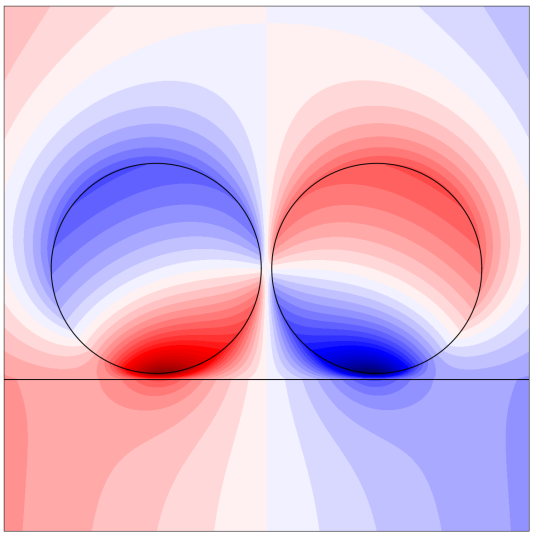
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Fredrik Knapskog

Plasmonic response of supported and interacting spherical nanoparticles

Master's thesis in Applied Physics and Mathemathics
Supervisor: Ingve Simonsen
June 2021



NTNU
Norwegian University of
Science and Technology

Fredrik Knapskog

Plasmonic response of supported and interacting spherical nanoparticles

Master's thesis in Applied Physics and Mathemathics
Supervisor: Ingve Simonsen
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

NTNU

Kunnskap for en bedre verden

Abstract

Particle growth is common in various industries and sciences. The optical properties of nanoparticles grown on a substrate are useful when trying to monitor the layer thickness or the composition of the thin film. The software package GranFilm implemented by I. Simonsen and R. Lazzari is based on the work of D. Bedeaux and J. Vlieger, which assumes the quasi-static approximation. It computes linear optical coefficients for truncated spheres and spheroids by a substrate, taking account for the interaction with the substrate and other particles. The software package allows for high order in the multipole expansion regarding interaction with the substrate, but is limited to quadrupole order for the multipole expansion regarding interaction between the particles. A python code is written for full spheres above the substrate taking high order interaction between particles into account. By comparison, the quadrupole approximation breaks down for a surface density of 55% for an Al_2O_3 substrate and 45% for a TiO_2 substrate. The resonances of the optical coefficients are studied along with the near field calculations. Lower heights above the substrate induce red shifts of the energy positions of the resonances for any direction of the incident electric field, and the largest red shift is obtained for incident electric field parallel to the z-axis. Shorter distances between neighbouring spheres induce a larger red shift than the corresponding decrease in height, but only if the incident field is parallel to a short lattice vector. Otherwise, the shorter distances induce a slight blue shift due to the potential enhancements being symmetric rather than anti-symmetric, and the Coulomb force is hence acting as a restoring force. Lastly, the multipole expansion method is compared to the discrete dipole approximation software DDSCAT with great agreement except in the region near the resonances. The discretisation at 523 305 point dipoles is insufficient for obtaining accurate results where the field enhancements are at their largest.

Sammendrag

Partikkelvekst er vanlig i vitenskap og diverse industrier. De optiske egenskapene til nanopartikler dyrket på et substrat er nyttige når det gjelder overvåking av tykkelsen og sammensetningen til den tynne filmen. Programvarepakken GranFilm implementert av I. Simonsen and R. Lazzari er basert på arbeidet til D. Bedeaux og J. Vlieger, som antar den kvasistatiske approksimasjonen. Den beregner lineære optiske koeffisienter for sfærer og sfæroider trunkert av et substrat. Programvaren tar i betraktning samspillet med substratet og de andre partiklene. Den tillater høy orden i multipolutviklingen som omhandler samspillet med substratet, men er begrenset til kvadrupol orden i multipolutviklingen som omhandler samspillet mellom partiklene. En pythonkode er skrevet for hele sfærer over et substrat og tar dermed høy orden i betraktning av samspillet mellom partiklene. Fra sammenligning er grensen for hvor kvadrupolapproksimasjonen bryter ned funnet til å være en overflatetetthet på 55% for et Al_2O_3 substrat og 45% for et TiO_2 substrat. Resonansen til de optiske koeffisientene er studert sammen med nærfeltsberegningene. Lavere høyder over substratet induserer rødforskyvninger av energiposisjonene til resonansene for alle retninger for det innkommende elektriske feltet, og den største rødforskyvningen oppnås for innkommende elektrisk felt parallelt med z-aksen. Kortere avstander mellom nabosfærer induserer en større rødforskyvning enn tilsvarende minking av høyde, men kun for innkommende felt parallelt med en kort gittervektor. Ellers vil kortere avstander indusere en svak blåforskyvning som følge av symmetriske potensialøkninger istedenfor antisymmetriske, og Coulombkraften vil da fungere som en gjenopprettende kraft. Til slutt er multipolutviklingsmetoden sammenlignet med den diskrete dipolapproksimasjonsprogramvaren DDSCAT med gode overenstemmelser med unntak av området i nærheten av resonansene. Diskretiseringen på 523 305 punktdipoler er utilstrekkelig for nøyaktige resultater når feltøkningene er på deres største.

Preface

This master's thesis concludes the five year long Master of Science program in Applied Physics and Mathematics at the Norwegian University of Science and Technology in Trondheim. The work on the thesis was performed at the tenth semester of the study program during the spring of 2021, with Prof. Ingve Simonsen as supervisor from the Department of Physics. This master's thesis is a continuation of the pre-masters project from the fall of 2020. Chapter 1, Secs. 2.1.1–2.2.5, 2.2.8, 2.2.11, 2.3, 3.1.1, 3.3, and 4.1.1–4.1.2 are hence heavily influenced by the pre-masters project if not unchanged at all.

As a student with interests in the fields of physics, mathematics and programming, a numerical project such as this one is rather rewarding, as numerics unifies the three sciences. Physics provides an application for the problem, mathematics provides the tools necessary for solving the problem and programming provides the force and power to execute solution. The result is a varied combination of setting up the problem, solving the problem and analysing the result.

I would like to thank Prof. Ingve Simonsen for all the great supervision and advice he has given me and for the interesting project he made for me. He is one to rely on, and he always shows up with a smile. During this CoVid-19 pandemic I would also like to thank my girlfriend of three years, Gunhild Holen Eimhjellen, for accompanying me in these solitude times. There are a lot of lonely students these days without the ability to socialise, and my thoughts are with them.

Fredrik Knapskog
Trondheim, June 2021

Acknowledgemets

I would like to acknowledge D. Bedeaux and J. Vlieger [1] for laying the foundation for the theory and P. A. Letnes, I. Simonsen and D. L. Mills [2, 3] for the numerical results used for verification of the implementations in App. A.1. I would like to acknowledge Numba [4], because without a just-in-time compiler the software could not have been written in python due to its slow nature. I would also like to acknowledge the freely available software SHTOOLS [5] for allowing calculation of spherical harmonics of higher order than $l = 86$ and the Sopra database [6] for providing experimental data for the dielectric functions. I would like to acknowledge B. T. Draine and P. J. Flatau [7, 8, 9] for their freely available software DDSCAT. Moreover, I would like to acknowledge the nanoHUB tool DDSCAT Shape Generator [10] for generating the discretisation of a sphere and the open data analysis and visualisation application ParaView [11] for visualising the discretisation of the scattering objects. I would like to acknowledge C. F. Bohren and D. R. Huffman [12] for the Fortran code for the absorption efficiency factor and H. Kaiser [13] for translating it to python. I would like to acknowledge I. Simonsen and R. Lazzari [14] for developing the freely available software GranFilm, and especially I. Simonsen for providing the reflectivities obtained from GranFilm in Sec. 4.3.2. Lastly, I would like to acknowledge the Department of physics at the Norwegian University of Science and Technology for allocation of computer time on their computer cluster.

Contents

Chapter 1

Introduction

The growth of nanoparticles is an active field of research with applications involving nanoelectronics, chemical sensing, composite materials, biology and medicine [15]. Thin films are produced daily in the semiconductors, glass and coatings industries. The nanoparticles are formed by condensing a gas of metallic atoms onto a substrate with a relatively large surface energy [15]. The deposited metal will thus start to form islands modelled by the Volmer-Weber growth [15]. The nanoparticles represent a perturbation from the easily solvable case of an ideally planar surface between two bulk media [15]. In order to learn more about the growth process and monitoring the layer thickness in-situ, it is common to perform ellipsometric measurements such as Surface Differential Reflectance Spectroscopy [15]. This involves measuring the specular reflectance spectrum of the surface by using linearly polarised radiation in and around the visible range. The Surface Differential Reflectance Spectroscopy is then the relative change, $\Delta R/R$, of the specular reflectance compared to the reference measurements, R, from before the nanoparticles were present i.e. for the plain substrate.

In the 1970s, the available theories for the reflectance of a thin film were essentially limited to effective medium theories, such as the Maxwell-Garnet theory and the Bruggeman formula, and to the dipolar Yamaguchi model [1]. The theories gave a decent qualitative description of the behaviour of thin metallic films, but lacked the quantitative descriptions. D. Bedeaux and J. Vlieger [1] improved the quantitative description when they developed their theory for thin island films and rough surfaces, based on classical electromagnetism where they solve Maxwell's equations in the area of the nanoparticles in the quasi-static limit [15]. The approach involves the potentials from the solution of Laplace's equation and solving for the multipole expansion coefficients from the equations which arise from the boundary conditions. The susceptibilities γ and β are the integrated surface polarisation parallel and perpendicular to the surface, respectively, and can be determined from the multipole expansion coefficients. From the susceptibilities the reflectivities for s- and p-polarised light can be calculated.

The shapes of the islands in the thin island films can be spheres or spheroids placed either above the substrate or truncated by the substrate. The theory of D. Bedeaux and J. Vlieger takes account of the interactions within the islands and the interaction with the substrate. For islands not truncated by the substrate the orthonormality of the spherical harmonics can be taken advantage of when solving for the expansion coefficients. Otherwise, the number of numerical integrals concerning the neighbour island interactions in the multipole expansion gets out of hand. I. Simonsen and R. Lazzari implemented the theory of D. Bedeaux and J. Vlieger for truncated island films into the software GranFilm [14]. Consequently, the contributions in the multipole expansion regarding the neighbour interactions are limited to quadrupole order. The optical properties of the thin film are

strongly dependent on the shapes of the islands, especially near the resonances. Hence, the accuracy of the substrate interaction and the shapes of the islands are of greater interest for GranFilm than the high surface density limit.

The aim of this thesis is to gain better insight to where the quadrupole approximation in the neighbour interaction is valid for GranFilm. The approach is to develop a software for whole spheres not truncated by the substrate and thus placed a separating height above the substrate. The software should be able to distinguish between the multipole order for the interaction with the substrate and the multipole order for the interaction with the other spheres. The simulations for quadrupole order in neighbour interactions can thus be compared to simulations for a high multipole order where the solution has converged sufficiently. In both simulations the interaction with the substrate should be calculated for a high multipole order. The comparisons for various surface densities may provide a decent overview of where the quadrupole approximation is valid and where it breaks down.

A completely different approach for the same problem is the discrete dipole approximation. Instead of limiting the problem to a sphere or a spheroid, the shape of the scattering object can take any form. The object is discretised to a set of sub volumes referred to as point dipoles, and the polarisation for each point dipole is solved for. The polarisations can be used to determine the Mueller matrix elements and thus the reflectivities for any degree of polarised light. The freely available software DDSCAT [7, 8, 9] utilises the discrete dipole approximation and is developed by B. T. Draine and P. J. Flatau. DDSCAT allows for periodic boundaries such that the reflectivities for lattices of islands obtained from GranFilm and the multipole expansion software developed here can be compared to DDSCAT. However, DDSCAT does not support substrates yet. The comparisons must hence be for islands hovering in the ambient medium.

Along the way this thesis will also study the characteristics of the resonances for the optical properties of the thin films. How the resonances are affected by various factors will be in focus. The factors are the dielectric functions of the substrate and the islands, and the island separation distance and height above the substrate. The interests in the plasmonic response of substrates for the purpose of enhancing electric fields of laser beams in their near vicinity are great [2]. Such enhancements due to excitation of collective plasmon modes have the potential of increasing the field intensity in the near vicinity by many orders [2]. The phenomenon was first explored in Raman scattering, but the use has been applied to cross sections of diverse nonlinear optical processes [2]. These field enhancements will be studied here along with the potential enhancements, for both a supported and unsupported dimer. The near field calculations can be rather useful in order to gain more insight in the characteristics of the resonances. The field enhancement is also rather useful for comparing the multipole expansion to DDSCAT.

Chapter 2

Theory

2.1 Bulk theory

2.1.1 Induced dipole moment in a metallic sphere

A metallic sphere exposed to an electric field, $\mathbf{E}$, experiences an induced polarisation due to the electrostatic force, $\mathbf{F}$, exerted on the sphere's charge carriers

$$\mathbf{F} = q\mathbf{E}. \tag{2.1.1}$$

A charge q's sign determines the direction of the force. Thus carriers of opposite charges are

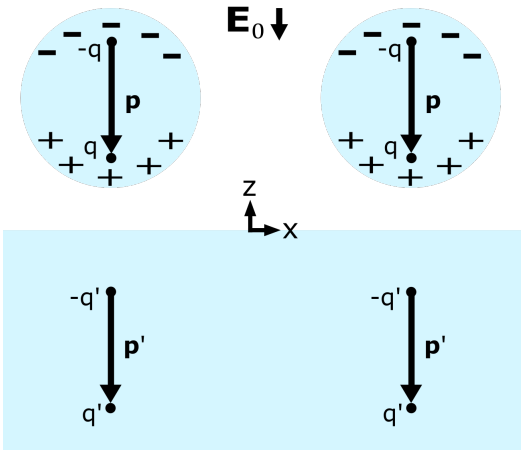


FIGURE 2.1: Two metallic spheres placed above a dielectric substrate occupying the half space $z < 0$. The induced charge distributions occur when they are exposed to an external electric field. The black dots represent point charge approximations of the charge distributions as well as the image charges of these as seen by an observer in the half space $z > 0$. The external electric field is here orthogonal to the substrate such that the induced dipole moments denoted with arrows are lined vertically.

separated in opposite directions. The separation continues until an equilibrium is reached due to the Coulomb force from the other charge carriers. Then, there will be two concentrations of opposite charges. The concentrations can cause the electric field to be locally stronger than the external electric field. The concentrations are placed at opposite sides of the sphere according to the direction of the external electric field. If the observer is far away from the sphere, the charge

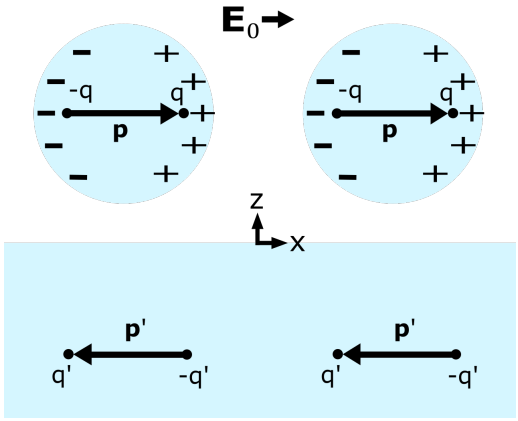


FIGURE 2.2: Two metallic spheres placed above a dielectric substrate occupying the half space $z < 0$. The induced charge distributions occur when they are exposed to an external electric field. The black dots represent point charge approximations of the charge distributions as well as the image charges of these as seen by an observer in the half space $z > 0$. The external electric field is here parallel to the substrate such that the induced dipole moments denoted with arrows are lined horizontally.

concentrations can be simplified into two point charges. An electric dipole moment, $\mathbf{p}$, will then occur as illustrated in Figs. 2.1 and 2.2

$$\mathbf{p} = \mathbf{d}|q|. \tag{2.1.2}$$

The vector $\mathbf{d}$ is the distance from the negative point charge to the positive one. As the figures show, the induced dipole moments in the spheres point in the same direction as the external electric field. Furthermore, for two spheres placed next to each other, the dipole moments line up in series when the external electric field is parallel to the axis through the centres of the two spheres. That is not the case for the electric field parallel to either of the other two axes.

2.1.2 Boundary conditions on the interface between two media

At the interface between media 1 and 2, the component of the electric field parallel with respect to the interface must be continuous [16]

$$\mathbf{E}_{1,\parallel} - \mathbf{E}_{2,\parallel} = 0, \tag{2.1.3}$$

while the component of the electric displacement field perpendicular with respect to the interface will be discontinuous for non zero free surface charge density, σ_f

$$\mathbf{D}_{1,\perp} - \mathbf{D}_{2,\perp} = \sigma_f. \tag{2.1.4}$$

The electric displacement field takes the form

$$\mathbf{D} \equiv \varepsilon_0 \mathbf{E} + \mathbf{P}, \tag{2.1.5}$$

where ε_0 is the vacuum permittivity and $\mathbf{P}$ is the polarisation density

$$\mathbf{P} \equiv \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}\tau}, \tag{2.1.6}$$

with τ denoting volume. For a linear, homogeneous and isotropic dielectric medium the polarisation density is linearly proportional to the electric field

$$\mathbf{P} = \varepsilon_0 \chi \mathbf{E}, \tag{2.1.7}$$

with the susceptibility, χ, multiplied by the vacuum permittivity as the constant of proportionality. Inserting the polarisation density in Eq.(2.1.7) into the electric displacement field from Eq. (2.1.5) yields

$$\mathbf{D} = \varepsilon_0\mathbf{E} + \varepsilon_0\chi\mathbf{E} = \varepsilon_0(1+\chi)\mathbf{E} = \varepsilon_0\varepsilon_r\mathbf{E} = \varepsilon\mathbf{E}. \tag{2.1.8}$$

The relative permittivity, ε_r, will from here be referred to as the dielectric function or as the permittivity, and ε will denote the relative permittivity instead of the absolute permittivity in Eq. (2.1.8). Often, the permittivities in the computations appear in fractions, both in the dividend and the divisor, and so forth the vacuum permittivities cancel out anyway.

2.1.3 The Drude model

The permittivity of the spheres can be modelled by the Drude model

$$\varepsilon_j(\omega) = 1 - \frac{\omega_p^2}{\omega(\omega + i\gamma)}, \tag{2.1.9}$$

with ω as the angular frequency of the incident plane wave, ω_p as the plasma frequency and γ as the inverse of the free carrier relaxation time. The model takes into account that frequencies greater than the plasma frequency results in the real part of the permittivity becoming negative. Hence, the spheres are not considered metallic if the frequency is too high. The transition is a consequence of the charge carriers in the metal being unable to oscillate sufficiently fast if the incident field changes too rapidly. Otherwise, the model is not a very accurate formula [16].

2.1.4 Method of images

If a sphere is placed above a substrate with a dielectric function greater than 1, the induced charges from the sphere will induce new charges in the substrate. The potential is required to be continuous at the surface of the substrate, and to be convergent in the far field limit. The method of images exploits the first uniqueness theorem [16] in order to deal with the substrate interaction. Consequently, if a function for the potential satisfies the two boundary conditions, the function is guaranteed to be the only solution to meet those requirements. The function for the potential is found by replacing the substrate with an image charge, q', with opposite sign of the corresponding point charge in the sphere, at the position mirrored by the substrate's surface. The amplitude of the image charge will most likely differ from the the one in the sphere, as it depends on the dielectric functions of the substrate and the ambient medium in order to fulfil the boundary condition at the interface of the two media. The image charges in Figs. 2.1–2.2 form point charge dipole moments as well. This time, the dipole moments in the substrate line up in series with the ones from the spheres when the external electric field is orthogonal to the substrate. When the field is parallel to the substrate on the other hand, the dipole moments from the spheres point in opposite direction of the ones from the substrate, and they are not in series.

