponding figure as the detector quantity. Other possible detector functions could map to the norm of the electric field at a certain point, or the scattered far field in a certain direction or whatever appears to be a suitable measure for the convergence of the simulation.

The automatic parameter selection routine then repeatedly runs the simulation and evaluates the detector quantity with subsequently modified numerical input parameters until the relative deviation of the detector quantity is less than the specified tolerance, see figure 7.

After termination, the simulation object is updated with the so determined convergence settings. The following code example illustrates a possible use of the feature:

```
from smuthi.utility.automatic_parameter_selection
import select_numerical_parameters

select_numerical_parameters(simulation=simul,
                           tolerance=1e-3)
```

Some aspects need to be taken into account when using the automatic parameter selection:

- Both the multiple scattering and the initial field default contour are updated with the same parameters. A separate optimization of the parameters for initial field and multiple scattering is currently not supported.
- The algorithm compares the detector value for subsequent simulation runs. The idea is that if the simulation results agree for different numerical input parameters, they have probably converged with regard to that parameter. However, in certain cases this assumption can be false, i.e., the simulation results

agree although they have not converged. The automatic parameter selection therefore does not replace critical judging of the results by the user.

- With the parameter `tolerance_steps`, the user can ask that the tolerance criterion is met multiple times in a row before the routine terminates.
- The simulation is repeated multiple times, such that the automatic parameter selection takes much more time than a single simulation.
- For flat particles on a substrate, it is recommended to provide `relative_convergence=True` as an input argument to the `select_numerical_parameters` method. This triggers a separate convergence run for the multipole cutoff parameters during each iteration of the n_{eff}^{\max} parameter to account for the cross-dependent convergence behavior of these two sets of parameters, see [37, 38].

For further details, see the online documentation.

4.5.1. Automatic parameter selection for simulations with many particles

A simulation with many particles can be busy for a considerable runtime. The above described automatic procedure might then be unpractical. In this case, we recommend to define a “proxy configuration”. The idea is to find a system that takes less time to simulate but that has similar requirements with regard to numerical parameters.

Let us for example assume that we want to simulate light scattering by one thousand identical flat nano-cylinders located on a thin film system covering a substrate. Then, the selection of n_{eff}^{\max} needs to be done with regard to the distance of the particles to the next planar interface, whereas l_{\max} and m_{\max} have to be chosen with regard to the particle geometry, material, and to the selected n_{eff}^{\max} . Finally, Δn_{eff} needs to be chosen with regard to the layer system response. All of these characteristics have nothing to do with the fact that we are interested in a many particles system². We can thus simulate scattering by a single cylinder (or, to be on the safe side, by two neighboring cylinders) on the thin film system and let the automatic parameter selection module determine suitable values for l_{\max} , m_{\max} , n_{eff}^{\max} and Δn_{eff} . These parameters are then used as input parameters for the 1000-particles simulation which we run without another call to the automatic parameter selection module.

4.6. Physical units

SMUTHI is committed to a “relative-units” philosophy. That means, all quantities have only relative meaning.

²One does, however, have to consider the implications of large lateral inter-particle distances on the choice of Δn_{eff} and $n_{\text{eff}}^{\text{imag}}$, see section 4.4.3

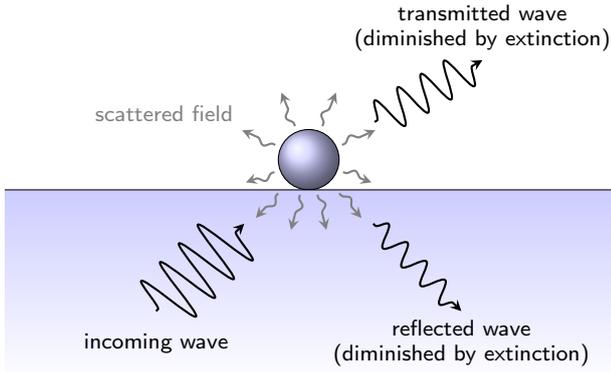


Figure 8: Concept of extinction in the presence of interfaces (compare [33]).

Length units. The user must consistently provide all length parameters (particle sizes, layer thicknesses, wavelengths, lookup resolutions, etc.) using a length unit of choice. Results will refer to that same unit. For example, if you specify the wavelength in nanometers, resulting cross sections will be in nm^2 . Quantities with an inverse length dimension (wavenumbers) also implicitly refer to the selected length unit.

Field strength units. When the electromagnetic fields are computed, their absolute value has no physical meaning. Only relative quantities can be used for further analysis. For example, the scattered field strength divided by the amplitude of the initial field does have a physical meaning.

Power units. Also power units have no meaning as absolute values. To get meaningful information, power-related figures always need to be regarded with reference to other power-related figures. Some examples:

- Scattering cross section as the quotient of scattered (angular) intensity and incident (power per area) intensity.
- Diffuse reflectivity as the total back scattered far field power divided by the initial Gaussian beam power.
- Purcell factor as the dissipated power of a dipole source divided by the dissipated power of the same source in an infinite homogeneous medium (i.e., in the absence of planar interfaces and scattering particles).

4.7. Cross sections

If the initial excitation is given by a plane wave, it is natural to discuss the far field properties of a scattering structure in terms of cross sections. However, in the context of scattering particles near planar interfaces, the commonly used concepts of cross sections need further clarification. In the following, we therefore discuss the meaning of cross sections as they are implemented in SMUTHI.