2.1.5 Multipole expansion of the Coulomb potential

The Coulomb potential, V, evaluated at position $\mathbf{r}$, from a set of point charges is described by

$$V(\mathbf{r}) = \frac{1}{4\pi\varepsilon}\sum_i \frac{q_i}{|\mathbf{r} - \mathbf{r}_i'|}, \tag{2.1.10}$$

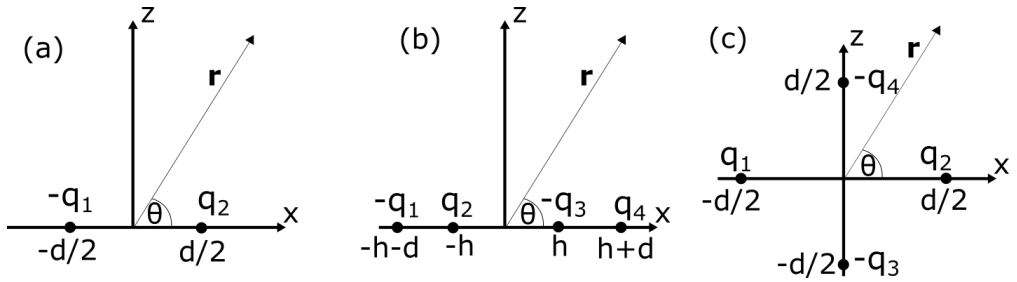


FIGURE 2.3: Three cases of point charges q placed to form (a) a dipole, (b) two dipoles in series, and (c) a quadrupole. The observation point is described by the vector $\mathbf{r}$, which makes an angle θ with the x-axis. For the single dipole, the charges are separated by a distance d. The dipoles in series are of the same lengths, d, but separated by a distance of $2h$. The point charges in the quadrupole are all placed a distance $d/2$ from the origin.

where ε is the absolute permittivity of the ambient medium, and $\mathbf{r}'_i$ is the position of point charge q_i. The distance in the denominator of Eq. (2.1.10) can be rewritten as

$$|\mathbf{r} - \mathbf{r}'_i| = \sqrt{r^2 - 2\mathbf{r}\cdot\mathbf{r}'_i + r'^2_i} = r\sqrt{1+\eta}, \tag{2.1.11}$$

where

$$\eta = -2\frac{\hat{\mathbf{r}}\cdot\mathbf{r}'_i}{r} + (\frac{r'_i}{r})^2. \tag{2.1.12}$$

The binomial series

$$(1+x)^s = \sum_{n=0}^{\infty}\binom{s}{n}x^n, \tag{2.1.13}$$

can be applied to expand Eq. (2.1.11) for small η when $\mathbf{r} \gg \mathbf{r}'_i$ as done in Ref. [16]. The name of the series originates from the binomial coefficients

$$\binom{s}{n} \equiv \frac{s!}{n!(s-n)!}, \tag{2.1.14}$$

where ! represents the factorial function. When inserting $-1/2$ for s and η for x in the expansion, the Coulomb potential takes the form

$$V(\mathbf{r}) = \frac{1}{4\pi\varepsilon r}\sum_i q_i(1 - \frac{1}{2}\eta + \frac{3}{8}\eta^2 + \dots) = \frac{1}{4\pi\varepsilon r}\sum_i q_i\left[1 + \frac{\hat{\mathbf{r}}\cdot\mathbf{r}'_i}{r} + \frac{3(\hat{\mathbf{r}}\cdot\mathbf{r}'_i)^2 - r'^2_i}{2r^2} + O(\frac{r'_i}{r})^3\right]. \tag{2.1.15}$$

The series can be written as

$$V(\mathbf{r}) = V_{\text{mono}}(\mathbf{r}) + V_{\text{di}}(\mathbf{r}) + V_{\text{quad}}(\mathbf{r}) + \dots, \tag{2.1.16}$$

where

$$\begin{aligned} V_{\text{mono}}(\mathbf{r}) &= \frac{1}{4\pi\varepsilon r}\sum_i q_i, \\ V_{\text{di}}(\mathbf{r}) &= \frac{1}{4\pi\varepsilon r^2}\sum_i q_i\hat{\mathbf{r}}\cdot\mathbf{r}'_i, \\ V_{\text{quad}}(\mathbf{r}) &= \frac{1}{4\pi\varepsilon r^3}\sum_i q_i\frac{3}{2}\left[(\hat{\mathbf{r}}\cdot\mathbf{r}'_i)^2 - r'^2_i\right] \\ &\vdots \end{aligned} \tag{2.1.17}$$

The term, V_{mono}, is the monopole term, V_{di} is the dipole term, V_{quad} is the quadrupole term and so on. This is where the expression multipole expansion originates from. As the order increases, the terms decay faster with distance r. Thus, the monopole term dominates when only one point charge is present. For a dipole of two equal but opposite charges separated by a distance d as in Fig. 2.3(a) on the other hand, the case will be different. The scalar product, $\hat{\mathbf{r}} \cdot \mathbf{r}'_i$, can be found from the angle, θ, in Fig. 2.3(a) and the distance between the two charges,

$$\hat{\mathbf{r}} \cdot \mathbf{r}'_1 = \frac{d}{2}\cos(\pi - \theta) = -\frac{d}{2}\cos\theta \quad \text{and} \quad \hat{\mathbf{r}} \cdot \mathbf{r}'_2 = \frac{d}{2}\cos\theta. \tag{2.1.18}$$

The potential from the dipole hence becomes

$$V(\mathbf{r}) = \frac{q}{4\pi\varepsilon r}\left[(1-1) + \left(\frac{d\cos\theta}{2r} - \frac{-d\cos\theta}{2r}\right) + O\left(\frac{r'_i}{r}\right)^2\right] = \frac{p\cos\theta}{4\pi\varepsilon r^2} + O(r^{-3}). \tag{2.1.19}$$

Here the monopole term has vanished and the potential is dominated by the dipole term. Consequently, the potential decays as r^{-2}. Putting two equal dipoles in series as shown in Fig. 2.3 (b) and using the same procedure gives

$$V(\mathbf{r}) = \frac{q}{4\pi\varepsilon r}\left[(2-2) + \frac{\cos\theta}{r}[(h+d) - h - h + (h+d)] + O\left(\frac{r'_i}{r}\right)^2\right] = \frac{p\cos\theta}{2\pi\varepsilon r^2} + O(r^{-3}), \tag{2.1.20}$$

which results in a twice as strong potential as for a single dipole. The quantity $2h$ is the distance used between the dipole moments, but the potential turned out independent of this separation when $r \gg h + d$. When looking at the interaction between two spheres in Fig. 2.2, where the field is parallel to the axis through the centres of the two spheres, the result from Eq. (2.1.20) is quite descriptive. Likewise for the interaction with the substrate in Fig. 2.1, where the external field is orthogonal to the substrate, if the amplitudes of the image charges are close to the amplitudes of the point charges in the spheres. For the interaction with the substrate in Fig. 2.2 on the other hand, the model from Fig. 2.3(c) can be used if the distance separating the dipole moments are close to the distances of the dipole moments, that is, the height above the substrate is small. Moreover, the amplitudes of the image charges must be close to the amplitudes of the point charges in the spheres. The four scalar products then turn into

$$\begin{aligned}
\hat{\mathbf{r}} \cdot \mathbf{r}'_1 &= \frac{d}{2}\cos(\pi - \theta) = -\frac{d}{2}\cos\theta \\
\hat{\mathbf{r}} \cdot \mathbf{r}'_2 &= \frac{d}{2}\cos\theta \\
\hat{\mathbf{r}} \cdot \mathbf{r}'_3 &= \frac{d}{2}\cos\big(\pi/2 + \theta\big) = \frac{d}{2}\cos\big(\pi/2 - (-\theta)\big) = \frac{d}{2}\sin(-\theta) = -\frac{d}{2}\sin\theta \\
\hat{\mathbf{r}} \cdot \mathbf{r}'_4 &= \frac{d}{2}\sin\theta,
\end{aligned} \tag{2.1.21}$$

which results in the potential

$$\begin{aligned}
V(\mathbf{r}) &= \frac{q}{4\pi\varepsilon r}\left[(2-2) + \left[\frac{\cos\theta}{r}(1-1) + \frac{\sin\theta}{r}(1-1)\right] + \frac{6d^2(\cos^2\theta - \sin^2\theta)}{8r^2} + O\left(\frac{r'_i}{r}\right)^3\right] \\
&= \frac{3Q\cos(2\theta)}{16\pi\varepsilon r^3} + O(r^{-4}),
\end{aligned} \tag{2.1.22}$$

where Q is the quadrupole moment

$$Q = d^2|q|. \tag{2.1.23}$$

The potential in Eq. (2.1.22) decays as r^{-3}, as it is a pure quadrupole.

2.2 Optical properties for a metasurface of supported and interacting nanospheres

2.2.1 The metasurface

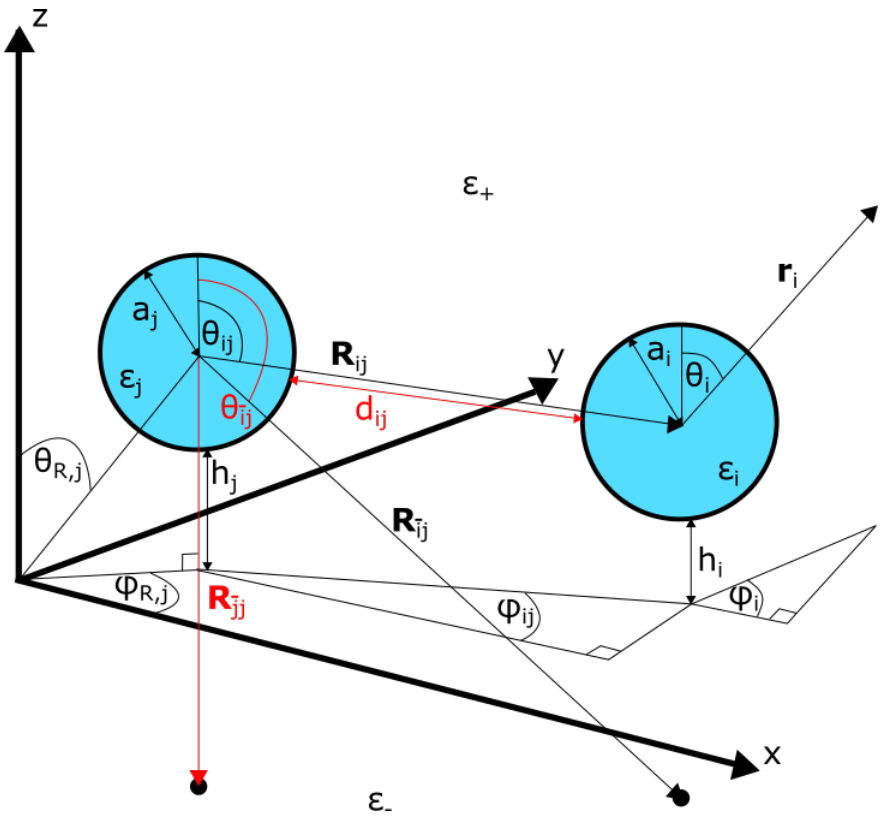


FIGURE 2.4: Two nanospheres, j and i, placed a distance h_j and h_i, respectively, above a substrate occupying the half space $z < 0$. The black dots represent the spheres' image multipoles as seen by an observer in the half space $z > 0$. Sphere i and j have radii a_i and a_j, respectively, and are separated by a distance d_{ij}. The medium above the substrate has a dielectric function $\varepsilon_+(\omega)$, and the substrate's dielectric function is $\varepsilon_-(\omega)$. The spheres have the dielectric functions $\varepsilon_j(\omega)$ and $\varepsilon_i(\omega)$. Sphere j's polar and azimuthal angle with respect to the origin are $\theta_{R,j}$ and $\phi_{R,j}$, respectively. Similarly, sphere i's polar and azimuthal angle relative to the centre of sphere j's coordinate system are θ_{ij} and ϕ_{ij}. The polar angle of sphere i's image multipole in sphere j's coordinate system is $\theta_{\bar{i}j}$. The position of sphere i relative to sphere j is $\mathbf{R}_{ij}$. The position of sphere j's image multipole relative to sphere j is $\mathbf{R}_{\bar{j}j}$, and the position of the image multipole of sphere i relative to sphere j is $\mathbf{R}_{\bar{i}j}$. The position vector relative to sphere i is $\mathbf{r}_i$ and its polar and azimuthal angle are θ_i and ϕ_i, respectively.

The metasurface is a set of nanospheres of radius a_i with corresponding dielectric functions ε_i as illustrated in Fig. 2.4. The spheres are placed in an ambient medium with a dielectric function ε_+ right above the interface with a dielectric substrate with a dielectric function ε_-. The height separating the spheres from the substrate is denoted as h_i, and the distance separating sphere i from sphere j is denoted as d_{ij}. The global coordinate system is defined with the z-axis orthogonal to the substrate and pointing upwards from the substrate. The global position vector is denoted as $\mathbf{r}$ and has its origin at the interface between the substrate and the ambient medium. The unit length used for $\mathbf{r}$ is the radius from one of the spheres. The position vectors of the centres of the spheres are denoted as $\mathbf{R}_i$, and their components in spherical coordinates are denoted as R_i, $\theta_{R,i}$

and $\phi_{R,i}$. Similarly, the position vectors for the image multipoles are denoted as $\mathbf{R}_{\bar{i}}$. The position vector with origin in the centre of sphere i is denoted as $\mathbf{r}_i = \mathbf{r} - \mathbf{R}_i$, and its components in spherical coordinates with origin in centre of sphere i are r_i, θ_i and ϕ_i. Similarly for the position vector with origin in the image multipole of sphere i, $\mathbf{r}_{\bar{i}} = \mathbf{r} - \mathbf{R}_{\bar{i}}$. Finally, the position vector of the centre of sphere i with origin in the centre of sphere j is denoted as $\mathbf{R}_{ij} = \mathbf{R}_i - \mathbf{R}_j$, and its components in spherical coordinates with origin in the centre of sphere j is R_{ij}, θ_{ij} and ϕ_{ij}. Similarly for the position vector of the image multipole of sphere i with origin in centre of sphere j, $\mathbf{R}_{\bar{i}j} = \mathbf{R}_{\bar{i}} - \mathbf{R}_j$.

2.2.2 Maxwell's equations and the quasistatic approximation

Maxwell's equations have the form [16]

$$\begin{aligned}
\nabla \cdot \mathbf{D} &= \rho \\
\nabla \cdot \mathbf{B} &= 0 \\
\nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\
\nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t},
\end{aligned} \tag{2.2.1}$$

where ρ is the electric charge density, $\mathbf{B}$ is the magnetic flux density, $\mathbf{H}$ is the magnetic field and $\mathbf{J}$ is the electric current density. As the divergence of $\mathbf{B}$ is zero, it is possible to rewrite $\mathbf{B}$ in terms of a vector potential $\mathbf{A}$

$$\mathbf{B} = \nabla \times \mathbf{A}, \tag{2.2.2}$$

since the divergence of a curl is always zero. Inserting the vector potential into the third of Maxwell's equations yields

$$\nabla \times \mathbf{E} = -\frac{\partial}{\partial t}(\nabla \times \mathbf{A}). \tag{2.2.3}$$

Factorising Eq. (2.2.3) with respect to the curl operator results in a conservative field

$$\nabla \times (\mathbf{E} + \frac{\partial \mathbf{A}}{\partial t}) = 0. \tag{2.2.4}$$

The conservative field can thus be written in terms of a scalar potential ϕ

$$\mathbf{E} + \frac{\partial \mathbf{A}}{\partial t} = -\nabla \phi. \tag{2.2.5}$$

On a scale much smaller than half a wave length of the incident plane wave, the variations in the electric field across the domain will be small at a given time. The quasi-static approximation hence considers a static image such that the time derivative is neglected

$$\mathbf{E} = -\nabla \phi. \tag{2.2.6}$$

Inserting the electric field from Eq. (2.2.6) into the first of Maxwell's equations

$$\nabla \cdot \mathbf{D} = \nabla \cdot (\varepsilon \mathbf{E}) = -\varepsilon \nabla^2 \phi = \rho \tag{2.2.7}$$

results in Poisson's equation for the potential

$$\nabla^2 \phi = -\frac{\rho}{\varepsilon}. \tag{2.2.8}$$

For a neutrally charged sphere, with hence $\rho = 0$, Poisson's equation is reduced to Laplace's equation

$$\nabla^2 \phi = 0. \tag{2.2.9}$$

2.2.3 The solution to Laplace's equation

The potential, ψ_j, from a neutrally charged sphere j positioned at $\mathbf{R}_j$, when viewed from the half space $z > 0$ as shown in Fig. 2.4, satisfies Eq. (2.2.9) when [1]

$$\psi_j(\mathbf{r}_j) = \begin{cases} \sum_{lm} A_{lm}^{(j)} r_j^{-l-1} Y_l^m(\theta_j, \phi_j), & r_j \geq a_j, \\ \sum_{lm} B_{lm}^{(j)} r_j^l Y_l^m(\theta_j, \phi_j), & r_j < a_j. \end{cases} \tag{2.2.10}$$

Similarly, the potential from the image multipole of sphere j satisfies Eq. (2.2.9) when

$$\psi_{\bar{j}}(\mathbf{r}_{\bar{j}}) = \sum_{lm} A_{lm}^{(j,R)} r_{\bar{j}}^{-l-1} Y_l^m(\theta_{\bar{j}}, \phi_{\bar{j}}). \tag{2.2.11}$$

However, when viewed from the half space $z < 0$, the image multipoles don't contribute and the potential from sphere j becomes

$$\psi_j^T(\mathbf{r}_j) = \sum_{lm} A_{lm}^{(j,T)} r_j^{-l-1} Y_l^m(\theta_j, \phi_j). \tag{2.2.12}$$

The shorthand notation used is $\sum_{lm} = \sum_{l=0}^{\infty} \sum_{m=-l}^{l}$, and the quantities $A_{lm}^{(j)}$, $B_{lm}^{(j)}$, $A_{lm}^{(j,R)}$ and $A_{lm}^{(j,T)}$ are unknown expansion coefficients to solve for.

2.2.4 Spherical harmonics

The eigenfunctions of the angular part of the Laplacian operator, $Y_l^m(\theta_j, \phi_j)$, are referred to as spherical harmonics [17]. Here the orthonormal normalisation is used with the Condon-Shortley phase included

$$Y_l^m(\theta, \phi) = (-1)^m \sqrt{\frac{2l+1}{4\pi} \frac{(l-m)!}{(l+m)!}} e^{im\phi} P_l^m(\cos(\theta)), \tag{2.2.13}$$

where P_l^m denotes the associated Legendre functions. The orthonormality of the spherical harmonics is fulfilled by

$$\int_0^{\pi} d\theta \sin\theta \int_0^{2\pi} d\phi \left[Y_l^m(\theta, \phi)\right]^* Y_{l'}^{m'}(\theta, \phi) = \delta_{ll'} \delta_{mm'}, \tag{2.2.14}$$

with the asterisk * denoting the complex conjugate, and consequently

$$\int_0^{\pi} d\theta \sin\theta \int_0^{2\pi} d\phi Y_l^m(\theta, \phi) = \int_0^{\pi} d\theta \sin\theta \int_0^{2\pi} d\phi \left[Y_l^m(\theta, \phi)\right]^* = \delta_{l,0} \sqrt{4\pi}. \tag{2.2.15}$$

2.2.5 External incident electric field

Considering the external electric field of an incident plane wave in the ambient medium

$$\mathbf{E}_0 = E_0(\sin\theta_0 \cos\phi_0, \sin\theta_0 \sin\phi_0, \cos\theta_0), \tag{2.2.16}$$

where E_0 is the field strength, θ_0 is the polar angle of the field, and ϕ_0 is the azimuthal angle. The corresponding potential can then be described as

$$\psi_0(\mathbf{r}) = -\mathbf{r} \cdot \mathbf{E}_0. \tag{2.2.17}$$

The inner product can be expanded in spherical harmonics by [1]

$$
\begin{aligned}
-\mathbf{r}\cdot\mathbf{E}_0 &= -rE_0\sqrt{\frac{2\pi}{3}}\left[\sqrt{2}\cos(\theta_0)Y_1^0(\theta,\phi)+\sin(\theta_0)\{e^{i\phi_0}Y_1^{-1}(\theta,\phi)-e^{-i\phi_0}Y_1^1(\theta,\phi)\}\right] \\
&= r\sum_{l,m} b_{l,m}Y_l^m(\theta,\phi). \qquad (2.2.18)
\end{aligned}
$$

where

$$
\begin{aligned}
b_{1,0} &= -E_0\sqrt{\frac{4\pi}{3}}\cos(\theta_0) \\
b_{1,\pm1} &= \pm E_0\sqrt{\frac{2\pi}{3}}\sin(\theta_0)e^{\mp i\phi_0} \\
b_{l_j,m_j} &= 0 \quad \text{else.} \qquad (2.2.19)
\end{aligned}
$$

As only the change in potential energy is of physical significance the position of the zero point is arbitrary. The constant $b_{0,0}$ is thus set to zero to maintain generality. The terms higher than $l=1$ being zero is a consequence of the plane wave approximation. For the electric field of an incident plane wave when evaluated in the substrate the same approach yields

$$
\psi_0^T(\mathbf{r}) = b_{0,0}^T - \mathbf{r}\cdot\mathbf{E}_0^T = b_{0,0}^T + r\sum_{l=1}^{\infty}\sum_{m=-l}^{m=l} b_{l,m}^T Y_l^m(\theta,\phi), \qquad (2.2.20)
$$

where

$$
\begin{aligned}
b_{0,0}^T &= -E_0 z_0\cos(\theta_0)\left(\frac{\varepsilon_+}{\varepsilon_-}-1\right) \\
b_{1,0}^T &= -E_0\frac{\varepsilon_+}{\varepsilon_-}\sqrt{\frac{4\pi}{3}}\cos(\theta_0) \\
b_{1,\pm1}^T &= \pm E_0\sqrt{\frac{2\pi}{3}}\sin(\theta_0)e^{\mp i\phi_0} \\
b_{l_j,m_j}^T &= 0 \quad \text{else.} \qquad (2.2.21)
\end{aligned}
$$

Here, z_0 is the z-coordinate at the origin of the coordinate system of r.

2.2.6 The potential in the various regions

The potential in the ambient medium, ψ_+, is a superposition of the potential from the incident plane wave, the spheres and their corresponding image multipoles [2]

$$
\psi_+(\mathbf{r}) = \psi_0(\mathbf{r}) + \sum_{j=1}^{N}\psi_j(\mathbf{r}_j) + \sum_{\bar{j}=1}^{N}\psi_{\bar{j}}(\mathbf{r}_{\bar{j}}). \qquad (2.2.22)
$$

Similarly, the potential in the substrate, ψ_-, is a superposition of the potential from the spheres and the incident plane wave

$$
\psi_-(\mathbf{r}) = \psi_0^T(\mathbf{r}) + \sum_{j=1}^{N}\psi_j^T(\mathbf{r}_j). \qquad (2.2.23)
$$

Inside a sphere on the other hand, the only contribution to the potential is from the sphere itself.

2.2.7 The electric field

The electric field can be found from Eq. (2.2.6). In spherical coordinates the gradient takes the form

$$\nabla_j = \frac{\partial}{\partial r_j}\hat{\mathbf{r}}_j + \frac{1}{r_j}\frac{\partial}{\partial \theta_j}\hat{\boldsymbol{\theta}}_j + \frac{1}{r_j \sin\theta_j}\frac{\partial}{\partial \phi_j}\hat{\boldsymbol{\phi}}_j, \tag{2.2.24}$$

such that the electric field from sphere j yields

$$\mathbf{E}_j(\mathbf{r}_j) = E_r^{(j)}\hat{\mathbf{r}}_j + E_\theta^{(j)}\hat{\boldsymbol{\theta}}_j + E_\phi^{(j)}\hat{\boldsymbol{\phi}}_j = -\frac{\partial \psi_j(\mathbf{r}_j)}{\partial r_j}\hat{\mathbf{r}}_j - \frac{1}{r_j}\frac{\partial \psi_j(\mathbf{r}_j)}{\partial \theta_j}\hat{\boldsymbol{\theta}}_j - \frac{1}{r_j \sin\theta_j}\frac{\partial \psi_j(\mathbf{r}_j)}{\partial \phi_j}\hat{\boldsymbol{\phi}}_j. \tag{2.2.25}$$

The only part of the potential with angular dependencies is the spherical harmonics. The differentiation with respect to the azimuthal angle is trivial

$$\frac{\partial Y_l^m(\theta,\phi)}{\partial \phi} = imY_l^m(\theta,\phi), \tag{2.2.26}$$

whereas to differentiate the spherical harmonics with respect to the polar angle, the differentiation of the Legendre functions with respect to θ must be applied [18]

$$\frac{\partial P_l^m(\cos(\theta))}{\partial \theta} = m\cot(\theta)P_l^m(\cos(\theta)) - P_l^{m+1}(\cos(\theta)). \tag{2.2.27}$$

The differentiation of the spherical harmonics with respect to the polar angle hence becomes

$$\begin{aligned}
\frac{\partial Y_l^m(\theta,\phi)}{\partial \theta} &= m\cot(\theta)Y_l^m(\theta,\phi) - (-1)^m\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}e^{im\phi}P_l^{m+1}(\cos(\theta)) \\
&= m\cot(\theta)Y_l^m(\theta,\phi) - \frac{(-1)^m\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}e^{im\phi}}{(-1)^{m+1}\sqrt{\frac{2l+1}{4\pi}\frac{(l-(m+1))!}{(l+m+1)!}}e^{i(m+1)\phi}}Y_l^{m+1}(\theta,\phi) \\
&= m\cot(\theta)Y_l^m(\theta,\phi) + \sqrt{(l+m+1)(l-m)}e^{-i\phi}Y_l^{m+1}(\theta,\phi).
\end{aligned} \tag{2.2.28}$$

When adding the contributions to the total electric field from the various sources it is useful to convert them to a global coordinate system to allow for component-wise addition. This is done by the rotation matrix

$$\mathbf{E}_j(\mathbf{r}) = \begin{pmatrix} \sin\theta_{R,j}\cos\phi_{R,j} & \cos\theta_{R,j}\cos\phi_{R,j} & -\sin\phi_{R,j} \\ \sin\theta_{R,j}\sin\phi_{R,j} & \cos\theta_{R,j}\sin\phi_{R,j} & \cos\phi_{R,j} \\ \cos\theta_{R,j} & -\sin\theta_{R,j} & 0 \end{pmatrix}\mathbf{E}_j(\mathbf{r}_j) \equiv \boldsymbol{\Omega}(\theta_{R,j},\phi_{R,j})\mathbf{E}_j(\mathbf{r}_j). \tag{2.2.29}$$

The electric field in the two regions then become

$$\mathbf{E}_+(\mathbf{r}) = \mathbf{E}_0 + \sum_{j=1}^{N}\boldsymbol{\Omega}(\theta_{R,j},\phi_{R,j})\mathbf{E}_j(\mathbf{r}_j) + \sum_{\bar{j}=1}^{N}\boldsymbol{\Omega}(\theta_{R,\bar{j}},\phi_{R,\bar{j}})\mathbf{E}_{\bar{j}}(\mathbf{r}_{\bar{j}}), \tag{2.2.30}$$

and

$$\mathbf{E}_-(\mathbf{r}) = \mathbf{E}_0^T + \sum_{j=1}^{N}\boldsymbol{\Omega}(\theta_{R,j},\phi_{R,j})\mathbf{E}_j^T(\mathbf{r}_j), \tag{2.2.31}$$

with

$$\mathbf{E}_j(\mathbf{r}_j) = -\nabla_j\psi_j(\mathbf{r}_j), \quad \mathbf{E}_{\bar{j}}(\mathbf{r}_{\bar{j}}) = -\nabla_{\bar{j}}\psi_{\bar{j}}(\mathbf{r}_{\bar{j}}), \quad \text{and} \quad \mathbf{E}_j^T(\mathbf{r}_j) = -\nabla_j\psi_j^T(\mathbf{r}_j). \tag{2.2.32}$$

2.2.8 Determining the expansion coefficients for a finite set of spheres from the boundary conditions

For the boundary condition on the interface between the ambient medium and the substrate to be satisfied by the method of images, the A coefficients can be related by [1]

$$A_{lm}^{(j,R)} = (-1)^{l+m}\beta A_{lm}^{(j)}, \quad \text{where} \quad \beta \equiv \frac{\varepsilon_+ - \varepsilon_-}{\varepsilon_+ + \varepsilon_-}, \tag{2.2.33}$$

and

$$A_{lm}^{(j,T)} = \gamma A_{lm}^{(j)}, \quad \text{where} \quad \gamma \equiv \frac{2\varepsilon_+}{\varepsilon_+ + \varepsilon_-}. \tag{2.2.34}$$

The expansion coefficients, $A_{lm}^{(j)}$, can then be found from two boundary conditions at the surface of the spheres. The first boundary condition requires the potential to be continuous at the interface of sphere j

$$\lim_{r_j \to a_j^-} \psi(\mathbf{r}) = \lim_{r_j \to a_j^+} \psi(\mathbf{r}). \tag{2.2.35}$$

Inserting the potential for inside sphere j on the left-hand-side and the potential for the ambient medium from Eq. (2.2.22) into the right-hand-side, the limits yield

$$\lim_{r_j \to a_j^-} \psi_j(\mathbf{r}_j) = \lim_{r_j \to a_j^+} \psi_+(\mathbf{r}) = \lim_{r_j \to a_j^+} \left\{ \psi_0(\mathbf{r}) + \sum_{i=1}^{N} \psi_i(\mathbf{r}_i) + \sum_{\bar{i}=1}^{N} \psi_{\bar{i}}(\mathbf{r}_{\bar{i}}) \right\}. \tag{2.2.36}$$

Inserting the potentials from Eqs. (2.2.10), (2.2.11) and (2.2.18) into Eq. (2.2.36) leads to

$$\begin{aligned}
&\lim_{r_j \to a_j^-} \sum_{l_j,m_j} B_{l_j,m_j}^{(j)} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) = \lim_{r_j \to a_j^+} \Bigg\{ r \sum_{l_j,m_j} b_{l_j,m_j} Y_{l_j}^{m_j}(\theta,\phi) \\
&+ \sum_{l_j,m_j} A_{l_j,m_j}^{(j)} r_j^{-l_j-1} Y_{l_j}^{m_j}(\theta_j,\phi_j) + \sum_{l_i,m_i} A_{l_i,m_i}^{(j,R)} r_{\bar{j}}^{-l_i-1} Y_{l_i}^{m_i}(\theta_{\bar{j}},\phi_{\bar{j}}) \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} A_{l_i,m_i}^{(i)} r_i^{-l_i-1} Y_{l_i}^{m_i}(\theta_i,\phi_i) + \sum_{i\neq j} \sum_{l_i,m_i} A_{l_i,m_i}^{(i,R)} r_{\bar{i}}^{-l_i-1} Y_{l_i}^{m_i}(\theta_{\bar{i}},\phi_{\bar{i}}) \Bigg\}.
\end{aligned} \tag{2.2.37}$$

The boundary condition has to be satisfied at all points on the surface of sphere j. As there can't be an infinite set of equations to solve numerically, a weak boundary condition has to be used instead. The weak boundary condition involves demanding the integral of the product of Eq. (2.2.35) and a test function, over the entire surface of sphere j to be satisfied. The orthonormality of the spherical harmonics and its complex conjugate from Eq. (2.2.14) is a good motivation for choosing the test function. The weak boundary condition from Eq. (2.2.35) hence takes the form

$$\int_0^\pi d\theta_j \sin\theta_j \int_0^{2\pi} d\phi_j \left[Y_{l'}^{m'}(\theta_j,\phi_j) \right]^* \left[\lim_{r_j \to a_j^-} \psi(\mathbf{r}) - \lim_{r_j \to a_j^+} \psi(\mathbf{r}) \right] = 0. \tag{2.2.38}$$

In order to exploit the orthonormality of the spherical harmonics and its complex conjugate from Eq. (2.2.14), all the contributions to the potential in Eq. (2.2.37) must be expressed in terms of $r_j^{-l_i-1} Y_{l_i}^{m_i}(\theta_j,\phi_j)$. For the incident electric field, the trivial substitution as defined in Sec. 2.2.1 is used [1]

$$\mathbf{r} \cdot \mathbf{E}_0 = \mathbf{R}_j \cdot \mathbf{E}_0 + (\mathbf{r} - \mathbf{R}_j) \cdot \mathbf{E}_0 = \mathbf{R}_j \cdot \mathbf{E}_0 + \mathbf{r}_j \cdot \mathbf{E}_0. \tag{2.2.39}$$

The shift in potential due to the change of the origin

$$\psi_0(\mathbf{R}_j) = -\mathbf{R}_j \cdot \mathbf{E}_0 = R_j \sum_{l,m} b_{l,m} Y_l^m(\theta_{R,j}, \phi_{R,j}), \tag{2.2.40}$$

is independent of $\mathbf{r}$ and thus $\mathbf{r}_j$. To express the potential from the other spheres in the coordinate system of sphere j on the other hand, is less trivial. The potentials have to be expanded around the centre of sphere j [1]

$$r_i^{-l_i-1} Y_{l_i}^{m_i}(\theta_i, \phi_i) = \sum_{l_j=0}^{\infty} \sum_{m_j=-l_j}^{l_j} H(l_j, m_j | l_i, m_i) \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij}, \phi_{ij})}{R_{ij}^{l_j+l_i+1}} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j, \phi_j), \tag{2.2.41}$$

where

$$H(l_j, m_j | l_i, m_i) = \sqrt{4\pi}(-1)^{l_i+m_j} \left[\frac{2l_i+1}{(2l_j+1)(2l+1)} \right]^{\frac{1}{2}} \left[\binom{l+m}{l_i+m_i} \binom{l-m}{l_j+m_j} \right]^{\frac{1}{2}}. \tag{2.2.42}$$

In Eq. (2.2.42), the notation used is $l = l_i + l_j$, $m = m_i - m_j$ and the binomial coefficients are familiar from Eq. (2.1.14). The limits in Eq. (2.2.36) expressed in the coordinates of sphere j thus become

$$\begin{aligned}
&\lim_{r_j \to a_j^-} \sum_{l_j,m_j} B_{l_j,m_j}^{(j)} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j, \phi_j) = \\
&\lim_{r_j \to a_j^+} \Bigg\{ -\mathbf{R}_j \cdot \mathbf{E}_0 + r_j \sum_{l_j,m_j} b_{l_j,m_j} Y_{l_j}^{m_j}(\theta_j, \phi_j) + \sum_{l_j,m_j} A_{l_j,m_j}^{(j)} r_j^{-l_j-1} Y_{l_j}^{m_j}(\theta_j, \phi_j) \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta A_{l_i,m_i}^{(j)} \sum_{l_j,m_j} H(l_j, m_j | l_i, m_i) \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{j}j}, \phi_{\bar{j}j})}{R_{\bar{j}j}^{l_j+l_i+1}} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j, \phi_j) \\
&+ \sum_{i \neq j} \sum_{l_i,m_i} A_{l_i,m_i}^{(i)} \sum_{l_j,m_j} H(l_j, m_j | l_i, m_i) r_j^{l_j} Y_{l_j}^{m_j}(\theta_j, \phi_j) \Bigg[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij}, \phi_{ij})}{R_{ij}^{l_j+l_i+1}} \\
&+ (-1)^{l_i+m_i} \beta \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j}, \phi_{\bar{i}j})}{R_{\bar{i}j}^{l_j+l_i+1}} \Bigg] \Bigg\}.
\end{aligned} \tag{2.2.43}$$

Inserting the limits from Eq. (2.2.43) into the integrand in the weak boundary condition from Eq. (2.2.38) and letting r_j approach a_j yields

$$\begin{aligned}
B_{l_j,m_j}^{(j)} a_j^{l_j} &= -\mathbf{R}_j \cdot \mathbf{E}_0 \delta_{0,l_j} + a_j b_{l_j,m_j} + A_{l_j,m_j}^{(j)} a_j^{-l_j-1} \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta A_{l_i,m_i}^{(j)} H(l_j, m_j | l_i, m_i) \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{j}j}, \phi_{\bar{j}j})}{R_{\bar{j}j}^{l_j+l_i+1}} a_j^{l_j} \\
&+ \sum_{i \neq j} \sum_{l_i,m_i} A_{l_i,m_i}^{(i)} H(l_j, m_j | l_i, m_i) a_j^{l_j} \left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij}, \phi_{ij})}{R_{ij}^{l_j+l_i+1}} + (-1)^{l_i+m_i} \beta \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j}, \phi_{\bar{i}j})}{R_{\bar{i}j}^{l_j+l_i+1}} \right].
\end{aligned} \tag{2.2.44}$$

Similarly, the second boundary condition, from Eq. (2.1.4), requires the perpendicular component of the electric displacement field to be continuous at the surface of sphere j, as the sphere is neutrally charged

$$\lim_{r_j \to a_j^-} \mathbf{D}^{\perp}(\mathbf{r}) = \lim_{r_j \to a_j^+} \mathbf{D}^{\perp}(\mathbf{r}). \tag{2.2.45}$$

Inserting the relation for the electric displacement field from Eq. (2.1.8) gives

$$\lim_{r_j \to a_j^-} \varepsilon_j \mathbf{E}^{\perp}(\mathbf{r}) = \lim_{r_j \to a_j^+} \varepsilon_+ \mathbf{E}^{\perp}(\mathbf{r}). \tag{2.2.46}$$

Furthermore, the component of the electric field perpendicular with respect to the surface of sphere j, is in a spherical coordinate system with origin in the centre of sphere j, the radial component of the electric field from Eq. (2.2.25)

$$\lim_{r_j \to a_j^-} \varepsilon_j \frac{\partial \psi_j(\mathbf{r}_j)}{\partial r_j} = \lim_{r_j \to a_j^+} \varepsilon_+ \frac{\partial \psi_+(\mathbf{r})}{\partial r_j}. \tag{2.2.47}$$

When inserting the potentials into the limits as done in Eq. (2.2.43), and differentiating with respect to r_j, the second set of limits yields

$$\begin{aligned}
&\lim_{r_j \to a_j^-} \sum_{l_j,m_j} \varepsilon_j B^{(j)}_{l_j,m_j} l_j r_j^{l_j-1} Y_{l_j}^{m_j}(\theta_j,\phi_j) = \\
&\lim_{r_j \to a_j^+} \Bigg\{ \varepsilon_+ \sum_{l_j,m_j} b_{l_j,m_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) - \sum_{l_j,m_j} \varepsilon_+ A^{(j)}_{l_j,m_j} (l_j+1) r_j^{-l_j-2} Y_{l_j}^{m_j}(\theta_j,\phi_j) \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta \varepsilon_+ A^{(j)}_{l_i,m_i} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R^{l_j+l_i+1}_{\bar{j}j}} l_j r_j^{l_j-1} Y_{l_j}^{m_j}(\theta_j,\phi_j) \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} \varepsilon_+ A^{(i)}_{l_i,m_i} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) l_j r_j^{l_j-1} Y_{l_j}^{m_j}(\theta_j,\phi_j) \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{R^{l_j+l_i+1}_{ij}} \right. \\
&\left. + (-1)^{l_i+m_i} \beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R^{l_j+l_i+1}_{\bar{i}j}} \right] \Bigg\}.
\end{aligned} \tag{2.2.48}$$

The weak boundary condition

$$\int_0^{\pi} d\theta_j \sin\theta_j \int_0^{2\pi} d\phi_j \left[Y_{l'}^{m'}(\theta_j,\phi_j) \right]^* \left[\lim_{r_j \to a_j^-} \varepsilon_j \frac{\partial \psi_j(\mathbf{r}_j)}{\partial r_j} - \lim_{r_j \to a_j^+} \varepsilon_+ \frac{\partial \psi_+(\mathbf{r})}{\partial r_j} \right] = 0, \tag{2.2.49}$$

thus takes the form

$$\begin{aligned}
&\varepsilon_j B^{(j)}_{l_j,m_j} l_j a_j^{l_j-1} = \varepsilon_+ b_{l_j,m_j} - \varepsilon_+ A^{(j)}_{l_j,m_j} (l_j+1) a_j^{-l_j-2} \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta \varepsilon_+ A^{(j)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R^{l_j+l_i+1}_{\bar{j}j}} l_j a_j^{l_j-1} \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} \varepsilon_+ A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) l_j a_j^{l_j-1} \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{R^{l_j+l_i+1}_{ij}} + (-1)^{l_i+m_i} \beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R^{l_j+l_i+1}_{\bar{i}j}} \right].
\end{aligned} \tag{2.2.50}$$