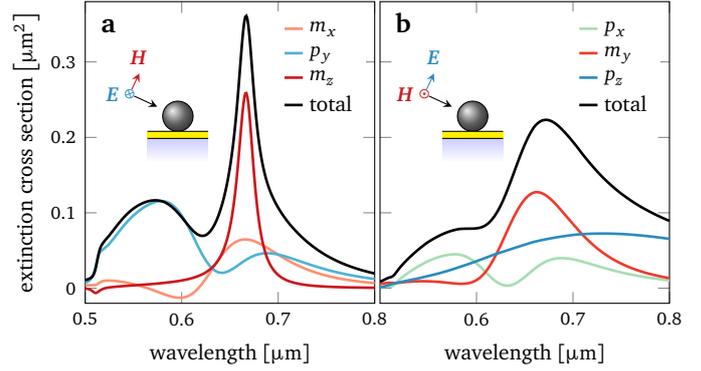


Figure 9: Multipole decomposition of the extinction cross section for a silicon sphere on a gold-coated glass substrate, illuminated by a) a TE-polarized and b) a TM-polarized wave propagating at an angle of 65° . Compare figure 3(b,d) of [45]. In the legend, p_x, p_y, p_z (m_x, m_y, m_z) refer to the Cartesian components of the electric (magnetic) dipole contributions.

4.7.1. Scattering cross section

The concept of a scattering cross section is straightforward: The incoming wave is specified by an intensity (power per area), whereas the scattered field is characterized by a power, such that the scattered signal divided by the initial signal yields an area.

The total scattering cross section reads

$$C_{\text{scat}} = W_{\text{scat}} I_{\text{init}} \quad (13)$$

where W_{scat} is the total scattered power and I_{init} is the incident irradiance (power per unit area perpendicular to the direction of propagation).

4.7.2. Extinction cross section

The term “extinction” means that particles take away power from the incident plane wave, such that they partially extinguish the incident wave. The power that they take away from the incoming wave is either absorbed or scattered into other channels, such that in the context of scattering of a plane wave by particles in a homogeneous medium, the extinction cross section is usually defined as the sum of the total scattering cross section and the absorption cross section.

However, this interpretation of extinction (i.e., the sum of particle absorption plus scattering) is not straightforward when a planarly layered medium is also involved. The reason is that besides particle absorption and scattering, also absorption and waveguiding in the layered medium has to be taken into account.

For this reason, we apply what is usually referred to as the optical theorem to define extinction (see section 3.8.1 of [33] for the mathematical details), by interpreting the term “extinction” strictly as a measure for *how much power is taken away by the particles from the incident plane wave*.

In fact, SMUTHI internally computes two extinction cross sections: one for the reflected incoming wave and one for

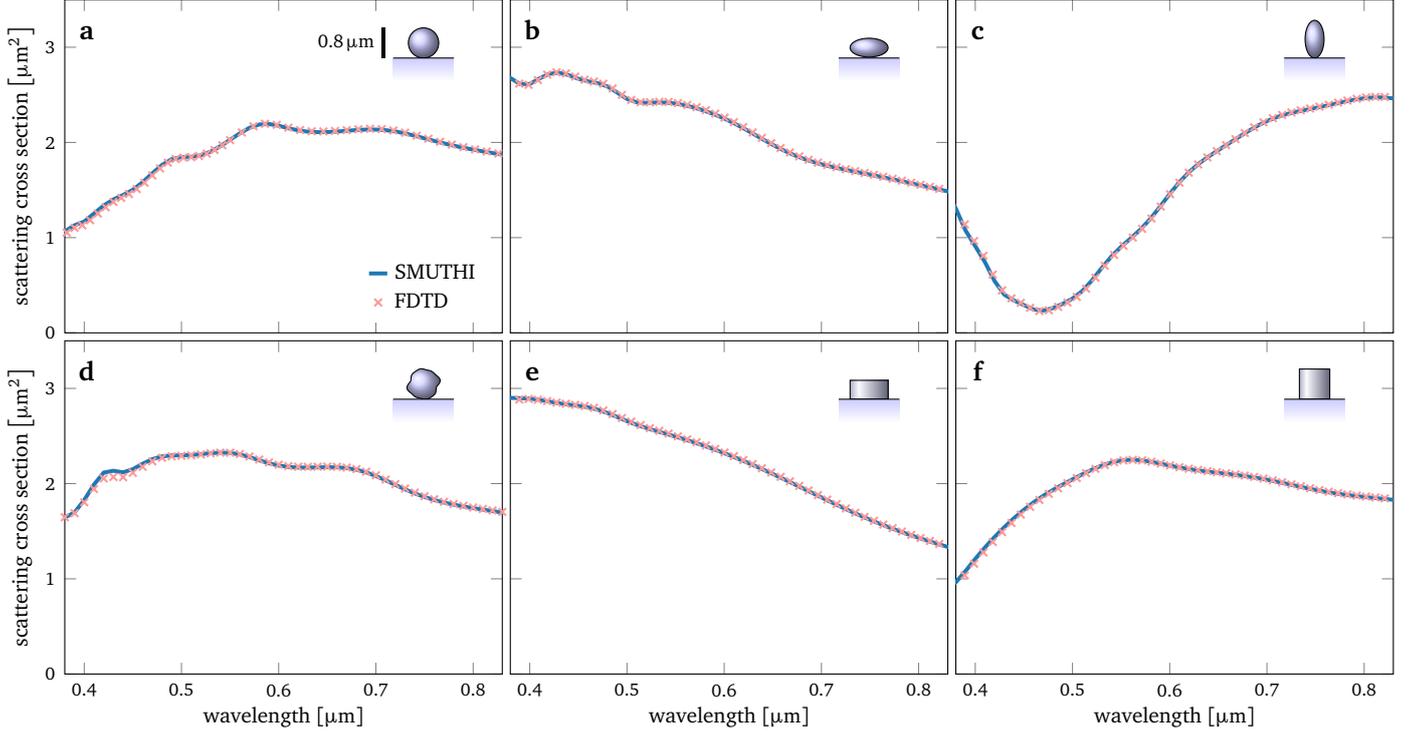


Figure 10: Scattering cross section spectra for a single glass particle on a glass substrate: a) sphere, b) prolate spheroid, c) oblate spheroid, d) free-form particle, e) flat cylinder, f) cylinder with unit aspect ratio. All particles have the same volume.

Table 1: Description of geometric and simulation parameters used to generate the data of figure 10. Particle parameters p_i correspond to the radius for the `Sphere`, the two semi-axes for the `Spheroid`, and the radius and height for the `FiniteCylinder`, respectively.

case	initial field			particle parameters					layer system				simulation parameters			
	type	θ [rad]	pol	type	p_1 [μm]	p_2 [μm]	z [μm]	n	t_1 [μm]	t_2 [μm]	n_1	n_2	l_{max}	m_{max}	$n_{\text{eff}}^{\text{max}}$	Δn_{eff}
a	PlaneWave	π	TE	Sphere	0.4	—	0.4	1.52	∞	∞	1.0	1.52	9	1	2.22	0.01
b	PlaneWave	π	TE	Spheroid	0.504	0.252	0.252	1.52	∞	∞	1.0	1.52	23	1	2.22	0.01
c	PlaneWave	π	TE	Spheroid	0.317	0.635	0.635	1.52	∞	∞	1.0	1.52	13	1	2.22	0.01
d	PlaneWave	π	TE	CustomParticle	—	—	0.326	1.52	∞	∞	1.0	1.52	10	3	2.42	0.01
e	PlaneWave	π	TE	FiniteCylinder	0.44	0.44	0.22	1.52	∞	∞	1.0	1.52	27	1	2.22	0.01
f	PlaneWave	π	TE	FiniteCylinder	0.35	0.7	0.35	1.52	∞	∞	1.0	1.52	19	1	2.22	0.01

the transmitted incoming wave, see figure 8. That means, the extinction cross section for reflection (transmission) refers to the destructive interference of the scattered signal with the specular reflection (transmission) of the initial wave. It thereby includes absorption in the particles, scattering, and a modified absorption by the layer system, e.g., through coupling into guided modes.

As a consequence, the extinction cross sections can be negative if (for example due to a modified absorption in the layer system) more light is reflected (or transmitted) in the specular direction than would be without the particles.

Conservation of energy is then expressed by the following statement: “For lossless particles near or inside a lossless planarly layered medium (that doesn’t support any waveguide modes), the sum of top and bottom extinction cross section equals the total scattering cross section”.

4.7.3. Multipole decomposition

As SMUTHI builds on the expansion of the scattered field in spherical vector wave functions, it naturally lends itself to a multipole analysis of the extinction cross section³, see figure 9. A convenience function to select the contribution of individual (spherical) multipole moments to the evaluation of the extinction cross section is built into the `extinction_cross_section` method of the `smuthi.postprocessing.far_field` module. See the online documentation for further details.

³A similar decomposition of the scattering cross section is in general not reasonable, because the scattered field intensity is not a linear function of the scattered field coefficients b_n^i , compare (3). For multiple particles or particles near planar interfaces, the contributions of individual multipole moments to the scattered field intensity is therefore not additive.

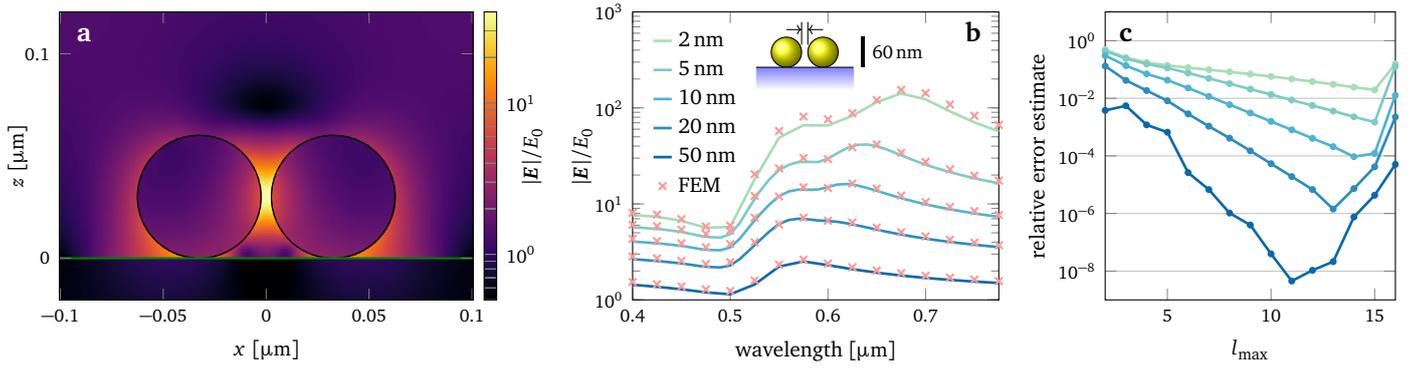


Figure 11: Gold dimer on a silicon substrate. a) Electric field distribution obtained for a plane wave illumination from above. The gap distance is 5 nm. b) Electric field in the middle of the gap as a function of wavelength for various gap distances. c) Error estimate of the electric field in the gap as a function of multipole degree truncation for various gap distances.

4.8. Restrictions

The following restrictions limit the range of applications for SMUTHI:

- All media are linear, homogeneous, isotropic and non-magnetic.
- Particles must not intersect with each other or with layer interfaces.
- The electric field inside the circumscribing sphere of a particle (i.e., the smallest sphere around a particle center that includes the particle volume) cannot be computed.
- The software was designed for particles with diameters in the range of up to a few wavelengths. Its overall numerical stability and validity has not been tested for larger particles.
- Closely adjacent non-spherical particles with intersecting circumscribing spheres can lead to incorrect results. The use of SMUTHI is therefore limited to geometries with particles that have disjoint circumscribing spheres. We note that by the use of regularized particle coupling approaches, this limitation can be relaxed [39, 46].
- Dipole sources must not be placed inside the circumscribing sphere of a particle.
- If oblate particles are located near interfaces, such that the circumscribing sphere of the particle intersects the interface, a correct simulation result can in principle be achieved [37]. However, special care has to be taken [38].
- The software was designed for thin-film systems with layer thicknesses of up to a few wavelengths. Simulations involving thick layers might fail or return wrong results due to numerical instability.