Solving Eqs. (2.2.44) and (2.2.50) for $B^{(j)}_{l_j,m_j}$ and setting them equal to each other results in

$$
\begin{aligned}
b_{l_j,m_j} a_j^{1-l_j} \left[\frac{\varepsilon_+}{l_j\varepsilon_j} - 1\right] &= A^{(j)}_{l_j,m_j} a_j^{-2l_j-1} \left[1 + \frac{\varepsilon_+(l_j+1)}{\varepsilon_j l_j}\right] \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta A^{(j)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R^{l_j+l_i+1}_{\bar{j}j}} \left[1 - \frac{\varepsilon_+}{\varepsilon_j}\right] \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{R^{l_j+l_i+1}_{ij}} + (-1)^{l_i+m_i}\beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R^{l_j+l_i+1}_{\bar{i}j}}\right] \left[1 - \frac{\varepsilon_+}{\varepsilon_j}\right].
\end{aligned}
\tag{2.2.51}
$$

The coefficients b_{l_j,m_j} in Eq. (2.2.21) cause the left-hand-side to be non zero only for $l_j = 1$. A Kronecker delta can thus be used in order to insert 1 into l_j on the left-hand-side. Dividing both sides by $[1 - \varepsilon_+/\varepsilon_j]$ then leads to the same result as Ref. [2]

$$
\begin{aligned}
-b_{1,m_j}\delta_{1l_j} &= A^{(j)}_{l_j,m_j} a_j^{-2l_j-1} \left[\frac{l_j\varepsilon_j + \varepsilon_+(l_j+1)}{l_j(\varepsilon_j - \varepsilon_+)}\right] \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta A^{(j)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R^{l_j+l_i+1}_{\bar{j}j}} \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{R^{l_j+l_i+1}_{ij}} + (-1)^{l_i+m_i}\beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R^{l_j+l_i+1}_{\bar{i}j}}\right].
\end{aligned}
\tag{2.2.52}
$$

Using vector notation, Eq. (2.2.52) can be shortened down to

$$
b_k = \tilde{C}^{(j)}_k A^{(j)}_k + \sum_{l=0}^{\infty} \bar{C}^{(j)}_{k,l} A^{(j)}_l + \sum_{i\neq j} \sum_{l=0}^{\infty} \hat{C}^{(i,j)}_{k,l} A^{(i)}_l. \tag{2.2.53}
$$

The $A^{(i)}_l$ coefficients and the C coefficients are here zero indexed with respect to k and l. The index k replaces l_j and m_j, and the index l replaces l_i and m_i. The quantum numbers l and m can be found from the indices using the conventions

$$
l(k) = \left\lfloor \sqrt{k+1} \right\rfloor, \quad m(k) = k + 1 - l(k)[l(k)+1], \tag{2.2.54}
$$

where $\lfloor\ \rfloor$ is the floor function returning the greatest integer less than or equal to its argument. The vector elements on the left-hand-side are

$$
b_k = -b_{1,m_j(k)}\delta_{1l_j(k)}, \tag{2.2.55}
$$

and the three types of C coefficients can be expressed as

$$\tilde{C}_k^{(j)} = a_j^{-2l_j(k)-1}\left[\frac{l_j(k)\varepsilon_j + \varepsilon_+(l_j(k)+1)}{l_j(k)(\varepsilon_j-\varepsilon_+)}\right]$$

$$\bar{C}_{k,l}^{(j)} = (-1)^{l_i(l)+m_i(l)}\beta H\left(l_j(k), m_j(k)|l_i(l), m_i(l)\right)\frac{Y_{l_i(l)+l_j(k)}^{m_i(l)-m_j(k)}(\theta_{\bar{j}j}, \phi_{\bar{j}j})}{R_{\bar{j}j}^{l_j(k)+l_i(l)+1}}$$

$$\hat{C}_{k,l}^{(i,j)} = H\left(l_j(k), m_j(k)|l_i(l), m_i(l)\right)\left[\frac{Y_{l_i(l)+l_j(k)}^{m_i(l)-m_j(k)}(\theta_{ij}, \phi_{ij})}{R_{ij}^{l_j(k)+l_i(l)+1}} + (-1)^{l_i(l)+m_i(l)}\beta\frac{Y_{l_i(l)+l_j(k)}^{m_i(l)-m_j(k)}(\theta_{\bar{i}j}, \phi_{\bar{i}j})}{R_{\bar{i}j}^{l_j(k)+l_i(l)+1}}\right]. \tag{2.2.56}$$

When solving equation (2.2.52) for all l_j and m_j at the surface of all N spheres, equation (2.2.53) compresses into

$$\mathbf{b} = \mathbf{C}\mathbf{A}, \tag{2.2.57}$$

where the infinite vectors are defined as

$$\mathbf{b} \equiv \begin{pmatrix} -b_{1,-1} \\ -b_{1,0} \\ -b_{1,1} \\ 0 \\ \vdots \\ -b_{1,-1} \\ -b_{1,0} \\ -b_{1,1} \\ 0 \\ \vdots \end{pmatrix}, \quad \text{and} \quad \mathbf{A} \equiv \begin{pmatrix} A_{1,-1}^{(1)} \\ A_{1,0}^{(1)} \\ A_{1,1}^{(1)} \\ A_{2,-2}^{(1)} \\ \vdots \\ A_{1,-1}^{(2)} \\ A_{1,0}^{(2)} \\ A_{1,1}^{(2)} \\ A_{2,-2}^{(2)} \\ \vdots \end{pmatrix}, \tag{2.2.58}$$

and the infinite square matrix is

$$\mathbf{C} \equiv \begin{pmatrix} \bar{C}_{0,0}^{(1)} + \tilde{C}_0^{(1)} & \bar{C}_{0,1}^{(1)} & \cdots & \hat{C}_{0,0}^{(2,1)} & \cdots & \hat{C}_{0,0}^{(N,1)} & \cdots \\ \bar{C}_{1,0}^{(1)} & \bar{C}_{1,1}^{(1)} + \tilde{C}_1^{(1)} & \cdots & \hat{C}_{1,0}^{(2,1)} & \cdots & \hat{C}_{1,0}^{(N,1)} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ \hat{C}_{0,0}^{(1,2)} & \hat{C}_{0,1}^{(1,2)} & \cdots & \bar{C}_{0,0}^{(2)} + \tilde{C}_0^{(2)} & \cdots & \hat{C}_{0,0}^{(N,2)} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ \hat{C}_{0,0}^{(1,N)} & \hat{C}_{0,1}^{(1,N)} & \cdots & \hat{C}_{0,0}^{(2,N)} & \cdots & \bar{C}_{0,0}^{(N)} + \tilde{C}_0^{(N)} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \end{pmatrix}. \tag{2.2.59}$$

The coefficients $A_{0,0}^{(i)}$ are excluded from $\mathbf{A}$. Consequently, the term $-\mathbf{R}_j \cdot \mathbf{E}_0\delta_{0,l_j}$ in Eq. (2.2.44) has also been excluded from the equations not regarding $l_j = 0$, but is still used for determining the coefficient $B_{0,0}^{(j)}$ in Eq. (2.2.44). The exclusion of $A_{0,0}^{(i)}$ is due to the normal derivative of the term $B_{0,0}^{(i)}Y_0^0(\theta_i, \phi_i)$ vanishing in Eq. (2.2.48). Hence, the coefficient $A_{0,0}^{(i)}$ has to be zero, which is related to the nanoparticles carrying zero net charge [2]. The denominator in equation (2.2.52) would also

have blown up if l_j had been allowed to be zero. Finally, the coefficient vector $\mathbf{A}$ can be represented as

$$\mathbf{A} = \mathbf{C}^{-1}\mathbf{b}. \tag{2.2.60}$$

2.2.9 Expanding the finite set of spheres to an infinite periodic lattice

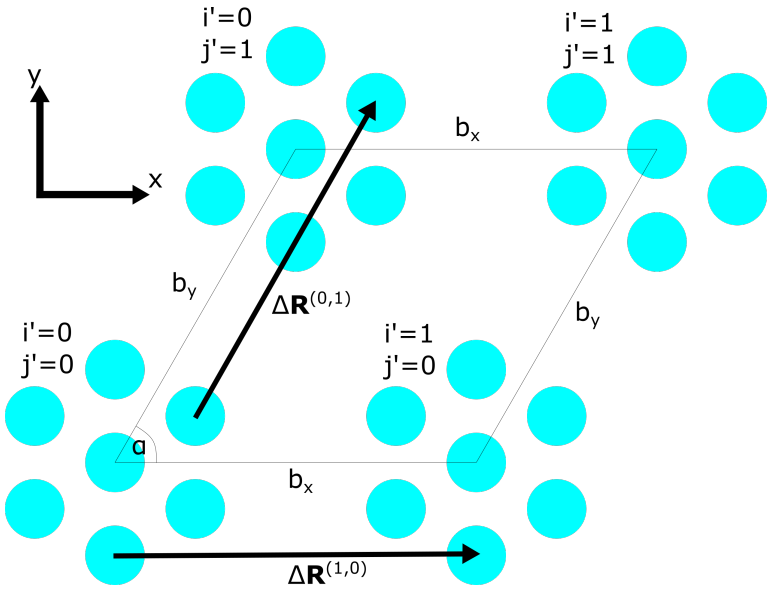


FIGURE 2.5: A section of four unit cells from an infinite periodic lattice. In this example the unit cell exists of 7 nanospheres in a formation of a hexagon with the seventh sphere placed in the centre. The lattice constant b_x is oriented along the x-axis and determines the distance between each unit cell at the same lattice coordinate j'. The lattice constant b_y on the other hand, is oriented an angle α up from the x-axis and determines the distance between each unit cell at the same lattice coordinate i'. The vector $\Delta\mathbf{R}^{(i',j')}$ is the position vector of lattice point (i', j') in a coordinate system with lattice point $(0, 0)$ as origin.

The finite set of N spheres can be repeated periodically as the unit cell of an infinite lattice as illustrated in Fig. 2.5. The lattice points are denoted as (i', j'), and the lattice constants are denoted as b_x and b_y. The lattice constant b_x is the distance between two unit cells positioned at lattice points (i', j') and $(i' + 1, j')$. Similarly, the lattice constant b_y is the distance between two unit cells positioned at lattice points (i', j') and $(i', j' + 1)$. Here, the lattice points at the same lattice coordinate i' fall on a line parallel to the x-axis, while the lattice points sharing the same lattice coordinate j' fall on a line oriented an angle α up from the x-axis. The position vector $\mathbf{R}^{(i',j')}$ is the position of the centre of the unit cell at lattice point (i', j'). The relative position vector with origin in the centre of the unit cell at lattice point $(0, 0)$ becomes

$$\Delta\mathbf{R}^{(i',j')} \equiv \mathbf{R}^{(i',j')} - \mathbf{R}^{(0,0)} = (i'b_x + j'b_y \cos\alpha, j'b_y \sin\alpha, 0). \tag{2.2.61}$$

Furthermore, a position vector with origin in the centre of sphere i, in the unit cell at lattice point (i', j'), is denoted as

$$\mathbf{r}_i^{(i',j')} = \mathbf{r}_i - \mathbf{R}^{(i',j')} = \mathbf{r} - (\mathbf{R}^{(i',j')} + \mathbf{R}_i) \equiv \mathbf{r} - \mathbf{R}_i^{(i',j')}, \tag{2.2.62}$$

where $\mathbf{R}_i^{(i',j')}$ is the position vector of sphere i in the unit cell at lattice point (i', j'). Note that whenever the lattice point index (i', j') is dropped it refers to the lattice point $(0, 0)$

$$\mathbf{r}_i \equiv \mathbf{r}_i^{(0,0')}, \quad \mathbf{R}_i \equiv \mathbf{R}_i^{(0,0)}, \mathbf{R}_{ij} \equiv \mathbf{R}_{ij}^{(0,0)}, \quad \text{etc.} \tag{2.2.63}$$

The position vector with origin in the image multipole of sphere i in the unit cell at lattice point (i', j') is denoted as $\mathbf{r}_{\bar{i}}^{(i',j')}$, and the position of that image multipole is $\mathbf{R}_{\bar{i}}^{(i',j')}$. The components of $\mathbf{r}_{\bar{i}}^{(i',j')}$ in spherical coordinates are $r_{\bar{i}}^{(i',j')}$, $\theta_{\bar{i}}^{(i',j')}$ and $\phi_{\bar{i}}^{(i',j')}$, and the components of $\mathbf{R}_{\bar{i}}^{(i',j')}$ in spherical coordinates are $R_{R,\bar{i}}^{(i',j')}$, $\theta_{R,\bar{i}}^{(i',j')}$ and $\phi_{R,\bar{i}}^{(i',j')}$. Similarly for $\mathbf{r}_{\bar{i}}^{(i',j')}$ and $\mathbf{R}_{\bar{i}}^{(i',j')}$. Lastly, the position vector of sphere i in the unit cell positioned at lattice point (i', j'), relative to the centre of sphere j in the unit cell positioned at lattice point $(0, 0)$, is denoted as

$$\mathbf{R}_{ij}^{(i',j')} \equiv \mathbf{R}_i^{(i',j')} - \mathbf{R}_j = \Delta\mathbf{R}^{(i',j')} + \mathbf{R}_{ij}, \tag{2.2.64}$$

and its components in spherical coordinates are denoted as $R_{ij}^{(i',j')}$, $\theta_{ij}^{(i',j')}$ and $\phi_{ij}^{(i',j')}$. Similarly for the position vector of the image multipole from sphere i in the unit cell positioned at lattice point (i', j'), relative to the centre of sphere j in the unit cell positioned at lattice point $(0, 0)$, $\mathbf{R}_{\bar{i}j}^{(i',j')}$. For an infinite and periodic lattice as in Fig. 2.5, the Bloch-Floquet theorem [3] can be applied. Thus, the potential from sphere i at the unit cell positioned at lattice point (i', j'), equals the potential from sphere i in the unit cell positioned at lattice point $(0, 0)$, multiplied by a phase factor

$$\psi_{\bar{i}}^{(i',j')}(\mathbf{r}_{\bar{i}}^{(i',j')}) = \psi_i^{(0,0)}(\mathbf{r}_i^{(i',j')})\mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}}. \tag{2.2.65}$$

The component of the incident wave vector parallel to the substrate, in the phase factor, can be written as

$$\mathbf{k}_\parallel = \frac{\omega}{c}\sin\theta(\cos\phi, \sin\phi, 0), \tag{2.2.66}$$

with c as the speed of light and θ and ϕ as the polar and azimuthal angle of the incident light, respectively. Here, the position vectors and the wave length of the plane wave use two different length scales. The position vectors use a length scale with the radius of a certain sphere as unit length. The wave length on the other hand, uses the standard SI scale as its length scale, as the plane wave energies are given in the unit electron volt. In order to include the angular frequency of the incident wave vector, a value for the radius used as unit length for the position vectors must be chosen.

With the shorthand notation $\sum_{i',l'} = \sum_{i'=-\infty}^{\infty}\sum_{j'=-\infty}^{\infty}$, the potential in the two regions from Eqs. (2.2.22)–(2.2.23) then become

$$\begin{aligned}\psi_+(\mathbf{r}) &= \psi_0(\mathbf{r}) + \sum_{i',j'}\sum_{j=1}^{N}\psi_j^{(i',j')}(\mathbf{r}_j^{(i',j')}) + \sum_{i',j'}\sum_{j=1}^{N}\psi_{\bar{j}}^{(i',j')}(\mathbf{r}_{\bar{j}}^{(i',j')}) \\ &= \psi_0(\mathbf{r}) + \sum_{i',j'}\mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}}\sum_{j=1}^{N}\left[\psi_j^{(0,0)}(\mathbf{r}_j^{(i',j')}) + \psi_{\bar{j}}^{(0,0)}(\mathbf{r}_{\bar{j}}^{(i',j')})\right],\end{aligned} \tag{2.2.67}$$

and

$$\psi_-(\mathbf{r}) = \psi_0^T(\mathbf{r}) + \sum_{i',j'}\sum_{j=1}^{N}\psi_j^{T,(i',j')}(\mathbf{r}_j^{(i',j')}) = \psi_0^T(\mathbf{r}) + \sum_{i',j'}\mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}}\sum_{j=1}^{N}\psi_j^{T,(0,0)}(\mathbf{r}_j^{(i',j')}). \tag{2.2.68}$$

Similarly, the electric field in the regions from Eqs. (2.2.30)–(2.2.31) becomes

$$\begin{aligned}\mathbf{E}_+(\mathbf{r}) &= \mathbf{E}_0 + \sum_{i',j'}\sum_{j=1}^{N} \mathbf{\Omega}(\theta_{R,j}^{(i',j')}, \phi_{R,j}^{(i',j')})\mathbf{E}_j^{(i',j')}(\mathbf{r}_j^{(i',j')}) + \sum_{i',j'}\sum_{j=1}^{N} \mathbf{\Omega}(\theta_{R,\bar{j}}^{(i',j')}, \phi_{R,\bar{j}}^{(i',j')})\mathbf{E}_{\bar{j}}^{(i',j')}(\mathbf{r}_{\bar{j}}^{(i',j')}) \\ &= \mathbf{E}_0 + \sum_{i',j'} \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{j=1}^{N} \left[\mathbf{\Omega}(\theta_{R,j}^{(i',j')}, \phi_{R,j}^{(i',j')})\mathbf{E}_j^{(0,0)}(\mathbf{r}_j^{(i',j')}) + \mathbf{\Omega}(\theta_{R,\bar{j}}^{(i',j')}, \phi_{R,\bar{j}}^{(i',j')})\mathbf{E}_{\bar{j}}^{(0,0)}(\mathbf{r}_{\bar{j}}^{(i',j')})\right],\end{aligned} \tag{2.2.69}$$

and

$$\begin{aligned}\mathbf{E}_-(\mathbf{r}) &= \mathbf{E}_0^T + \sum_{i',j'}\sum_{j=1}^{N} \mathbf{\Omega}(\theta_{R,j}^{(i',j')}, \phi_{R,j}^{(i',j')})\mathbf{E}_j^{T,(i',j')}(\mathbf{r}_j^{(i',j')}) \\ &= \mathbf{E}_0^T + \sum_{i',j'} \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{j=1}^{N} \mathbf{\Omega}(\theta_{R,j}^{(i',j')}, \phi_{R,j}^{(i',j')})\mathbf{E}_j^{T,(0,0)}(\mathbf{r}_j^{(i',j')}).\end{aligned} \tag{2.2.70}$$

2.2.10 Determining the expansion coefficients for an infinite periodic set of spheres from the boundary conditions

Due to the interaction with the image multipoles, the symmetry breaks with the xy-plane. For this reason, there is not much to be gained by the use of a Fourier representation of the lattice sums, even though the lattice has a discrete translational symmetry [3]. Instead, the approach from Sec. 2.2.8 has to be used. However, as a result of the Bloch-Floquet theorem from Eq. (2.2.65), only the A coefficients of the N spheres in the unit cell at lattice point $(0,0)$ need to be solved for. Then, the A coefficients for the rest of the spheres can be found from the Bloch-Floquet theorem. Starting with the first boundary condition from Sec. 2.2.8, the potential has to be continuous across the surface of sphere j as in Eq. (2.2.35). Inserting the potential inside sphere j on the left-hand-side and the potential in the ambient medium from Eq. (2.2.67) on the right-hand-side, as in Eq. (2.2.36), leads to

$$\begin{aligned}\lim_{r_j\to a_j^-} \psi_j^{(0,0)}(\mathbf{r}_j) &= \lim_{r_j\to a_j^+} \psi_+(\mathbf{r}) \\ &= \lim_{r_j\to a_j^+} \left\{\psi_0(\mathbf{r}) + \sum_{i',j'} \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \left[\sum_{j=1}^{N} \psi_j^{(0,0)}(\mathbf{r}_j^{(i',j')}) + \sum_{\bar{j}=1}^{N} \psi_{\bar{j}}^{(0,0)}(\mathbf{r}_{\bar{j}}^{(i',j')})\right]\right\}.\end{aligned} \tag{2.2.71}$$