In general, SMUTHI does not provide error checking of user input, nor does it check if the numerical parameters

specified by the user are sufficient for accurate simulation results or if the specified model falls into the scope of simulation scenes for which SMUTHI can compute valid results. It is thus required that the user develops some understanding of the influence of various numerical parameters on the validity of the results and also for the limits of SMUTHI's capabilities.

5. Use case examples

In the following, we present a number of representative example configurations that can be studied with SMUTHI. For selected examples, we will also compare our results to accurate third-party benchmark simulations or results published in literature. Scripts to reproduce the results can be downloaded from the examples section in SMUTHI's online git repository.

5.1. Single particle on glass substrate

A typical use case is the scattering of a plane wave by a single particle on a substrate. In this application example, we investigate the scattering and extinction cross section as a function of wavelength for dielectric particles ($n = 1.52$) of different shapes on a dielectric substrate ($n = 1.52$). See the caption of figure 10 for further details. The particles are sized such that the equivalent volume radius is 400 nm. Results are compared to FDTD data obtained using Lumerical [47].

Numerical parameters used for SMUTHI simulations are determined by means of the automatic parameter selection feature, see section 4.5. To this end, we run the automatic parameter selection for the smallest wavelength of the considered spectral range and use the resulting numerical parameters for all wavelengths. The underlying assumption is that the smallest wavelength implies the largest particle size parameter, such that the corresponding numerical settings are a conservative choice for the other wavelengths, too. The automatic parameter selection is called with the

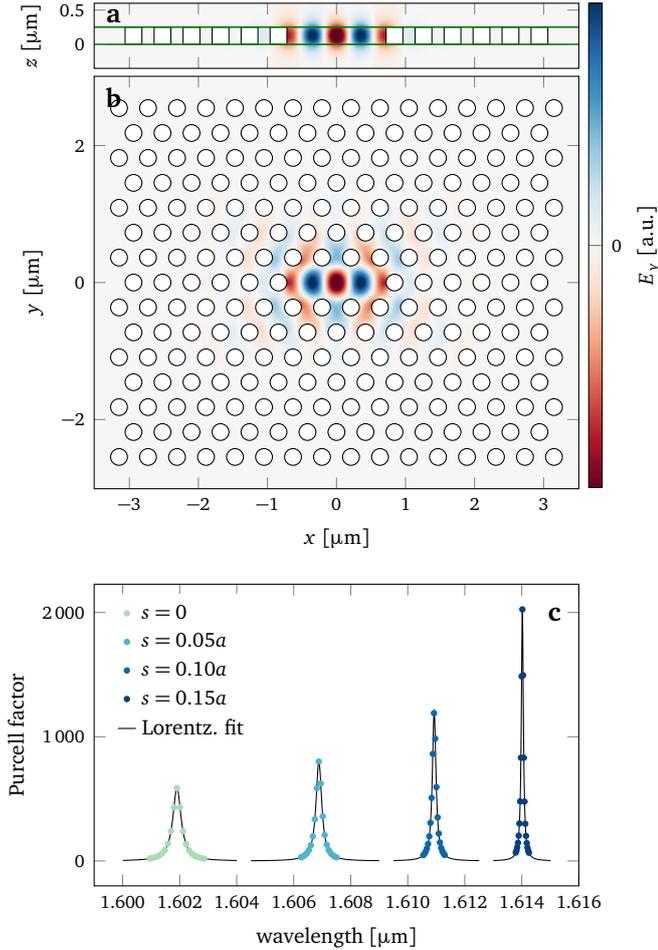


Figure 12: Field enhancement in a photonic crystal cavity. a-b) Electric field distribution for a y -aligned dipole source in a suspended photonic crystal membrane at resonance wavelength 1.602 μm). Plotting planes correspond to $y = 0 \mu\text{m}$ and $z = t/2$ where t is the slab thickness. c) Purcell factor as a function of wavelength, compare figure 4 of [48].

default relative accuracy tolerance of 10^{-3} . Figure 10 displays the spectral cross section for illumination by a plane wave under normal incidence.

The automatic parameter selection resulted in the settings as summarized in table 1. The low value returned for m_{max} is typical for single axisymmetric particles illuminated under normal incidence.

In all cases, the agreement to FDTD results is very good.

5.2. Field enhancement between two plasmonic nano spheres on a substrate

Electric field hot spots are important for non-linear applications such as Raman spectroscopy [49]. In this example we want to explore how SMUTHI can be used to analyze plasmonic structures with field enhancement. Two gold nanospheres (radius 30 nm) are placed in water ($n = 1.33$) on a silicon substrate. A gap of width d defines the distance between the particles, compare figure 3c of [49]. The

scene is illuminated by a plane wave under normal incidence from above, with a polarization such that the electric field is parallel to the distance between the particle centers. Figure 11a shows the norm of the resulting electric field with a logarithmic color map for a gap width of $d = 5 \text{ nm}$. Panel 11b shows the norm of the electric field in the middle of the gap compared against results obtained via a finite element method using COMSOL [50].

Figure 11c shows the relative difference of the calculated gap fields for subsequent values of the multipole degree truncation l_{max} . We interpret this difference as a measure for the relative error of the gap field at that multipole truncation. As expected, the error decreases with growing l_{max} . Configurations with a very narrow gap converge more slowly than configurations with a wider gap. At some threshold value, a numerical instability prevents further convergence, and the results become instead less accurate with growing l_{max} . As a consequence, a relative accuracy of 1% is achieved at $l_{\text{max}} = 2$ for $d = 50 \text{ nm}$, at $l_{\text{max}} = 5$ for $d = 20 \text{ nm}$, at $l_{\text{max}} = 8$ for $d = 10 \text{ nm}$, at $l_{\text{max}} = 11$ for $d = 5 \text{ nm}$ and never for gap widths $d \leq 2 \text{ nm}$.

5.3. Photonic crystal slab

In this application example, we reproduce the experimental findings presented in [48]. The geometry is given by a $0.25 \mu\text{m}$ thick silicon slab with cylindrical air holes of radius $0.12 \mu\text{m}$. The air holes are arranged in a hexagonal grid with lattice constant $a = 0.42 \mu\text{m}$. A defect of three missing air holes in a row acts as a photonic nano-cavity (known in the literature as the “L3 cavity”). By shifting the air holes next to the defect location by a distance s into the outward direction, the quality factor Q of the cavity is optimized with a maximum for $s = 0.15 a$ [48].

We have reproduced the configuration in SMUTHI, assuming a constant silicon refractive index of $n = 3.473$ [51] and modeling the cavity by a rectangular domain of air holes with a wall “thickness” of seven lattice units in each direction, see figure 12a-b, for a total of 230 air holes. The initial field is given by a point dipole source placed at the cavity center with a dipole moment aligned in the y -direction. In order to find suitable numerical settings, we used a proxy configuration (see section 4.5.1) including only two air holes and launched an automatic parameter selection. The resulting parameters ($l_{\text{max}} = 5$, $m_{\text{max}} = 2$, $n_{\text{eff}}^{\text{max}} = 5.67$) were then used for the full simulations of the photonic crystal slab.

For wavelengths close to a high- Q resonance (in particular, wavelengths close to the peak resonance of $1.1614 \mu\text{m}$ in the $s = 0.15 a$ case), the iterative solvers exhibited a slower convergence to the accurate solution – sometimes stagnating altogether or converging to unphysical results. In the most critical case, convergence was eventually achieved using the GCR0TMK solver [52] as implemented in the `scipy.sparse.linalg` library and relaxing the relative

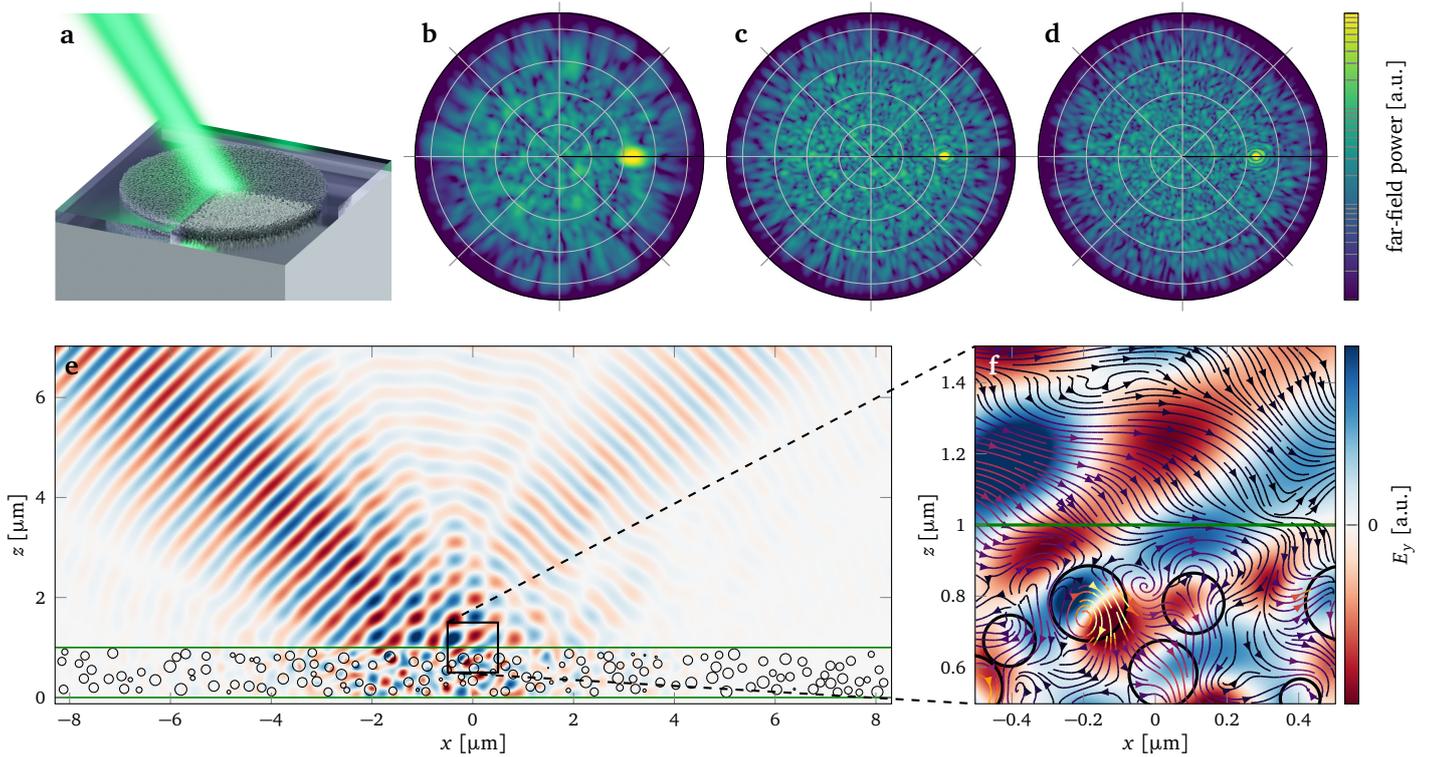


Figure 13: Tilted Gaussian beam impinging on a paint micro-layer over an iron substrate. a) Rendering of the tested configuration comprising 10^4 particles. b-d) Reflected far-field power for incident beam waist values of 4, 6 and $8\ \mu\text{m}$, respectively. The specular reflection is visible as a bright spot in the intensity distribution. e) E_y component of the electric field at the $y = 0\ \mu\text{m}$ plane. f) Detail of the near field distribution at the air-paint layer interface, with superimposed Poynting vector flux lines.

tolerance to 3×10^3 . Despite the difficulties to achieve convergence with low tolerance values, the resulting Purcell factors can be perfectly fitted by a Lorentzian lineshape, as expected.