When inserting the potentials from Eqs. (2.2.10), (2.2.11) and (2.2.18) and using the shorthand notation $\sum_{i',l'}^{'} = \sum_{(i',l')\neq(0,0)}$, the limits take the form

$$\begin{aligned}
&\lim_{r_j\to a_j^-}\sum_{l_j,m_j} B_{l_j,m_j}^{(j)} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) = \lim_{r_j\to a_j^+}\Bigg\{ r\sum_{l_j,m_j} b_{l_j,m_j} Y_{l_j}^{m_j}(\theta,\phi) \\
&+\sum_{l_j,m_j} A_{l_j,m_j}^{(j)} r_j^{-l_j-1} Y_{l_j}^{m_j}(\theta_j,\phi_j) + \sum_{l_i,m_i} A_{l_i,m_i}^{(j,R)} r_{\bar{j}}^{-l_i-1} Y_{l_i}^{m_i}(\theta_{\bar{j}},\phi_{\bar{j}}) \\
&+\sum_{i\neq j}\sum_{l_i,m_i}\Big[A_{l_i,m_i}^{(i)} r_i^{-l_i-1} Y_{l_i}^{m_i}(\theta_i,\phi_i) + A_{l_i,m_i}^{(i,R)} r_{\bar{i}}^{-l_i-1} Y_{l_i}^{m_i}(\theta_{\bar{i}},\phi_{\bar{i}})\Big] \\
&+\sum_{i',j'}^{'} \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^{N}\sum_{l_i,m_i}\Big[A_{l_i,m_i}^{(i)} (r_i^{(i',j')})^{-l_i-1} Y_{l_i}^{m_i}(\theta_i^{(i',j')},\phi_i^{(i',j')}) \\
&+A_{l_i,m_i}^{(i,R)} (r_{\bar{i}}^{(i',j')})^{-l_i-1} Y_{l_i}^{m_i}(\theta_{\bar{i}}^{(i',j')},\phi_{\bar{i}}^{(i',j')})\Big]\Bigg\}. \qquad (2.2.72)
\end{aligned}$$

Expressing the potentials in the coordinate system with origin in the centre of sphere j in the unit cell located at lattice point $(0,0)$, by using the identities from Eqs. (2.2.41) and (2.2.39) and the relation for $A_{l_i,m_i}^{(j,R)}$ from the boundary condition at the surface of the substrate from Eq. (2.2.33), the limits

$$\begin{aligned}
&\lim_{r_j\to a_j^-}\sum_{l_j,m_j} B_{l_j,m_j}^{(j)} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) = \\
&\lim_{r_j\to a_j^+}\Bigg\{ -\mathbf{R}_j\cdot\mathbf{E}_0 + r_j\sum_{l_j,m_j} b_{l_j,m_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) + \sum_{l_j,m_j} A_{l_j,m_j}^{(j)} r_j^{-l_j-1} Y_{l_j}^{m_j}(\theta_j,\phi_j) \\
&+\sum_{l_i,m_i}(-1)^{l_i+m_i}\beta A_{l_i,m_i}^{(j)} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i)\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R_{\bar{j}j}^{l_j+l_i+1}} r_j^{l_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) \\
&+\sum_{i\neq j}\sum_{l_i,m_i} A_{l_i,m_i}^{(i)} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) r_j^{l_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) \\
&\times\left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij},\phi_{ij})}{R_{ij}^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R_{\bar{i}j}^{l_j+l_i+1}}\right] \\
&+\sum_{i',j'}^{'} \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^{N}\sum_{l_i,m_i} A_{l_i,m_i}^{(i)} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) r_j^{l_j} Y_{l_j}^{m_j}(\theta_j,\phi_j) \\
&\times\left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij}^{(i',j')},\phi_{ij}^{(i',j')})}{(R_{ij}^{(i',j')})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j}^{(i',j')},\phi_{\bar{i}j}^{(i',j')})}{(R_{\bar{i}j}^{(i',j')})^{l_j+l_i+1}}\right]\Bigg\}, \qquad (2.2.73)
\end{aligned}$$

allow for fully exploiting the orthonormality of the spherical harmonics in the weak boundary condition from Eq. (2.2.38). Letting r_j approach a_j, the first set of equations for the B coefficients

then become

$$
\begin{aligned}
B^{(j)}_{l_j,m_j} a_j^{l_j} &= -\delta_{0,l_j}\mathbf{R}_j\cdot\mathbf{E}_0 + a_j^{l_j} b_{l_j,m_j} + A^{(j)}_{l_j,m_j} a_j^{-l_j-1} \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i}\beta A^{(j)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{(R_{\bar{j}j})^{l_j+l_i+1}} a_j^{l_j} \\
&+ \sum_{i\neq j}\sum_{l_i,m_i} A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) a_j^{l_j} \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{(R_{ij})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{(R_{\bar{i}j})^{l_j+l_i+1}} \right] \\
&+ {\sum_{i',j'}}' \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^{N}\sum_{l_i,m_i} A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) a_j^{l_j} \\
&\times \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta^{(i',j')}_{ij},\phi^{(i',j')}_{ij})}{(R^{(i',j')}_{ij})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta^{(i',j')}_{\bar{i}j},\phi^{(i',j')}_{\bar{i}j})}{(R^{(i',j')}_{\bar{i}j})^{l_j+l_i+1}} \right] \Bigg\}. \qquad (2.2.74)
\end{aligned}
$$

The second set of equations can be found from the second boundary condition from Sec. 2.2.8. In Eq. (2.2.45), the perpendicular component of the electric displacement field with respect to the surface of sphere j has to be continuous across the surface. Using the relation between the electric displacement field and the electric field from Eq. (2.1.8), and the relation between the potential and the electric field from Eq. (2.2.25), results in the boundary condition

$$
\lim_{r_j\to a_j^-} \varepsilon_j \frac{\partial\psi_j^{(0,0)}(\mathbf{r}_j)}{\partial r_j} = \lim_{r_j\to a_j^+} \varepsilon_+ \frac{\partial\psi_+(\mathbf{r})}{\partial r_j}. \qquad (2.2.75)
$$

Differentiating Eq. (2.2.73) with respect to r_j and multiplying by the respective permittivites

$$
\begin{aligned}
&\lim_{r_j\to a_j^-} \sum_{l_j,m_j} \varepsilon_j B^{(j)}_{l_j,m_j} l_j r_j^{l_j-1} Y^{m_j}_{l_j}(\theta_j,\phi_j) = \lim_{r_j\to a_j^+} \Bigg\{ \varepsilon_+ \sum_{l_j,m_j} b_{l_j,m_j} Y^{m_j}_{l_j}(\theta_j,\phi_j) \\
&- \sum_{l_j,m_j} \varepsilon_+ A^{(j)}_{l_j,m_j} (l_j+1) r_j^{-l_j-2} Y^{m_j}_{l_j}(\theta_j,\phi_j) \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i}\beta\varepsilon_+ A^{(j)}_{l_i,m_i} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R^{l_j+l_i+1}_{\bar{j}j}} l_j r_j^{l_j-1} Y^{m_j}_{l_j}(\theta_j,\phi_j) \\
&+ \sum_{i\neq j}\sum_{l_i,m_i} \varepsilon_+ A^{(i)}_{l_i,m_i} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) l_j r_j^{l_j-1} Y^{m_j}_{l_j}(\theta_j,\phi_j) \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{R^{l_j+l_i+1}_{ij}} \right. \\
&\left. + (-1)^{l_i+m_i}\beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R^{l_j+l_i+1}_{\bar{i}j}} \right] \\
&+ {\sum_{i',j'}}' \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^{N}\sum_{l_i,m_i} \varepsilon_+ A^{(i)}_{l_i,m_i} \sum_{l_j,m_j} H(l_j,m_j|l_i,m_i) l_j r_j^{l_j-1} Y^{m_j}_{l_j}(\theta_j,\phi_j) \\
&\times \left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta^{(i',j')}_{ij},\phi^{(i',j')}_{ij})}{(R^{(i',j')}_{ij})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y^{m_i-m_j}_{l_i+l_j}(\theta^{(i',j')}_{\bar{i}j},\phi^{(i',j')}_{\bar{i}j})}{(R^{(i',j')}_{\bar{i}j})^{l_j+l_i+1}} \right] \Bigg\}, \qquad (2.2.76)
\end{aligned}
$$

are the final preparations before the weak boundary condition from Eq. (2.2.49) can be applied

$$\int_0^\pi d\theta_j \sin\theta_j \int_0^{2\pi} d\phi_j \left[Y_{l'}^{m'}(\theta_j,\phi_j)\right]^* \left[\lim_{r_j\to a_j^-} \varepsilon_j \frac{\partial \psi_j^{(0,0)}(\mathbf{r}_j)}{\partial r_j} - \lim_{r_j\to a_j^+} \varepsilon_+ \frac{\partial \psi_+(\mathbf{r})}{\partial r_j}\right] = 0. \tag{2.2.77}$$

When r_j approaches a_j, the second set of equations for the B coefficients become

$$\begin{aligned}
&\varepsilon_j B_{l_j,m_j}^{(j)} l_j a_j^{l_j-1} = \varepsilon_+ b_{l_j,m_j} - \varepsilon_+ A_{l_j,m_j}^{(j)} (l_j+1) a_j^{-l_j-2} \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta \varepsilon_+ A_{l_i,m_i}^{(j)} H(l_j,m_j|l_i,m_i) \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R_{\bar{j}j}^{l_j+l_i+1}} l_j a_j^{l_j-1} \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} \varepsilon_+ A_{l_i,m_i}^{(i)} H(l_j,m_j|l_i,m_i) l_j a_j^{l_j-1} \left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij},\phi_{ij})}{R_{ij}^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R_{\bar{i}j}^{l_j+l_i+1}}\right] \\
&+ {\sum_{i',j'}}' \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^N \sum_{l_i,m_i} \varepsilon_+ A_{l_i,m_i}^{(i)} H(l_j,m_j|l_i,m_i) l_j a_j^{l_j-1} \\
&\times \left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij}^{(i',j')},\phi_{ij}^{(i',j')})}{(R_{ij}^{(i',j')})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j}^{(i',j')},\phi_{\bar{i}j}^{(i',j')})}{(R_{\bar{i}j}^{(i',j')})^{l_j+l_i+1}}\right].
\end{aligned} \tag{2.2.78}$$

Solving Eqs. (2.2.74) and (2.2.78) for $B_{l_j,m_j}^{(j)}$ and subtracting one from the other results in a coupled system of equations for the A coefficients

$$\begin{aligned}
&b_{l_j,m_j} a_j^{1-l_j} \left[\frac{\varepsilon_+}{l_j\varepsilon_j} - 1\right] = A_{l_j,m_j}^{(j)} a_j^{-2l_j-1} \left[1 + \frac{\varepsilon_+(l_j+1)}{\varepsilon_j l_j}\right] \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i} \beta A_{l_i,m_i}^{(j)} H(l_j,m_j|l_i,m_i) \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R_{\bar{j}j}^{l_j+l_i+1}} \left[1 - \frac{\varepsilon_+}{\varepsilon_j}\right] \\
&+ \sum_{i\neq j} \sum_{l_i,m_i} A_{l_i,m_i}^{(i)} H(l_j,m_j|l_i,m_i) \left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij},\phi_{ij})}{R_{ij}^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R_{\bar{i}j}^{l_j+l_i+1}}\right] \left[1 - \frac{\varepsilon_+}{\varepsilon_j}\right] \\
&+ {\sum_{i',j'}}' \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^N \sum_{l_i,m_i} A_{l_i,m_i}^{(i)} H(l_j,m_j|l_i,m_i) \\
&\times \left[\frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{ij}^{(i',j')},\phi_{ij}^{(i',j')})}{(R_{ij}^{(i',j')})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta \frac{Y_{l_i+l_j}^{m_i-m_j}(\theta_{\bar{i}j}^{(i',j')},\phi_{\bar{i}j}^{(i',j')})}{(R_{\bar{i}j}^{(i',j')})^{l_j+l_i+1}}\right] \left[1 - \frac{\varepsilon_+}{\varepsilon_j}\right].
\end{aligned} \tag{2.2.79}$$

Inserting a Kronecker delta for $l_j = 1$ on the left-hand-side, due to the plane wave simplification of the incident photon, and dividing by $[1 - \varepsilon_+/\varepsilon_j]$ then leads to Eq. (2.2.52)

$$
\begin{aligned}
&- b_{1,m_j}\delta_{1l_j} = A^{(j)}_{l_j,m_j} a_j^{-2l_j-1}\left[\frac{l_j\varepsilon_j + \varepsilon_+(l_j+1)}{l_j(\varepsilon_j - \varepsilon_+)}\right] \\
&+ \sum_{l_i,m_i} (-1)^{l_i+m_i}\beta A^{(j)}_{l_i,m_i} H(l_j,m_j|l_i,m_i)\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{j}j},\phi_{\bar{j}j})}{R^{l_j+l_i+1}_{\bar{j}j}} \\
&+ \sum_{i\neq j}\sum_{l_i,m_i} A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i)\left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{ij},\phi_{ij})}{R^{l_j+l_i+1}_{ij}} + (-1)^{l_i+m_i}\beta\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta_{\bar{i}j},\phi_{\bar{i}j})}{R^{l_j+l_i+1}_{\bar{i}j}}\right] \\
&+ {\sum_{i',j'}}' \mathrm{e}^{\mathbf{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} \sum_{i=1}^{N}\sum_{l_i,m_i} A^{(i)}_{l_i,m_i} H(l_j,m_j|l_i,m_i) \\
&\times\left[\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta^{(i',j')}_{ij},\phi^{(i',j')}_{ij})}{(R^{(i',j')}_{ij})^{l_j+l_i+1}} + (-1)^{l_i+m_i}\beta\frac{Y^{m_i-m_j}_{l_i+l_j}(\theta^{(i',j')}_{\bar{i}j},\phi^{(i',j')}_{\bar{i}j})}{(R^{(i',j')}_{\bar{i}j})^{l_j+l_i+1}}\right],
\end{aligned}
\tag{2.2.80}
$$

but with the contributions from all the other unit cells in the lattice included. In the same manner as in Eq. (2.2.53), Eq. (2.2.80) can be shortened down to

$$
b_k = \tilde{C}^{(j)}_k A^{(j)}_k + \sum_{l=0}^{\infty}\bar{C}^{(j)}_{k,l} A^{(j)}_l + \sum_{i\neq j}\sum_{l=0}^{\infty}\hat{C}^{(i,j)}_{k,l} A^{(i)}_l + \sum_{i=1}^{N}\sum_{l=0}^{\infty}\check{C}^{(i,j)}_{k,l} A^{(i)}_l,
\tag{2.2.81}
$$

with

$$
\begin{aligned}
\check{C}^{(i,j)}_{k,l} = {\sum_{i',j'}}' \mathrm{e}^{\mathbf{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} H\left(l_j(k),m_j(k)\middle|l_i(l),m_i(l)\right)\left[\frac{Y^{m_i(l)-m_j(k)}_{l_i(l)+l_j(k)}\left(\theta^{(i',j')}_{ij},\phi^{(i',j')}_{ij}\right)}{\left(R^{(i',j')}_{ij}\right)^{l_j(k)+l_i(l)+1}}\right. \\
\left. +(-1)^{l_i(l)+m_i(l)}\beta\frac{Y^{m_i(l)-m_j(k)}_{l_i(l)+l_j(k)}\left(\theta^{(i',j')}_{\bar{i}j},\phi^{(i',j')}_{\bar{i}j}\right)}{\left(R^{(i',j')}_{\bar{i}j}\right)^{l_j(k)+l_i(l)+1}}\right].
\end{aligned}
\tag{2.2.82}
$$

The conventions used for $l_j(k)$, $m_j(k)$, $l_i(l)$ and $m_i(l)$ are the same as from Eq. (2.2.54). When setting up the matrix system as done in Eq. (2.2.57), the $\mathbf{b}$ vector is unchanged from Eq. (2.2.58). The same unknown coefficients are also contained by the A vector, but their values will most likely differ from the ones found in Sec. 2.2.8. The $\mathbf{C}$ matrix on the other hand, can be written as the one for the finite set of spheres in the unit cell located at lattice point $(0,0)$ from Eq. (2.2.59), plus the matrix containing the contributions from the unit cells located at all the other lattice points

$$
\mathbf{C} = \mathbf{C}^{(0,0)} + \check{\mathbf{C}},
\tag{2.2.83}
$$

where

$$\check{\mathbf{C}} \equiv \begin{pmatrix} \check{C}_{0,0}^{(1,1)} & \check{C}_{0,1}^{(1,1)} & \dots & \check{C}_{0,0}^{(2,1)} & \dots & \check{C}_{0,0}^{(N,1)} & \dots \\ \check{C}_{1,0}^{(1,1)} & \check{C}_{1,1}^{(1,1)} & \dots & \check{C}_{1,0}^{(2,1)} & \dots & \check{C}_{1,0}^{(N,1)} & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ \check{C}_{0,0}^{(1,2)} & \check{C}_{0,1}^{(1,2)} & \dots & \check{C}_{0,0}^{(2,2)} & \dots & \check{C}_{0,0}^{(N,2)} & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \\ \check{C}_{0,0}^{(1,N)} & \check{C}_{0,1}^{(1,N)} & \dots & \check{C}_{0,0}^{(2,N)} & \dots & \check{C}_{0,0}^{(N,N)} & \dots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots \end{pmatrix}. \tag{2.2.84}$$

The A coefficients for the spheres within the unit cell located at lattice point $(0,0)$ can then be presented as

$$\mathbf{A}^{(0,0)} = \mathbf{C}^{-1}\mathbf{b}. \tag{2.2.85}$$

The A coefficients for the spheres within the unit cell at lattice point $(i',j') \neq (0,0)$ on the other hand, can be found by applying the Bloch-Floquet theorem from Eq. (2.2.65)

$$\mathbf{A}^{(i',j')} = \mathbf{A}^{(0,0)} \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel \cdot \Delta\mathbf{R}^{(i',j')}}. \tag{2.2.86}$$

2.2.11 Dimensionless dipole moment

The dipole moment of sphere j at angular frequency ω, $\mathbf{p}^{(j)}(\omega)$, consists of the Cartesian components

$$p_x^{(j)} = \sqrt{\frac{3}{8\pi}} \frac{A_{1,-1}^{(j)} - A_{1,1}^{(j)}}{a_j^2}, \quad p_y^{(j)} = -i\sqrt{\frac{3}{8\pi}} \frac{A_{1,-1}^{(j)} + A_{1,1}^{(j)}}{a_j^2}, \quad \text{and} \quad p_z^{(j)} = \sqrt{\frac{3}{4\pi}} \frac{A_{1,0}^{(j)}}{a_j^2}. \tag{2.2.87}$$

It may be more convenient to present the dimensionless version of the dipole moment

$$\bar{\mathbf{p}}^{(j)} = \frac{\mathbf{p}^{(j)}}{a_j^3 \varepsilon_0 E_0}. \tag{2.2.88}$$

As the dipole moment can be a complex vector quantity, it can be useful to display the modulus of the total dimensionless dipole moment

$$\bar{p}(\omega) \equiv |\bar{\mathbf{p}}(\omega)| = \sqrt{\bar{\mathbf{p}}^\dagger \bar{\mathbf{p}}}, \tag{2.2.89}$$

where † denotes the Hermitian transpose. As showed in Sec. 2.1.5, the dipole moment has a lower decay rate than the higher order multipole moments, and thus it dominates the far field potential when the net charge is neutral. This is the motivation behind the choice of computing the dipole moment.

2.2.12 Polarisation density and the effective dielectric tensor

The polarisation density of the layer of spheres, defined as the total dipole moment per unit volume in Eq. (2.1.6), can be found from the susceptibilities as in Eq. (2.1.7) [3]

$$P_k = \frac{\varepsilon_0}{(2a+h) b_x b_y \sin\alpha} \sum_{i=1}^{N} \sum_{l=1}^{3} \chi_{kl}^{(i)} E_{0,l}, \tag{2.2.90}$$

with P_k as the k-th component of $\mathbf{P}$ and $E_{0,l}$ as the l-th component of $\mathbf{E}_0$. The susceptibility tensor is diagonal with the elements

$$\chi_{xx}^{(i)} = \sqrt{\frac{3}{2\pi}} A_{1,-1}^{(i)}, \quad \chi_{yy}^{(i)} = -\mathrm{i}\sqrt{\frac{3}{2\pi}} A_{1,-1}^{(i)}, \quad \text{and} \quad \chi_{zz}^{(i)} = \sqrt{\frac{3}{4\pi}} A_{1,0}^{(i)}. \tag{2.2.91}$$

However, the elements $\chi_{kk}^{(i)}$ are only valid when the electric field is parallel to the axis k. This is the reason why the elements of $\chi^{(i)}$ differ from the ones in Eq. (2.2.87) [3]. If the polarisation density is inserted back into the electric displacement field from Eq. (2.1.5)

$$\begin{aligned} D_k \equiv \varepsilon_0 E_{0,k} + P_k &= \varepsilon_0 \left(E_{0,k} + \frac{1}{(2a+h) b_x b_y \sin\alpha} \sum_{i=1}^{N} \sum_{l=1}^{3} \chi_{kl}^{(i)} E_{0,l} \right) \\ &= \varepsilon_0 \sum_{l=1}^{3} \left(\delta_{kl} + \frac{1}{(2a+h) b_x b_y \sin\alpha} \sum_{i=1}^{N} \chi_{kl}^{(i)} \right) E_{0,l} = \varepsilon_0 \sum_{l=1}^{3} \varepsilon_{kl} E_{0,l}, \end{aligned} \tag{2.2.92}$$

the elements of the effective dielectric tensor of the lattice can be determined

$$\varepsilon_{kl} = \delta_{kl} + \frac{1}{(2a+h) b_x b_y \sin\alpha} \sum_{i=1}^{N} \chi_{kl}^{(i)}. \tag{2.2.93}$$

2.2.13 Reflectance and transmittance

The reflectance and transmittance can be defined as the ratio of reflected and transmitted energies to the incident energy, respectively [16]. When all the reflected energy is reflected in the specular direction as for the island films considered here in the quasi-static limit, the reflectivity is equal to the reflectance. The reflectance and transmittance can be expressed in terms of the amplitude of the reflected and transmitted wave, A_R and A_T [19], respectively

$$R = |A_R|^2, \tag{2.2.94}$$

$$T = \frac{n^- \cos\theta_T}{n^+ \cos\theta} |A_T|^2. \tag{2.2.95}$$

The refractive indices of the ambient medium and the substrate are denoted as n^+ and n^-, respectively. The reflection angle is equal to the incident angle, θ, and the transmission angle is given by Snell's law

$$n_- \sin\theta_T = n_+ \sin\theta. \tag{2.2.96}$$

Considering a non-magnetic system, the amplitudes can be written in terms of the electric dipole susceptibilities

$$\gamma_e = \rho \langle \alpha_\parallel \rangle, \tag{2.2.97}$$

$$\beta_e = \rho \frac{\langle \alpha_\perp \rangle}{\varepsilon_+^2}, \tag{2.2.98}$$

where γ_e and β_e are the integrated surface polarisation parallel and perpendicular to the surface, respectively, and ρ is the number of islands per unit surface area. The average dipole polarizabilities

parallel, $\alpha_\parallel$, and normal, $\alpha_\perp$, to the surface can be found from the expansion coefficients

$$\langle \alpha_\parallel \rangle = -\frac{4\pi\varepsilon_+ \langle A_{1,1} \rangle}{\sqrt{\frac{2\pi}{3}} E_0 \sin\theta_0 \exp(-\mathrm{i}\phi_0)}, \tag{2.2.99}$$

$$\langle \alpha_\perp \rangle = \frac{2\pi\varepsilon_+ \langle A_{1,0} \rangle}{\sqrt{\frac{\pi}{3}} E_0 \cos\theta_0}, \tag{2.2.100}$$

with θ_0 and ϕ_0 as the polar angle and the azimuthal angle of the electric field from the incident plane wave in Sec. 2.2.5. The incident photons can be polarised such that the orientation of the electric field is fixed. For p-polarised light, the electric field is parallel to the scattering plane spanned by the incident and reflected wave vectors. Similarly, for the s-polarised light, the electric field is perpendicular to the scattering plane and thus parallel to the substrate. The polar and azimuthal angle for the electric field for s- and p-polarised are hence related to the angles for the incident plane wave vector by

$$\theta_0^{(s)} = \frac{\pi}{2}, \qquad \phi_0^{(s)} = \phi + \frac{\pi}{2}, \tag{2.2.101}$$

$$\theta_0^{(p)} = \frac{\pi}{2} - \theta, \qquad \phi_0^{(p)} = \phi. \tag{2.2.102}$$

The amplitudes of reflected and transmitted waves expressed with the dipole susceptibilities by Bedeaux and Vlieger [19] for s-polarised light are then

$$A_R^{(s)} = \frac{n_+ \cos\theta - n_- \cos\theta_T + \frac{\mathrm{i}\omega}{c}\gamma_e}{n_+ \cos\theta + n_- \cos\theta_T - \frac{\mathrm{i}\omega}{c}\gamma_e}, \tag{2.2.103}$$

$$A_T^{(s)} = \frac{2n_+ \cos\theta}{n_+ \cos\theta + n_- \cos\theta_T - \frac{\mathrm{i}\omega}{c}\gamma_e}, \tag{2.2.104}$$

and for p-polarised light

$$A_R^{(p)} = \frac{(n_- \cos\theta - n_+ \cos\theta_T)\left[1 - \frac{\omega^2}{4c^2}\varepsilon_+\gamma_e\beta_e \sin^2\theta\right] - \frac{\mathrm{i}\omega}{c}\left[\gamma_e \cos\theta \cos\theta_T - n_+ n_- \varepsilon_+ \beta_e \sin^2\theta\right]}{(n_- \cos\theta + n_+ \cos\theta_T)\left[1 - \frac{\omega^2}{4c^2}\varepsilon_+\gamma_e\beta_e \sin^2\theta\right] - \frac{\mathrm{i}\omega}{c}\left[\gamma_e \cos\theta \cos\theta_T + n_+ n_- \varepsilon_+ \beta_e \sin^2\theta\right]}, \tag{2.2.105}$$

$$A_T^{(p)} = \frac{2n_+ \cos\theta \left[1 + \frac{\omega^2}{4c^2}\varepsilon_+\beta_e\gamma_e \sin^2\theta\right]}{(n_- \cos\theta + n_+ \cos\theta_T)\left[1 - \frac{\omega^2}{4c^2}\varepsilon_+\gamma_e\beta_e \sin^2\theta\right] - \frac{\mathrm{i}\omega}{c}\left[\gamma_e \cos\theta \cos\theta_T + n_+ n_- \varepsilon_+ \beta_e \sin^2\theta\right]}. \tag{2.2.106}$$

2.3 Resonance energies

2.3.1 The resonance energy of an isolated sphere

For the case of a single sphere in vacuum with no substrate, $\varepsilon_- = \varepsilon_+ = 1$, and hence β becomes zero. The remaining contribution left in equation (2.2.52) becomes

$$-b_{1m}\delta_{1l} = A_{lm}^{(j)} a_j^{-2l-1} \frac{l\varepsilon_j + \varepsilon_+(l+1)}{l(\varepsilon_j - \varepsilon_+)}. \tag{2.3.1}$$

Solving the equation with respect to $A_{lm}^{(j)}$ results in

$$A_{lm}^{(j)} = -b_{1m}\delta_{1l}a_j^{2l+1}\frac{l(\varepsilon_j - \varepsilon_+)}{l\varepsilon_j + \varepsilon_+(l+1)}, \tag{2.3.2}$$

which will become resonant when the real part of the denominator becomes zero

$$\mathrm{Re}\big\{l\varepsilon_j + \varepsilon_+(l+1)\big\} = 0. \tag{2.3.3}$$

Isolating the dielectric function of the sphere on the left-hand-side

$$-\mathrm{Re}\big\{\varepsilon_j\big\} = \mathrm{Re}\{\varepsilon_+\}\frac{l+1}{l}, \tag{2.3.4}$$

makes it easier to facilitate the replacement of ε_j by the Drude model expression. Inserting the dielectric function of the vacuum and the Drude model for the sphere, letting the free carrier relaxation time approach infinite, Eq. (2.3.4) becomes

$$\frac{\omega_p^2}{\omega^2} - 1 = \frac{l+1}{l}. \tag{2.3.5}$$

Solving with respect to the frequency ω gives

$$\omega = \omega_p\sqrt{\frac{l}{2l+1}}, \tag{2.3.6}$$

and the Mie result is obtained [2]. For $l = 1$ and $\hbar\omega_p = 3\,\mathrm{eV}$, the resonance energy becomes $3\sqrt{1/3}\ \mathrm{eV} = \sqrt{3}\ \mathrm{eV}$.

2.3.2 The resonance energy at dipole order for a single supported sphere

To find the resonance energy of a coefficient A_{lm} for a single sphere placed a height h above a substrate, a similar approach to the one in Sec. 2.3.1 can be used. An expression for the coefficient must be obtained, and the energies resulting in the real part of the denominator in this expression are the resonance energies. D. Bedeaux and J. Vlieger [1] provided the relation

$$A_{l,1}\exp\{i\phi_0\} = -A_{l,-1}\exp\{-i\phi_0\}, \tag{2.3.7}$$

resulting in the same resonance energies for $A_{l,1}$ and $A_{l,-1}$. This combined with the property of the spherical harmonics being nonzero only for $m = 0$ when $\theta = \pi$ reduces the problem for a single sphere to two independent equations

$$\begin{aligned}
-b_{1,1} &= A_{1,1}a^{-3}\frac{\varepsilon + 2\varepsilon_+}{\varepsilon - \varepsilon_+} + \beta A_{1,1}H(1,1|1,1)\frac{Y_2^0(\pi,0)}{8(a+h)^3},\\
-b_{1,0} &= A_{1,0}a^{-3}\frac{\varepsilon + 2\varepsilon_+}{\varepsilon - \varepsilon_+} - \beta A_{1,0}H(1,0|1,0)\frac{Y_2^0(\pi,0)}{8(a+h)^3}.
\end{aligned} \tag{2.3.8}$$

The expressions for the coefficients $A_{1,1}$ and $A_{1,0}$ hence take the form

$$\begin{aligned}
A_{1,1} &= \frac{-b_{1,1}}{a^{-3}\frac{\varepsilon+2\varepsilon_+}{\varepsilon-\varepsilon_+} + \beta H(1,1|1,1)\frac{Y_2^0(\pi,0)}{8(a+h)^3}},\\
A_{1,0} &= \frac{-b_{1,0}}{a^{-3}\frac{\varepsilon+2\varepsilon_+}{\varepsilon-\varepsilon_+} - \beta H(1,0|1,0)\frac{Y_2^0(\pi,0)}{8(a+h)^3}}.
\end{aligned} \tag{2.3.9}$$

The resonance energies for the coefficients $A_{1,1}$ and $A_{1,0}$, and thus the dipole moment, will occur where the real part of the corresponding denominator becomes zero

$$\mathrm{Re}\left\{ a^{-3}\frac{\varepsilon+2\varepsilon_+}{\varepsilon-\varepsilon_+} + \beta H(1,1|1,1)\frac{Y_2^0(\pi,0)}{8(a+h)^3} \right\} = 0,$$
$$\mathrm{Re}\left\{ a^{-3}\frac{\varepsilon+2\varepsilon_+}{\varepsilon-\varepsilon_+} - \beta H(1,0|1,0)\frac{Y_2^0(\pi,0)}{8(a+h)^3} \right\} = 0. \tag{2.3.10}$$

Assuming the real part of the dielectric function ε is much greater than the imaginary part and

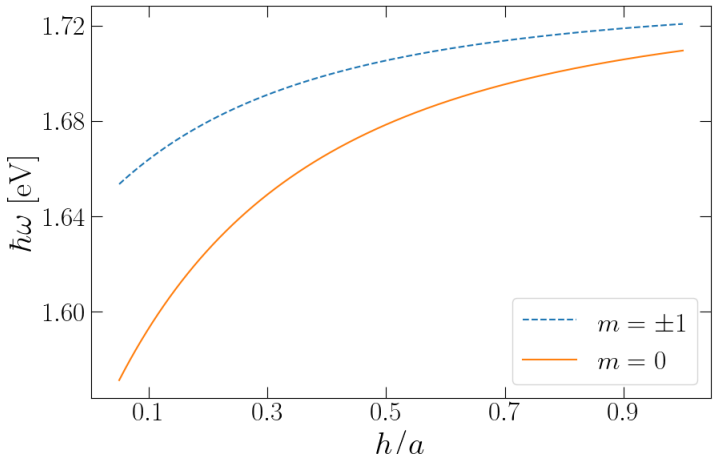


FIGURE 2.6: The energies required for the real part of the denominators of the coefficients $A_{1,0}$ and $A_{1,\pm1}$ to become zero, for a single sphere of radius a placed a height h above a substrate with dielectric function $\varepsilon_- = 10$. The dielectric function of the sphere is the Drude model with plasma frequency $\hbar\omega_p = 3\,\mathrm{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\mathrm{eV}$. The denominators are calculated analytically at dipole order, and the the resonance energies are plotted for various heights.

using the Drude model, the first term in the two denominators is a decreasing function of ω becoming zero at the Mie frequency. The second terms in the two denominators thus shift the energies for which the denominators become zero. The second term in both denominators are negative and consequently shift the resonance peaks to lower frequencies the greater the absolute value of the second term. As the second terms follow $(a+h)^{-3}$, the red shifts will be stronger for lower heights. The red shift is visualised in Fig. 2.6 for a plasma frequency $\hbar\omega_p = 3\,\mathrm{eV}$, an inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\mathrm{eV}$, and the dielectric function of the substrate $\varepsilon_- = 10$.

2.3.3 The resonance energy at dipole order for a supported dimer

For two spheres, the resonance energy equations become more intricate. The relation from Eq. (2.3.7) still applies, so the coefficients are reduced from six to four. The second sphere is placed at the same height h above the substrate as the first sphere and a distance d next to the sphere along the x-axis. If the heights, h, and dielectric functions of the two spheres, ε, are set equal, the only differences in the terms for the two spheres in Eq. (2.2.52) are in the spherical harmonics. The first sphere's azimuthal angle in the second sphere's coordinate system is rotated by 180°. The spherical harmonics for the second sphere are thus related to the ones for the first sphere by

$$Y_l^m(\theta,\pi) = (-1)^m Y_l^m(\theta,0). \tag{2.3.11}$$

This results in all the interactions with odd $m = m_i - m_j$ to switch sign. The interactions are the ones between $A^{(0)}_{1,\pm1}$ and $A^{(1)}_{1,0}$, and between $A^{(0)}_{1,0}$ and $A^{(1)}_{1,\pm1}$. As a consequence of the two identical set of equations, except for those interactions with flipped signs, the coefficients of the second sphere can be related to the ones in the first sphere by

$$A^{(1)}_{l,m} = (-1)^{m+1} A^{(0)}_{l,m}. \tag{2.3.12}$$

Hence, the problem is reduced to a coupled pair of equations with two coefficients to solve for

$$\begin{aligned} -b_{1,1} = {} & A_{1,1} a^{-3} \frac{\varepsilon + 2\varepsilon_+}{\varepsilon - \varepsilon_+} + \beta A_{1,1} H(1,1|1,1) \frac{Y_2^0(\pi,0)}{R_\perp^3} \\ & + (-1) A_{1,0} H(1,1|1,0) \left[\frac{Y_2^{-1}(\frac{\pi}{2},0)}{R_\parallel^3} - \beta \frac{Y_2^{-1}(\bar\theta,0)}{\bar R_\parallel^3} \right] \\ & + A_{1,1} H(1,1|1,1) \left[\frac{Y_2^{0}(\frac{\pi}{2},0)}{R_\parallel^3} + \beta \frac{Y_2^{0}(\bar\theta,0)}{\bar R_\parallel^3} \right] \\ & - A_{1,1} H(1,1|1,-1) \left[\frac{Y_2^{-2}(\frac{\pi}{2},0)}{R_\parallel^3} + \beta \frac{Y_2^{-2}(\bar\theta,0)}{\bar R_\parallel^3} \right], \end{aligned} \tag{2.3.13}$$

and

$$\begin{aligned} -b_{1,0} = {} & A_{1,0} a^{-3} \frac{\varepsilon + 2\varepsilon_+}{\varepsilon - \varepsilon_+} + \beta A_{1,0} H(1,0|1,0) \frac{Y_2^0(\pi,0)}{R_\perp^3} \\ & + (-1) A_{1,0} H(1,0|1,0) \left[\frac{Y_2^{0}(\frac{\pi}{2},0)}{R_\parallel^3} - \beta \frac{Y_2^{0}(\bar\theta,0)}{\bar R_\parallel^3} \right] \\ & + A_{1,1} [1 + (-1)^{-1+1}] H(1,0|1,1) \left[\frac{Y_2^{1}(\frac{\pi}{2},0)}{R_\parallel^3} + \beta \frac{Y_2^{1}(\bar\theta,0)}{\bar R_\parallel^3} \right], \end{aligned} \tag{2.3.14}$$

with

$$R_\perp = 2(a+h), \quad R_\parallel = 2a + d, \quad \bar R_\parallel = \sqrt{R_\perp^2 + R_\parallel^2}, \quad \text{and} \quad \bar\theta = \arctan\left(\frac{R_\parallel}{-R_\perp} \right). \tag{2.3.15}$$

Here, $R_\perp$ represents the distance from the sphere's centre to its image multipole, $R_\parallel$ is the distance between the two sphere's centres, and $\bar R_\parallel$ is the distance between a sphere's centre and its neighbour's image multipole. The angle $\bar\theta$ is the polar angle of the neighbours image multipole. The term $(-1)^{-1+1}$ in Eq. (2.3.14) comes from Eq. (2.3.7) and from

$$Y_l^{-m}(\theta,\phi) = -\left[Y_l^m(\theta,\phi) \right]^*, \tag{2.3.16}$$

and

$$H(1,0|1,-1) = H(1,0|1,1). \tag{2.3.17}$$

The values for the spherical harmonics are all real in this case, so the conjugate part is dropped in

Eq. (2.3.14). The two coupled equations can be shortened down to

$$\begin{aligned} b_1 &= c_{10}A_{1,0} + c_{11}A_{1,1} \\ b_0 &= c_{00}A_{1,0} + c_{01}A_{1,1}, \end{aligned} \tag{2.3.18}$$

where

$$\begin{aligned} b_0 &= -b_{1,0} \\ b_1 &= -b_{1,1} \\ c_{00} &= a^{-3}\frac{\varepsilon + 2\varepsilon_+}{\varepsilon - \varepsilon_+} + \beta H(1,0|1,0)\frac{Y_2^0(\pi,0)}{R_\perp^3} - H(1,0|1,0)\left[\frac{Y_2^0(\frac{\pi}{2},0)}{R_\parallel^3} - \beta\frac{Y_2^0(\bar{\theta},0)}{\bar{R}_\parallel^3}\right] \\ c_{01} &= 2H(1,0|1,1)\left[\frac{Y_2^1(\frac{\pi}{2},0)}{R_\parallel^3} + \beta\frac{Y_2^1(\bar{\theta},0)}{\bar{R}_\parallel^3}\right] \\ c_{10} &= -H(1,0|1,0)\left[\frac{Y_2^0(\frac{\pi}{2},0)}{R_\parallel^3} - \beta\frac{Y_2^0(\bar{\theta},0)}{\bar{R}_\parallel^3}\right] \\ c_{11} &= a^{-3}\frac{\varepsilon + 2\varepsilon_+}{\varepsilon - \varepsilon_+} + \beta H(1,1|1,1)\frac{Y_2^0(\pi,0)}{R_\perp^3} \\ &\quad + H(1,1|1,1)\left[\frac{Y_2^0(\frac{\pi}{2},0)}{R_\parallel^3} + \beta\frac{Y_2^0(\bar{\theta},0)}{\bar{R}_\parallel^3}\right] - H(1,1|1,-1)\left[\frac{Y_2^{-2}(\frac{\pi}{2},0)}{R_\parallel^3} + \beta\frac{Y_2^{-2}(\bar{\theta},0)}{\bar{R}_\parallel^3}\right]. \end{aligned} \tag{2.3.19}$$

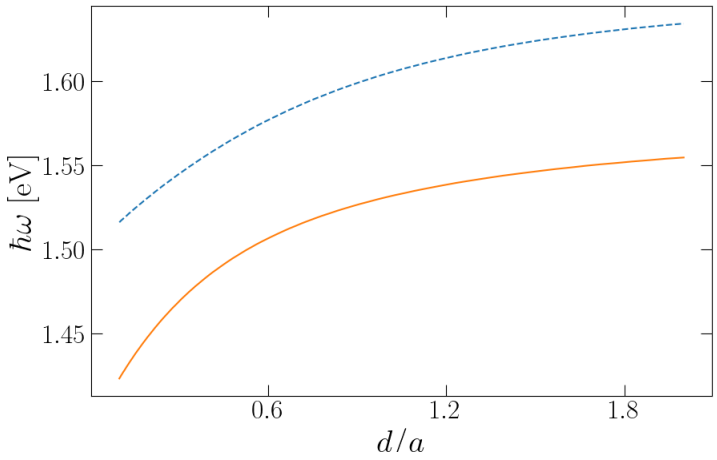


FIGURE 2.7: The energy positions at the two minimums of the modulus of the denominator of the coefficients $A_{1,m}$, for two spheres of radius a placed a height $h = 0.05a$ above a substrate with dielectric function $\varepsilon_- = 10$. The dielectric function of the spheres is the Drude model with plasma frequency $\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\gamma = 0.03\,\text{eV}$. The denominator is calculated analytically at dipole order, and the the resonance energies are plotted for various distances d separating the two spheres. The dashed line lies close to the real part of the denominator being zero.