The upper panels of figure 12 show different crosscuts of the simulated electric field distribution, which show excellent agreement with numerical results published in the literature for the L3 cavity [48]. The Purcell factor as a function of wavelength is reported in the lower panel. When going from shift parameter $s = 0$ to $s = 0.15a$ in steps of $0.05a$, the subsequently narrower resonances with higher amplitudes indicate a larger Q -factor of the cavity. At the same time, the resonance wavelength is shifted to longer wavelengths. These trends are again in good qualitative agreement with experimental data (compare figure 4 of [48]).

5.4. Beam reflection by paint micro layer on metal surface

A final example refers to a Gaussian beam that is incident on an iron surface covered with a scattering layer under an angle of 45° . The scattering layer has a thickness of $1\ \mu\text{m}$ and a refractive index of 1.52. A total of 10^4 poly-disperse spherical TiO_2 particles are randomly dispersed in the scattering layer with a volume density of 20% over an area of $\sim 200\ \mu\text{m}^2$ (see figure 13a). The sphere radii follow a Gaussian distribution around $r = 0.1\ \mu\text{m}$ with a

standard deviation of $\sigma_r = 0.01\ \mu\text{m}$. Panels b-d show the far field intensity distribution obtained for increasing beam waist values, exhibiting an intense reflection peak in the specular direction. Panels e-f show the electric field E_y at a cross-cut plane through the scattering layer, with a detail of the Poynting vector flux lines across the layer interface.

5.5. Further examples published in the literature

In addition to the presented use case examples, the interested reader can refer to the literature for additional applications where SMUTHI has been used during its developing stages. Published works cover various systems in physics and electrical engineering research, including the design of nanoparticle based spectrally selective reflector layers in color conversion films [53], scattering layers for light outcoupling from organic light emitting diodes [28], resonance analysis of spherical particles on a coated substrate [54], photon excitation from electron inelastic tunneling near a metallic or dielectric nano-sphere [55], the study of disordered meta surfaces [56, 57] and the design of plasmonic gratings for sensing applications [58].

6. Conclusions and outlook

In this paper, we have introduced a Python package for the simulation of electromagnetic scattering by multiple

objects near or inside a planarly layered medium. After a brief overview on the theoretical background, we have provided a short manual highlighting the code interface from a user perspective.

The accuracy of the software was validated by comparison to FDTD results for the illustrative use case of extinction by various particle shapes on a dielectric substrate.

Near field enhancement for a plasmonic system of two metal spheres at variable distance on a silicon substrate was also evaluated, showing good agreement against FEM results down to a gap size of 5 nm at optical wavelengths. We conclude that although the evaluation of the near field is limited to the domain outside the circumscribing sphere of a particle, SMUTHI can be potentially used for near field studies of selected plasmonic structures.

Further, we have studied the resonant behavior of a high-Q cavity in a silicon photonic crystal slab. To our knowledge, the simulation of photonic crystal slabs has not previously been demonstrated with the T-matrix method (related approaches that rely on the expansion of the scattered field in cylindrical waves rather than spherical waves have been published in [59, 60]). Due to its ability to handle large numbers of scatterers in contact with layer interfaces, we believe that SMUTHI is a particularly powerful tool for this class of applications, which includes relevant platforms such as perforated membranes as well as meta-surface layouts.

Finally, we demonstrated the use of SMUTHI for the simulation of disordered volumetric aggregates of scattering nanoparticles in a dielectric micro film, showing the feasibility of simulations comprising $\sim 10^4$ wavelength-scale particles in a layered medium. To the best of our knowledge, SMUTHI is the only currently available tool that allows addressing this class of problems at such a large scale.

We therefore believe that SMUTHI will allow to expand the complexity of configurations that can be modeled rigorously, enabling the study of aspects that are traditionally difficult to investigate, such as finite-size effects, collective resonances, fabrication defects, effective medium approximations, scaling of transport properties, multi-scale heterogeneity and aperiodic structures in general.

For the future, we plan two major additions to enhance the capability of SMUTHI for the particularly interesting use case of photonic meta surfaces. These additions will address the run time, limited by the solution of the linear system (7), as well as the important case of close particles with intersecting circumscribing spheres.

Finally, an extension of the simulation method to infinitely laterally periodic structures will be published in the near future.

7. Acknowledgments

We would like to thank the authors of the NFM-DS code (Adrian Doicu, Thomas Wriedt and Yuri Eremin) for allowing us to use their software in SMUTHI. Furthermore, we thank Håkan Johansson for extending his pywigxjpf code [32] to our needs. We acknowledge code additions and important user feedback from Giacomo Mazzamuto, Ilija Rasskazov, Fabio Mangini and Alan Zhan. LP acknowledges NVIDIA Corporation for the donation of a Titan Xp GPU through the GPU Grant program. KMC acknowledges support by the Polish National Science center via the project 2020/37/N/ST3/03334. DT acknowledges funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy via the Excellence Cluster 3D Matter Made to Order (EXC-2082/1 - 390761711) and the project GO 2615/2-1 (project number 410400458) within the DFG-SPP 1839 "Tailored Disorder". KL and ASK acknowledge the support of the Russian Science Foundation (Project 21-72-30018).