The solutions to Eq. (2.3.18) yields

$$A_{1,0} = \frac{c_{11}b_0 - c_{01}b_1}{c_{00}c_{11} - c_{01}c_{10}}$$
$$A_{1,1} = \frac{c_{00}b_1 - c_{10}b_0}{c_{00}c_{11} - c_{01}c_{10}}. \tag{2.3.20}$$

The coefficients $A_{1,0}$ and $A_{1,1}$ thus have equal denominators. To find the resonance energies, it is no longer sufficient to find the roots of the real part of the denominator. The minimums of the modulus of the denominator now have to be found in order to determine the resonance energies for the dimensionless dipole moment

$$\min |c_{00}c_{11} - c_{01}c_{10}|. \tag{2.3.21}$$

The two minimums are presented in Fig. 2.7, where the plasma frequency is $\hbar\omega_p = 3\,\mathrm{eV}$ and the inverse of the free carrier relaxation time is $\hbar\gamma = 0.03\,\mathrm{eV}$. They are red shifted when decreasing the distance d separating the two spheres as in Fig. 2.6 when decreasing the height h above the substrate. The dashed line lies close to the root of the real part of the denominator in Eq. (2.3.20), but in order to find the second resonance energy position the approach in Eq. (2.3.21) was necessary.

2.4 The Discrete dipole approximation

2.4.1 System of equations

The discrete dipole approximation (DDA) [20] deals with scattering and absorption problems by discretising the volume of the scattering object into a set of cubical subvolumes of volume d^3. The subvolumes can also be referred to as point dipoles on a grid with inter-dipole distance d. Hence, the scattering objects can have arbitrary shapes, as opposed the special geometries needed in order to obtain exact solutions to Maxwell's equations. To solve the scattering and absorption problems, the dipole moments, also referred to as polarisations, of the point dipoles must be determined from the set of linear equations which arise from the interaction between the point dipoles and the incident field. There are multiple methods developed taking advantage of the DDA, and the equations and unknowns to solve for depend on the chosen method. However, the final equations for the various derivations are essentially the same [20]. The method developed by B. T. Draine and P. J. Flatau [7] solves for the unknown polarisations from Eq. (2.1.2)

$$\mathbf{p}_j = \alpha_j \mathbf{E}(\mathbf{R}_j), \tag{2.4.1}$$

where α_j and $\mathbf{p}_j$ are the polarisability and dipole moment of the subvolume at position $\mathbf{R}_j$, respectively. The total electric field evaluated inside the subvolume j consists of two main contributions and can be written in the following form

$$\mathbf{E}(\mathbf{R}_j) = \mathbf{E}_0(\mathbf{R}_j) + \sum_{k \neq j} \mathbf{G}(\mathbf{R}_{jk})\mathbf{p}_k. \tag{2.4.2}$$

The first term on the right-hand-side represents the electric field from the time dependent incident plane wave

$$\mathbf{E}_0(\mathbf{R}_j) = \mathbf{E}_0 \exp\big(\mathrm{i}\mathbf{k}_0 \cdot \mathbf{R}_j - \mathrm{i}\omega t\big), \tag{2.4.3}$$

and the second term represents the scattered electric field from all the other $N-1$ point dipoles. The wave vector of the incident wave is $\mathbf{k}_0 \equiv \hat{\mathbf{k}}_0\omega/c$, with $\omega \equiv 2\pi/\lambda$ as the angular frequency, λ as

the wave length and c as the speed of light. The contribution to the electric field in subvolume j, due to interaction with dipole k, including retardation effects, is $\mathbf{G}(\mathbf{R}_{jk})\mathbf{p}_k$. The interaction matrix, with dimensions 3×3, is also known as the free space dyadic Green's function [20]

$$\begin{aligned}\mathbf{G}(\mathbf{R}_{jk}) &= \left[\mathbf{I}_3 + \frac{1}{k_0^2}\nabla^2\right]\frac{\exp\bigl(\mathrm{i}k_0 R_{jk}\bigr)}{R_{jk}} \\ &= \frac{\exp\bigl(\mathrm{i}k_0 R_{jk}\bigr)}{R_{jk}}\left[k_i^2\left(\mathbf{I}_3 - \hat{\mathbf{R}}_{jk}\hat{\mathbf{R}}_{jk}\right) - \frac{1-\mathrm{i}k_0 R_{jk}}{R_{jk}^2}\left(\mathbf{I}_3 - 3\hat{\mathbf{R}}_{jk}\hat{\mathbf{R}}_{jk}\right)\right], \quad j \neq k. \end{aligned} \tag{2.4.4}$$

The position of point dipole at position $\mathbf{R}_j$ relative to the point dipole at position $\mathbf{R}_k$ is denoted as

$$\mathbf{R}_{jk} = \mathbf{R}_j - \mathbf{R}_k, \tag{2.4.5}$$

with its length $R_{jk} \equiv |\mathbf{R}_{jk}|$ and its unit vector $\hat{\mathbf{R}}_{jk} \equiv \mathbf{R}_{jk}/R_{jk}$. Lastly, the 3×3 identity dyad is written as $\mathbf{I}_3$. Inserting Eq. (2.4.2) for the electric field at point dipole j into Eq. (2.4.1) and dividing by the polarisability α_j leads to

$$\alpha_j^{-1}\mathbf{p}_j = \mathbf{E}_0(\mathbf{R}_j) + \sum_{k\neq j}\mathbf{G}(\mathbf{R}_{jk})\mathbf{p}_k. \tag{2.4.6}$$

If defining

$$\mathbf{G}(\mathbf{R}_{jj}) \equiv -\alpha_j^{-1}, \tag{2.4.7}$$

the equations to solve for $\mathbf{p}$ become

$$-\mathbf{E}_0(\mathbf{R}_j) = \mathbf{G}(\mathbf{R}_{jj})\mathbf{p}_j + \sum_{k\neq j}\mathbf{G}(\mathbf{R}_{jk})\mathbf{p}_k = \sum_{k=1}^{N}\mathbf{G}(\mathbf{R}_{jk})\mathbf{p}_k. \tag{2.4.8}$$

As the polarisations consist of three components, there are $3N$ complex linear equations when solving Eq. (2.4.8) at all point dipoles j. For periodic structures, the Bloch-Floquet theorem from Eq. (2.2.65) can be applied

$$\mathbf{p}_k^{(m,n)} = \mathbf{p}_k^{(0,0)}\exp\bigl[\mathrm{i}(m\mathbf{k}_0\cdot\mathbf{L}_u + n\mathbf{k}_0\cdot\mathbf{L}_v)\bigr], \tag{2.4.9}$$

where m is the lattice coordinate along lattice vector $\mathbf{L}_u = L_u\hat{\mathbf{u}}$, and n is the lattice coordinate along lattice vector $\mathbf{L}_v = L_v\hat{\mathbf{v}}$. The interaction matrix including the contribution from point dipole k at all the lattice sites (m,n) can then be written as

$$\bar{\mathbf{G}}(\mathbf{R}_{jk}) = \sum_{m,n}\mathbf{G}(\mathbf{R}_{jk}^{(m,n)})\exp\bigl[\mathrm{i}(m\mathbf{k}_0\cdot\mathbf{L}_u + n\mathbf{k}_0\cdot\mathbf{L}_v)\bigr], \tag{2.4.10}$$

where the position vector $\mathbf{R}_{jk}^{(m,n)} = \mathbf{R}_j^{(0,0)} - \mathbf{R}_k^{(m,n)}$ is the position of point dipole j at lattice site $(0,0)$ relative to point dipole k at lattice site (m,n), and only $\mathbf{G}(\mathbf{R}_{jj}^{(0,0)}) \equiv -\alpha_j^{-1}$. Equation (2.4.8) expanded to an infinite periodic set of point dipoles thus yields

$$-\mathbf{E}_0(\mathbf{R}_j) = \sum_{k=1}^{N}\bar{\mathbf{G}}(\mathbf{R}_{jk})\mathbf{p}_k. \tag{2.4.11}$$

When the unknown polarisations $\mathbf{p}$ are solved for, the electric near field can be determined [9]

$$\mathbf{E}(\mathbf{R}_j) = \begin{cases} \alpha_j^{-1}\mathbf{p}_j, & \text{lattice site } j \text{ is occupied,} \\ \mathbf{E}_0(\mathbf{R}_j) + \sum_{k=1}^{N} \bar{\mathbf{G}}(\mathbf{R}_{jk})\mathbf{p}_k, & \text{lattice site } j \text{ is unoccupied.} \end{cases} \tag{2.4.12}$$

Only the polarisations for the occupied lattice sites are non zero and hence included in the summation of the scattered electric field.

2.4.2 Polarizabilities

There are several approximations for the polarizabilities, α_j, used in Eq. (2.4.7), and the first DDA methods used the Clausius-Mosotti (CM) polarizabilities [7]

$$\alpha_j^{\text{CM}} = \frac{3d^3}{4\pi}\frac{\varepsilon_j - 1}{\varepsilon_j + 2}, \tag{2.4.13}$$

with d^3 as the cubic subvolume and ϵ_j as the dielectric function of subvolume j. These polarizabilities are excact for an infinite cubic lattice in the dc limit $k_0 d \to 0$, but otherwise they do not satisfy the energy conservation [20]. Thus, the radiative reaction (RR) correction of order $O(k_0 d)^3$ was added

$$\alpha_j^{\text{RR}} = \frac{\alpha_j^{\text{CM}}}{1 + (\alpha_j^{\text{CM}}/d^3)[-(2/3)\text{i}(k_0 d)^3]}. \tag{2.4.14}$$

A few attempts were made at adding a correction term of order $O(k_0 d)^2$ [7], but either the assumption of the electric field being uniform over the cubical subvolumes lead to errors of order $O(k_0 d)^2$ itself or the cubical subvolumes were treated as finite spheres of diameter d with modified dielectric functions. Instead, B. T. Draine and J. J. Goodman [7] found the polarizabilities, $\alpha(\omega)$, for which an infinite lattice of polarisable point dipoles have the same lattice dispersion relation (LDR) as a continuum with refractive index $n(\omega) = \sqrt{\varepsilon(\omega)}$ for non-magnetic materials

$$\alpha_j^{\text{LDR}} \approx \frac{\alpha^{\text{CM}}}{1 + (\alpha^{\text{CM}}/d^3)[(b_1^{\text{LDR}} + \varepsilon b_2^{\text{LDR}} + \varepsilon b_3^{\text{LDR}} S)(k_0 d)^2 - (2/3)\text{i}(k_0 d)^3]},$$
$$b_1^{\text{LDR}} = -1.891531, \quad b_2^{\text{LDR}} = 0.1648469, \quad b_3^{\text{LDR}} = -1.7700004, \quad S \equiv \sum_{j=1}^{3} (\hat{k}_{0,j}\hat{e}_j)^2, \tag{2.4.15}$$

in the long-wavelength limit $k_0 d \ll 1$. The unit vectors $\hat{\mathbf{k}}_0$ and $\hat{\mathbf{e}}$ are the direction of the incident plane wave and the polarisation state, respectively. However, this polarizability does not satisfy the transversality condition [20]. Consequently, B. T. Draine and D. Gutkowicz-Krusin [21] proposed a corrected LDR (CLDR) such that the polarizability tensor can only be diagonal instead of isotropic and independent of the incident polarisation

$$\alpha_{\mu\nu}^{\text{CLDR}} \approx \frac{\alpha^{\text{CM}}\delta_{\mu\nu}}{1 + (\alpha^{\text{CM}}/d^3)[(b_1^{\text{LDR}} + \varepsilon b_2^{\text{LDR}} + \varepsilon b_3^{\text{LDR}} a_\mu^2)(k_0 d)^2 - (2/3)\text{i}(k_0 d)^3]}. \tag{2.4.16}$$

The indices μ and ν corresponds to the component of the polarisation and electric field, respectively, and $\delta_{\mu\nu}$ is the Kronecker delta. The lattice dispersion relation polarizability, α^{LDR} is not correct for the dipoles near the surface of the scattering object. B. T. Draine together with M. J. Collinge [20] empirically combined the CLDR polarizability with a method from A. Rahmani, P. C. Chaumet and G. W. Bryant (RCB), in order to deal with the mistreated surface dipoles. However, the surface corrected LDR (SCLDR) method is limited to very specific shapes of the scattering object.

2.4.3 Cross sections and efficiency factors

When the polarizations are determined, the extinction and absorption cross sections can be evaluated [7]

$$C_{\text{ext}} = \frac{4\pi k_0}{E_0^2} \sum_{j=1}^{N} \text{Im}\left(\mathbf{E}_0(\mathbf{R}_j)^* \cdot \mathbf{p}_j\right), \tag{2.4.17}$$

$$C_{\text{abs}} = \frac{4\pi k_0}{E_0^2} \sum_{j=1}^{N} \left[\text{Im}\left(\mathbf{p}_j \cdot (\alpha_j^{-1})^* \mathbf{p}_j^*\right) - \frac{2}{3} k_0^3 p_j^2\right], \tag{2.4.18}$$

with the asterisk * denoting the complex conjugate as for the spherical harmonics. The scattering cross section can be found as the difference between the other two cross sections

$$C_{\text{sca}} = C_{\text{ext}} - C_{\text{abs}}. \tag{2.4.19}$$

The cross sections are probabilities for physical processes to occur when an incident photon intersects with a particle. A cross section has the same units as area, as the probability can be thought of as an effective area for the incident photon to intersect. The efficiency factor, Q, is the cross section divided by the projected area, A, of the scattering object in the direction of the propagation of the incident photon

$$Q = \frac{C}{A}, \tag{2.4.20}$$

which is easier to interpret. The projected area of the scattering object in the direction of the propagation of the incident photon can be difficult to compute for a given set of point dipoles. As an approximation, an effective radius is defined as [7]

$$a_{\text{eff}} \equiv \left(\frac{3V}{4\pi}\right)^{\frac{1}{3}}, \tag{2.4.21}$$

where the total volume, V, of the scattering object is the sum of all the N cubic subvolumes

$$V = Nd^3. \tag{2.4.22}$$

The efficiency factors then become

$$Q = \frac{C}{\pi a_{\text{eff}}^2}, \tag{2.4.23}$$

which works great if the shape of the scattering object happens to be a sphere.

2.4.4 Reflectance from Stokes vectors and the Mueller matrix

For a metasurface consisting of a target unit cell (TUC) repeated periodically in two dimensions, the reflectance as in Sec. 2.2.13 can be defined as the ratio of the reflected beam of photons' intensity, I_s, to the incident one, I_0 [16]

$$R \equiv \frac{I_s}{I_0}. \tag{2.4.24}$$

To find the relation between the incident and scattered intensities, the Stokes vector and the Mueller matrix can be used [12]

$$\begin{pmatrix} I_s \\ Q_s \\ U_s \\ V_s \end{pmatrix} = \frac{1}{n_+^2 k_0^2 r^2} \begin{pmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{pmatrix} \begin{pmatrix} I_0 \\ Q_0 \\ U_0 \\ V_0 \end{pmatrix}, \tag{2.4.25}$$

where k_0 is the length of the wave vector from Eq. (2.4.3), n_+ is the refractive index of the ambient medium, and r is the distance away from the origin of the scattering. The first element of the Stokes vector, I, is the total intensity of the beam of photons. The second Stokes parameter, Q, describes the degree of linear polarisation with $\mathbf{E}$ perpendicular or parallel to the the scattering plane. The scattering plane is spanned by the wave vector for the incident and the scattered beams of photons, $\mathbf{k}_0$ and $\mathbf{k}_s$, respectively. Similarly, the Stokes parameter, U, describes the degree of linear polarisation with $\mathbf{E}$ perpendicular or parallel to the plane obtained from rotating the scattering plane 45° around the wave vector of the incident beam. Lastly, the Stokes parameter V describes the degree of circular polarisation. The components of the electric field parallel and orthogonal to the scattering plane are $E_\parallel = \mathbf{E} \cdot \hat{\mathbf{e}}_\parallel$ and $E_\perp = \mathbf{E} \cdot \hat{\mathbf{e}}_\perp$, respectively. The formalism for the electric field is opposite to the formalism for the wave vector, as the wave vector components are denoted with respect to the substrate in Sec. 2.2. The incident and scattered polarisation vectors orthogonal and parallel to the scattering plane are [8]

$$\begin{aligned} \hat{\mathbf{e}}_{0,\perp} = \hat{\mathbf{e}}_{s,\perp} &\equiv \frac{\hat{\mathbf{k}}_s \times \hat{\mathbf{k}}_0}{|\hat{\mathbf{k}}_s \times \hat{\mathbf{k}}_0|} = \frac{\hat{\mathbf{k}}_s \times \hat{\mathbf{k}}_0}{1 - (\hat{\mathbf{k}}_s \cdot \hat{\mathbf{k}}_0)^2} = -\hat{\phi}_s, \\ \hat{\mathbf{e}}_{0,\parallel} &\equiv \hat{\mathbf{k}}_0 \times \hat{\mathbf{e}}_{0,\perp} = \frac{\hat{\mathbf{k}}_s - (\hat{\mathbf{k}}_s \cdot \hat{\mathbf{k}}_0)\hat{\mathbf{k}}_0}{1 - (\hat{\mathbf{k}}_s \cdot \hat{\mathbf{k}}_0)^2}, \\ \hat{\mathbf{e}}_{s,\parallel} &\equiv \hat{\mathbf{k}}_s \times \hat{\mathbf{e}}_{s,\perp} = \frac{-\hat{\mathbf{k}}_0 + (\hat{\mathbf{k}}_s \cdot \hat{\mathbf{k}}_0)\hat{\mathbf{k}}_s}{1 - (\hat{\mathbf{k}}_s \cdot \hat{\mathbf{k}}_0)^2} = \hat{\theta}_s, \end{aligned} \tag{2.4.26}$$

with $\hat{\phi}_s$ and $\hat{\theta}_s$ as the unit vectors of the azimuthal and polar coordinates of the scattered beam, respectively. The Stokes parameters can then be written as [12]

$$\begin{aligned} I &= \langle E_\parallel E_\parallel^* + E_\perp E_\perp^* \rangle, \\ Q &= \langle E_\parallel E_\parallel^* - E_\perp E_\perp^* \rangle, \\ U &= \langle E_\parallel E_\perp^* + E_\perp E_\parallel^* \rangle, \\ V &= \langle E_\parallel E_\perp^* - E_\perp E_\parallel^* \rangle, \end{aligned} \tag{2.4.27}$$

where the angular brackets denote the time average over one period. The intensity of the scattered beam hence becomes

$$I_s = \frac{1}{n_+^2 k_0^2 r^2} \left(S_{11} I_0 + S_{12} Q_0 + S_{13} U_0 + S_{14} V_0 \right). \tag{2.4.28}$$