References

- [1] S. Jahani, Z. Jacob, All-dielectric metamaterials, *Nature nanotechnology* 11 (2016) 23–36.
- [2] I. Staude, J. Schilling, Metamaterial-inspired silicon nanophotonics, *Nature photonics* 11 (2017) 274–284.
- [3] D. C. Prieve, Measurement of colloidal forces with TIRM, *Advances in Colloid and Interface Science* 82 (1999) 93–125.
- [4] J. F. Li, Y. F. Huang, Y. Ding, Z. L. Yang, S. B. Li, X. S. Zhou, F. R. Fan, W. Zhang, Z. Y. Zhou, B. Ren, et al., Shell-isolated nanoparticle-enhanced Raman spectroscopy, *Nature* 464 (2010) 392–395.
- [5] M. L. Juan, M. Righini, R. Quidant, Plasmon nano-optical tweezers, *Nature photonics* 5 (2011) 349.
- [6] K. Saxena, V. Jain, D. S. Mehta, A review on the light extraction techniques in organic electroluminescent devices, *Optical Materials* 32 (2009) 221–233.
- [7] M. L. Brongersma, Y. Cui, S. Fan, Light management for photovoltaics using high-index nanostructures, *Nature materials* 13 (2014) 451–460.
- [8] T. Yoshie, A. Scherer, J. Hendrickson, G. Khitrova, H. Gibbs, G. Rupper, C. Ell, O. Shchekin, D. Deppe, Vacuum Rabi splitting with a single quantum dot in a photonic crystal nanocavity, *Nature* 432 (2004) 200–203.
- [9] A. I. Kuznetsov, A. E. Miroschnichenko, M. L. Brongersma, Y. S. Kivshar, B. Luk'yanchuk, Optically resonant dielectric nanostructures, *Science* 354 (2016).
- [10] V. E. Babicheva, A. B. Evlyukhin, Resonant lattice Kerker effect in metasurfaces with electric and magnetic optical responses, *Laser & Photonics Reviews* 11 (2017) 1700132.
- [11] F. Riboli, N. Caselli, S. Vignolini, F. Intonti, K. Vynck, P. Barthelemy, A. Gerardino, L. Balet, L. H. Li, A. Fiore, et al., Engineering of light confinement in strongly scattering disordered media, *Nature materials* 13 (2014) 720–725.
- [12] A. B. Khanikaev, S. H. Mousavi, W.-K. Tse, M. Kargarian, A. H. MacDonald, G. Shvets, Photonic topological insulators, *Nature materials* 12 (2013) 233–239.
- [13] L.-H. Wu, X. Hu, Scheme for achieving a topological photonic crystal by using dielectric material, *Physical Review Letters* 114 (2015) 223901.
- [14] C. W. Hsu, B. Zhen, J. Lee, S.-L. Chua, S. G. Johnson, J. D. Joannopoulos, M. Soljačić, Observation of trapped light within the radiation continuum, *Nature* 499 (2013) 188–191.
- [15] P. Waterman, Matrix formulation of electromagnetic scattering, *Proceedings of the IEEE* 53 (1965) 805–812.

- [16] B. Peterson, S. Ström, T-matrix for electromagnetic scattering from an arbitrary number of scatterers and representations of $E(3)$, *Physical Review D* 8 (1973) 3661.
- [17] D. W. Mackowski, M. I. Mishchenko, Calculation of the T-matrix and the scattering matrix for ensembles of spheres, *JOSA A* 13 (1996) 2266–2278.
- [18] D. Mackowski, M. Mishchenko, A multiple sphere T-matrix Fortran code for use on parallel computer clusters, *Journal of Quantitative Spectroscopy and Radiative Transfer* 112 (2011) 2182–2192.
- [19] J. Markkanen, A. J. Yuffa, Fast superposition T-matrix solution for clusters with arbitrarily-shaped constituent particles, *Journal of Quantitative Spectroscopy and Radiative Transfer* 189 (2017) 181–188.
- [20] A. Egel, L. Pattelli, G. Mazzamuto, D. S. Wiersma, U. Lemmer, CELES: CUDA-accelerated simulation of electromagnetic scattering by large ensembles of spheres, *Journal of Quantitative Spectroscopy and Radiative Transfer* 199 (2017) 103–110.
- [21] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, A. Fasih, PyCUDA and PyOpenCL: A scripting-based approach to GPU runtime code generation, *Parallel Computing* 38 (2012) 157–174.
- [22] M. Mishchenko, G. Videen, V. Babenko, N. Khlebtsov, T. Wriedt, T-matrix theory of electromagnetic scattering by particles and its applications: A comprehensive reference database, *Journal of Quantitative Spectroscopy and Radiative Transfer* 88 (2004) 357–406.
- [23] G. Kristensson, Electromagnetic scattering from buried inhomogeneities – a general three-dimensional formalism, *Journal of Applied Physics* 51 (1980) 3486–3500.
- [24] D. W. Mackowski, Exact solution for the scattering and absorption properties of sphere clusters on a plane surface, *Journal of Quantitative Spectroscopy and Radiative Transfer* 109 (2008) 770–788.
- [25] A. Egel, U. Lemmer, Dipole emission in stratified media with multiple spherical scatterers: Enhanced outcoupling from OLEDs, *Journal of Quantitative Spectroscopy and Radiative Transfer* 148 (2014) 165–176.
- [26] A. Doicu, T. Wriedt, Calculation of the T-matrix in the null-field method with discrete sources, *JOSA A* 16 (1999) 2539–2544.
- [27] A. Doicu, T. Wriedt, Y. A. Eremin, *Light scattering by systems of particles*, volume 124, Springer, 2006.
- [28] A. Egel, G. Gomard, S. W. Kettlitz, U. Lemmer, Accurate optical simulation of nano-particle based internal scattering layers for light outcoupling from organic light emitting diodes, *Journal of Optics* 19 (2017) 025605.
- [29] S. Stein, Addition theorems for spherical wave functions, *Quarterly of Applied Mathematics* 19 (1961) 15–24.
- [30] O. R. Cruzan, Translational addition theorems for spherical vector wave functions, *Quarterly of Applied Mathematics* 20 (1962) 33–40.
- [31] M. I. Mishchenko, L. D. Travis, A. A. Lacis, *Scattering, absorption, and emission of light by small particles*, Cambridge University Press, 2002.
- [32] H. T. Johansson, C. Forssén, Fast and accurate evaluation of Wigner $3j$, $6j$, and $9j$ symbols using prime factorization and multiword integer arithmetic, *SIAM Journal on Scientific Computing* 38 (2016) A376–A384.
- [33] A. Egel, Accurate optical simulation of disordered scattering layers for light extraction from organic light emitting diodes, Ph.D. thesis, Karlsruhe Institute of Technology, 2019.
- [34] Colab, Google Colaboratory, <https://colab.research.google.com>, 2021.
- [35] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, *International journal for numerical methods in engineering* 79 (2009) 1309–1331.
- [36] Dawson-Haggerty et al., trimesh, <https://trimsh.org/>, 2021.
- [37] A. Egel, D. Theobald, Y. Donie, U. Lemmer, G. Gomard, Light scattering by oblate particles near planar interfaces: on the validity of the T-matrix approach, *Optics Express* 24 (2016) 25154–25168.
- [38] A. Egel, Y. Eremin, T. Wriedt, D. Theobald, U. Lemmer, G. Gomard, Extending the applicability of the T-matrix method to light scattering by flat particles on a substrate via truncation of Sommerfeld integrals, *Journal of Quantitative Spectroscopy and Radiative Transfer* 202 (2017) 279–285.
- [39] D. Theobald, A. Egel, G. Gomard, U. Lemmer, Plane-wave coupling formalism for T-matrix simulations of light scattering by nonspherical particles, *Physical Review A* 96 (2017) 033822.
- [40] E. M. Purcell, Spontaneous emission probabilities at radio frequencies, in: *Confined Electrons and Photons*, Springer, 1995, pp. 839–839.
- [41] A. A. R. Neves, D. Pisignano, Effect of finite terms on the truncation error of Mie series, *Optics letters* 37 (2012) 2418–2420.
- [42] W. J. Wiscombe, Improved Mie scattering algorithms, *Applied optics* 19 (1980) 1505–1509.
- [43] W. C. Chew, *Waves and fields in inhomogeneous media*, IEEE press, 1995.
- [44] A. Egel, S. W. Kettlitz, U. Lemmer, Efficient evaluation of Sommerfeld integrals for the optical simulation of many scattering particles in planarly layered media, *JOSA A* 33 (2016) 698–706.
- [45] I. Sinev, I. Iorsh, A. Bogdanov, D. Permyakov, F. Komissarenko, I. Mukhin, A. Samusev, V. Valuckas, A. I. Kuznetsov, B. S. Luk'yanchuk, et al., Polarization control over electric and magnetic dipole resonances of dielectric nanoparticles on metallic films, *Laser & Photonics Reviews* 10 (2016) 799–806.
- [46] T. Martin, T-matrix method for closely adjacent obstacles, *Journal of Quantitative Spectroscopy and Radiative Transfer* 234 (2019) 40–46.
- [47] Lumerical, Lumerical Inc., <https://www.lumerical.com/>, 2021.
- [48] Y. Akahane, T. Asano, B.-S. Song, S. Noda, High-Q photonic nanocavity in a two-dimensional photonic crystal, *Nature* 425 (2003) 944–947.
- [49] S.-Y. Ding, J. Yi, J.-F. Li, B. Ren, D.-Y. Wu, R. Panneerselvam, Z.-Q. Tian, Nanostructure-based plasmon-enhanced Raman spectroscopy for surface analysis of materials, *Nature Reviews Materials* 1 (2016) 1–16.
- [50] COMSOL, Comsol Multiphysics, <https://www.comsol.com/>, 2021.
- [51] H. Li, Refractive index of silicon and germanium and its wavelength and temperature derivatives, *Journal of Physical and Chemical Reference Data* 9 (1980) 561–658.
- [52] E. De Sturler, Truncation strategies for optimal Krylov subspace methods, *SIAM Journal on Numerical Analysis* 36 (1999) 864–889.
- [53] D. Theobald, S. Yu, G. Gomard, U. Lemmer, Design of selective reflectors utilizing multiple scattering by core-shell nanoparticles for color conversion films, *ACS Photonics* 7 (2020) 1452–1460.
- [54] D. Pidgayko, Z. Sadrieva, K. Ladutenko, A. Bogdanov, Polarization-controlled selective excitation of Mie resonances in a dielectric nanoparticle on a coated substrate, *Physical Review B* 102 (2020) 245406.
- [55] L. Dvoretckaia, K. Ladutenko, A. Mozharov, G. Zograf, A. Bogdanov, I. Mukhin, Electrically driven metal and all-dielectric nanoantennas for plasmon polariton excitation, *Journal of Quantitative Spectroscopy and Radiative Transfer* 244 (2020) 106825.
- [56] K. M. Czajkowski, T. J. Antosiewicz, Electromagnetic coupling in optical devices based on random arrays of dielectric nanoresonators, *The Journal of Physical Chemistry C* 124 (2019) 896–905.
- [57] K. M. Czajkowski, M. Bancerek, T. J. Antosiewicz, Multipole analysis of substrate-supported dielectric nanoresonator metasurfaces via the T-matrix method, *Physical Review B* 102 (2020) 085431.
- [58] A. Warren, M. Alkai, C. Moore, Design of 2D plasmonic diffraction gratings for sensing and super-resolution imaging applications, in: *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, IEEE, pp. 1–6.
- [59] S. Boscolo, M. Midrio, Three-dimensional multiple-scattering technique for the analysis of photonic-crystal slabs, *Journal of lightwave technology* 22 (2004) 2778.
- [60] D. Pisssoort, E. Michielssen, D. V. Ginste, F. Olyslager, Fast-multipole analysis of electromagnetic scattering by photonic crystal slabs, *Journal of lightwave technology* 25 (2007) 2847–2863.