As in Sec. 2.2.13, light with the electric field $\mathbf{E}$ perpendicular to the scattering plane, and thus parallel to the metasurface, is defined as s-polarised light or transverse electric waves. Similarly, light with the electric field parallel to the scattering plane is defined as p-polarised light or transverse magnetic waves. The Stokes vector representations of s- and p-polarised light are

$$\begin{aligned} \mathbf{S}^{(s)} &= (1, 1, 0, 0) I, \\ \mathbf{S}^{(p)} &= (1, -1, 0, 0) I, \end{aligned} \tag{2.4.29}$$

with I as the total intensity of the beam. Inserting the values for the s- and p-polarised light into the scattered intensity from Eq. (2.4.28) results in the reflectances

$$\begin{aligned} R^{(s)} &= \frac{(S_{11}+S_{12})I_0}{n_+^2 k^2 r^2 I_0} = \frac{S_{11}+S_{12}}{n_+^2 k^2 r_0^2}, \\ R^{(p)} &= \frac{(S_{11}-S_{12})I_0}{n_+^2 k^2 r^2 I_0} = \frac{S_{11}-S_{12}}{n_+^2 k^2 r_0^2}. \end{aligned} \tag{2.4.30}$$

The Mueller matrix elements are related to the scattering amplitudes S_1, S_2, S_3 and S_4 by [12]

$$\begin{aligned} S_{11} &= \frac{1}{2}\left(|S_1|^2+|S_2|^2+|S_3|^2+|S_4|^2\right), & S_{12} &= \frac{1}{2}\left(|S_2|^2-|S_1|^2+|S_4|^2-|S_3|^2\right), \\ S_{13} &= \mathrm{Re}(S_2S_3^*+S_1S_4^*), & S_{14} &= \mathrm{Im}(S_2S_3^*-S_1S_4^*), \\ S_{21} &= \frac{1}{2}\left(|S_2|^2-|S_1|^2-|S_4|^2+|S_3|^2\right), & S_{22} &= \frac{1}{2}\left(|S_2|^2+|S_1|^2-|S_4|^2-|S_3|^2\right), \\ S_{23} &= \mathrm{Re}(S_2S_3^*-S_1S_4^*), & S_{24} &= \mathrm{Im}(S_2S_3^*+S_1S_4^*), \\ S_{31} &= \mathrm{Re}(S_2S_4^*+S_1S_3^*), & S_{32} &= \mathrm{Re}(S_2S_4^*-S_1S_3^*), \\ S_{33} &= \mathrm{Re}(S_1S_2^*+S_3S_4^*), & S_{34} &= \mathrm{Im}(S_2S_1^*+S_4S_3^*), \\ S_{41} &= \mathrm{Im}(S_4S_2^*+S_1S_3^*), & S_{42} &= \mathrm{Im}(S_4S_2^*-S_1S_3^*), \\ S_{43} &= \mathrm{Im}(S_1S_2^*-S_3S_4^*), & S_{44} &= \mathrm{Re}(S_1S_2^*-S_3S_4^*), \end{aligned} \tag{2.4.31}$$

where for targets periodic in two dimensions, the 2×2 scattering amplitude matrix occurs as [8]

$$\begin{pmatrix} \mathbf{E}_s\cdot\hat{\mathbf{e}}_{s,\parallel} \\ \mathbf{E}_s\cdot\hat{\mathbf{e}}_{s,\perp} \end{pmatrix} = \mathrm{i}\exp(\mathrm{i}\mathbf{k}_s\cdot\mathbf{r}-\mathrm{i}\omega t)\begin{pmatrix} S_2 & S_3 \\ S_4 & S_1 \end{pmatrix}\begin{pmatrix} \mathbf{E}_i\cdot\hat{\mathbf{e}}_{0,\parallel} \\ \mathbf{E}_i\cdot\hat{\mathbf{e}}_{0,\perp} \end{pmatrix}. \tag{2.4.32}$$

The periodic target in two dimensions constitutes a diffraction grating which limits scattering directions, $\mathbf{k}_s$ in Eq. (2.4.32), to [8]

$$\begin{aligned} \mathbf{k}_s &= \pm\mathbf{k}_{0,\perp}+\mathbf{k}_{0,\parallel}+M\mathbf{u}+N\mathbf{v}, \\ k_{s,\perp} &= \left[k_0^2-|\mathbf{k}_{0,\parallel}+M\mathbf{u}+N\mathbf{v}|^2\right]^{\frac{1}{2}}, \end{aligned} \tag{2.4.33}$$

also referred to as diffraction orders (M,N). The $+$ sign gives the scattering directions for transmission and the - sign for reflection. The wave vectors $\mathbf{k}_\parallel$ and $\mathbf{k}_\perp$ denote the wave vectors parallel and perpendicular to the surface, respectively. The notation refers to the surface instead of the scattering plane in order to be consistent with Eq. (2.2.65). The reciprocal lattice vectors are defined as

$$\begin{aligned} \mathbf{u} &\equiv \frac{2\pi\hat{\mathbf{x}}\times\mathbf{L}_v}{\hat{\mathbf{x}}\cdot(\mathbf{L}_u\times\mathbf{L}_v)}, \\ \mathbf{v} &\equiv \frac{2\pi\hat{\mathbf{x}}\times\mathbf{L}_u}{\hat{\mathbf{x}}\cdot(\mathbf{L}_v\times\mathbf{L}_u)}, \end{aligned} \tag{2.4.34}$$

with $\hat{\mathbf{x}}$ as the unit vector for the surface normal. The energy must also be conserved in the scattering process resulting in the requirement $k_{s,\perp}^2>0$. For short lattice constants L_u and L_v compared to the wave length of the incident photon, such as it tends to be in the quasi-static limit, only the scattering order $(M,N)=(0,0)$ is allowed. Then, the reflectivity will be equal to the reflectance. For the special case of forward scattering, when the scattered wave vector is equal to the incident wave

vector $\mathbf{k}_s = \mathbf{k}_i$ and the scattering order is $(M, N) = (0, 0)$, the scattering plane can't be spanned by the two vectors anymore. Using the normal vector to the surface instead of the scattered wave vector, the polarisation vectors can then be defined as

$$\begin{aligned}\hat{\mathbf{e}}_{0,\perp} = \hat{\mathbf{e}}_{s,\perp} &\equiv \frac{\hat{\mathbf{k}}_0 \times \hat{\mathbf{k}}_{s,\parallel}}{|\hat{\mathbf{k}}_0 \times \hat{\mathbf{k}}_{s,\parallel}|}, \\ \hat{\mathbf{e}}_{0,\parallel} \equiv \hat{\mathbf{k}}_0 \times \hat{\mathbf{e}}_{0,\perp}, &\quad \hat{\mathbf{e}}_{s,\parallel} \equiv \hat{\mathbf{k}}_s \times \hat{\mathbf{e}}_{s,\perp}.\end{aligned} \tag{2.4.35}$$

For the forward scattering in the limit where $r \to \infty$, the scattering amplitude matrix can be written as [8]

$$\begin{pmatrix} \mathbf{E}_s \cdot \hat{\mathbf{e}}_{s,\parallel} \\ \mathbf{E}_s \cdot \hat{\mathbf{e}}_{s,\perp} \end{pmatrix} = \mathrm{i} \exp(\mathrm{i}\mathbf{k}_i \cdot \mathbf{r} - \mathrm{i}\omega t) \begin{pmatrix} S_2 - \mathrm{i} & 0 \\ 0 & S_1 - \mathrm{i} \end{pmatrix} \begin{pmatrix} \mathbf{E}_i \cdot \hat{\mathbf{e}}_{0,\parallel} \\ \mathbf{E}_i \cdot \hat{\mathbf{e}}_{0,\perp} \end{pmatrix}. \tag{2.4.36}$$

The scattering amplitude elements can for an infinite two dimensional periodic metasurface be written as [8]

$$\begin{aligned} S_1 &= \frac{2\pi}{k_0^2 A_{\mathrm{TUC}} \sin\alpha_s} \hat{\mathbf{e}}_{s,\perp} \cdot \mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s, \mathbf{E}_0 = \hat{\mathbf{e}}_{0,\perp}), \\ S_2 &= \frac{2\pi}{k_0^2 A_{\mathrm{TUC}} \sin\alpha_s} \hat{\mathbf{e}}_{s,\parallel} \cdot \mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s, \mathbf{E}_0 = \hat{\mathbf{e}}_{0,\parallel}), \\ S_3 &= \frac{2\pi}{k_0^2 A_{\mathrm{TUC}} \sin\alpha_s} \hat{\mathbf{e}}_{s,\parallel} \cdot \mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s, \mathbf{E}_0 = \hat{\mathbf{e}}_{0,\perp}), \\ S_4 &= \frac{2\pi}{k_0^2 A_{\mathrm{TUC}} \sin\alpha_s} \hat{\mathbf{e}}_{s,\perp} \cdot \mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s, \mathbf{E}_0 = \hat{\mathbf{e}}_{0,\parallel}), \end{aligned} \tag{2.4.37}$$

with $A_{\mathrm{TUC}} = |\mathbf{L}_u \times \mathbf{L}_v|$ as the area of the target unit cell and

$$\sin\alpha_0 \equiv \frac{|k_{0,\perp}|}{k_0}, \qquad \sin\alpha_s \equiv \frac{|k_{s,\perp}|}{k_0}, \tag{2.4.38}$$

which in the far field for an infinite two dimensional periodic metasurface can replace the fraction in Eq. (2.4.25)

$$\frac{1}{n_+^2 k_0^2 r^2} = \frac{\sin\alpha_s}{\sin\alpha_0} = \frac{|k_{s,\perp}|}{|k_{0,\perp}|}. \tag{2.4.39}$$

Furthermore, the vector $\mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s)$ appears in the equation for the scattered electric field from the point dipoles [8]

$$\mathbf{E}_s(\mathbf{r}) = \frac{\exp(\mathrm{i}\mathbf{k}_s \cdot \mathbf{r} - \mathrm{i}\omega t)}{k_0 r} \mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s) D(r, \mathbf{k}_s), \tag{2.4.40}$$

with

$$\begin{aligned} D(r, \mathbf{k}_s) \equiv \sum_{m,n} \exp(\mathrm{i}\Phi_{mn}), \qquad \Phi_{mn} \equiv m(\mathbf{k}_0 - \mathbf{k}_s) \cdot \mathbf{L}_u + n(\mathbf{k}_0 - \mathbf{k}_s) \cdot \mathbf{L}_v \\ + \frac{1}{2k_0 r} \left[m^2(k_0^2 - k_{u,s}^2) L_u^2 + n^2(k_0^2 - k_{v,s}^2) L_v^2 + 2mn(k_0^2 \mathbf{L}_u \cdot \mathbf{L}_v - k_{u,s} k_{v,s} L_u L_v) \right], \end{aligned} \tag{2.4.41}$$

where $k_{u,s}$ and $k_{v,s}$ are the components of the scattered wave vector along the lattice vectors $\mathbf{L}_u$ and $\mathbf{L}_v$, respectively. For an infinite periodic metasurface in two dimensions,

$$D(r, \mathbf{k}_s) = \frac{2\pi \mathrm{i} r}{k_0 A_{\mathrm{TUC}} \sin \alpha_s}, \tag{2.4.42}$$

which inserted into Eq. (2.4.40) yields

$$\mathbf{E}_s(\mathbf{r}) = \frac{2\pi \mathrm{i} \exp(\mathrm{i}\mathbf{k}_s \cdot \mathbf{r} - \mathrm{i}\omega t)}{k_0^2 A_{\mathrm{TUC}} \sin \alpha_s} \mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s). \tag{2.4.43}$$

Lastly, the vector $\mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s)$ is defined as

$$\mathbf{F}_{\mathrm{TUC}}(\hat{\mathbf{k}}_s) \equiv k_0^3 [1 - \hat{\mathbf{k}}_s \hat{\mathbf{k}}_s] \sum_{j=1}^{N} \mathbf{p}_j^{(0,0)} \exp\Big(\mathrm{i}\omega t - \mathrm{i}\mathbf{k}_s \cdot \mathbf{R}_j^{(0,0)}\Big). \tag{2.4.44}$$

2.4.5 Mie cross sections

The scattering and extinction cross sections for spheres can from Mie theory [12] be expressed analytically by the scattering coefficients a_n and b_n

$$\begin{aligned} C_{\mathrm{sca}} &= \frac{2\pi}{k_0^2} \sum_{n=1}^{\infty} (2n+1)(|a_n|^2 + |b_n|^2), \\ C_{\mathrm{ext}} &= \frac{2\pi}{k_0^2} \sum_{n=1}^{\infty} (2n+1) \operatorname{Re}(a_n + b_n^2), \end{aligned} \tag{2.4.45}$$

and the absorption cross section can be found as the difference

$$C_{\mathrm{abs}} = C_{\mathrm{ext}} - C_{\mathrm{sca}}. \tag{2.4.46}$$

The scattering coefficients are

$$\begin{aligned} a_n &= \frac{\mu_+ m^2 j_n(mx)[x j_n(x)]' - \mu j_n(x)[mx j_n(mx)]'}{\mu_+ m^2 j_n(mx)[x h_n^{(1)}(x)]' - \mu h_n^{(1)}(x)[mx j_n(mx)]'}, \\ b_n &= \frac{\mu j_n(mx)[x j_n(x)]' - \mu_+ j_n(x)[mx j_n(mx)]'}{\mu j_n(mx)[x h_n^{(1)}(x)]' - \mu_+ h_n^{(1)}(x)[mx j_n(mx)]'}, \end{aligned} \tag{2.4.47}$$

where μ and μ_+ are the permeabilities of the sphere and the ambient medium, respectively. The relative refractive index is $m = n/n_+$ and the size parameter $x = k_0 n_+ a = 2\pi n_+ a/\lambda$, with a as the radius of the sphere and λ as the vacuum wave length of the incident photon. The prime indicates differentiation with respect to the argument inside the parentheses. The spherical Bessel functions of first and second kinds are

$$\begin{aligned} j_n(x) &= x^n \left(-\frac{1}{x}\frac{\mathrm{d}}{\mathrm{d}x}\right)^n \frac{\sin x}{x}, \\ y_n(x) &= -x^n \left(-\frac{1}{x}\frac{\mathrm{d}}{\mathrm{d}x}\right)^n \frac{\cos x}{x}, \end{aligned} \tag{2.4.48}$$

and the spherical Hankel function of first kind is

$$h_n^{(1)}(x) = j_n(x) + \mathrm{i} y_n(x). \tag{2.4.49}$$

2.4.6 Reflectance and transmittance for a thin film

The reflectance and transmittance for a thin film of thickness d and refractive index n, surrounded by an ambient medium of refractive index n_+, can be written as [22]

$$R = \left| \frac{r_2 + r_1 \exp(2\mathrm{i}knd\cos\theta_T)}{1 + r_1 r_2 \exp(2\mathrm{i}knd\cos\theta_T)} \right|^2,$$
$$T = \left| \frac{t_1 t_2 \exp(\mathrm{i}knd\cos\theta_T)}{1 + r_1 r_2 \exp(2\mathrm{i}knd\cos\theta_T)} \right|^2, \tag{2.4.50}$$

where the incident photon has a wave number k and an incident angle θ. The transmitted angle, θ_T can be found from Snell's law, $n_+ \sin\theta = n \sin\theta_T$. For p-polarised light, the reflection and transmission coefficients are

$$r_1^{(p)} = \frac{n\cos\theta - n_+\cos\theta_T}{n\cos\theta + n_+\cos\theta_T}, \qquad r_2^{(p)} = \frac{n_+\cos\theta_T - n\cos\theta}{n_+\cos\theta_T + n\cos\theta},$$
$$t_1^{(p)} = \frac{2n\cos\theta_T}{n\cos\theta + n_+\cos\theta_T}, \qquad t_2^{(p)} = \frac{2n_+\cos\theta}{n_+\cos\theta_T + n\cos\theta}, \tag{2.4.51}$$

and similarly for s-polarised light

$$r_1^{(s)} = \frac{n\cos\theta_T - n_+\cos\theta}{n\cos\theta_T + n_+\cos\theta}, \qquad r_2^{(s)} = \frac{n_+\cos\theta - n\cos\theta_T}{n_+\cos\theta + n\cos\theta_T},$$
$$t_1^{(s)} = \frac{2n\cos\theta_T}{n\cos\theta_T + n_+\cos\theta}, \qquad t_2^{(s)} = \frac{2n_+\cos\theta}{n\cos\theta_T + n_+\cos\theta}. \tag{2.4.52}$$

Chapter 3

Method

3.1 Truncating the system of equations

3.1.1 Truncating the system of equations for a finite set of spheres

In Eq. (2.2.52) there are two sums over l_i. Since infinite sums can not be calculated numerically, the sums are truncated at some finite value of l_i. The truncation in the first sum is denoted as $L_\perp$ and the latter as $L_\parallel$, since they represent the interactions perpendicular and parallel to the surface of the substrate. Setting $L = \max(L_\parallel, L_\perp)$ as the largest of those truncations results in $M \equiv (L+1)^2 - 1$ unknown coefficients $A_l^{(j)}$, as the coefficients appear in both summations in Eq. (2.2.53). The -1 contribution originates from $A_{0,0}^{(j)}$ being zero. The truncated version of Eq. (2.2.53) then yields

$$b_k = \tilde{C}_k^{(j)} A_k^{(j)} + \sum_{l=0}^{(L_\perp+1)^2-2} \bar{C}_{k,l}^{(j)} A_l^{(j)} + \sum_{i\neq j} \sum_{l=0}^{(L_\parallel+1)^2-2} \hat{C}_{k,l}^{(i,j)} A_l^{(i)}. \tag{3.1.1}$$

The relative azimuthal angle of an image multipole, $\phi_{\bar{i}j}$, is equal to the relative azimuth angle of its sphere ϕ_{ij}. Furthermore, the polar angle of an image multipole relative to its own sphere, $\theta_{\bar{j}j}$, is always equal to 180° as the image multipole is located directly below its sphere. The spherical harmonics of $\theta = 180°$ is only nonzero for $m = 0°$, and thus independent of ϕ. A Kronecker's delta can be used to avoid computing $\bar{C}_{k,l}^{(j)}$ when $m_i \neq m_j$. Next, if a sum reaches its truncation before the other, its C coefficients are simply set to zero by the Heaviside function, $u(l)$. The revised C coefficients can then be written on the form

$$\tilde{C}_k^{(j)} = a_j^{-2l_j(k)-1} \left[\frac{l_j(k)\varepsilon_j + \varepsilon_+(l_j(k)+1)}{l_j(k)(\varepsilon_j - \varepsilon_+)} \right]$$

$$\bar{C}_{k,l}^{(j)} = \delta_{m_i m_j} (-1)^{l_i(l)+m_i(l)} \beta H\left(l_j(k), m_j(k) \middle| l_i(l), m_i(l)\right) \frac{Y^0_{l_i(l)+l_j(k)}(\pi, \phi_{jj})}{R_{\bar{j}j}^{l_j(k)+l_i(l)+1}} u(L_\perp - l_j(k))$$

$$\hat{C}_{k,l}^{(i,j)} = H\left(l_j(k), m_j(k) \middle| l_i(l), m_i(l)\right) \left[\frac{Y^{m_i(l)-m_j(k)}_{l_i(l)+l_j(k)}(\theta_{ij}, \phi_{ij})}{R_{ij}^{l_j(k)+l_i(l)+1}} \right.$$

$$\left. + (-1)^{l_i(l)+m_i(l)} \beta \frac{Y^{m_i(l)-m_j(k)}_{l_i(l)+l_j(k)}(\theta_{\bar{i}j}, \phi_{ij})}{R_{\bar{i}j}^{l_j(k)+l_i(l)+1}} \right] u(L_\parallel - l_i(l)). \tag{3.1.2}$$

When solving equation (2.2.52) at the surface of all N spheres for all $l_j \leq L$, the system of equations becomes closed and finite dimensional

$$\mathbf{b} = \mathbf{C}\mathbf{A}, \tag{3.1.3}$$

where

$$\mathbf{b} = \begin{pmatrix} -b_{1,-1} \\ -b_{1,0} \\ -b_{1,1} \\ 0 \\ \vdots \\ 0 \\ -b_{1,-1} \\ -b_{1,0} \\ -b_{1,1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} A^{(1)}_{1,-1} \\ A^{(1)}_{1,0} \\ A^{(1)}_{1,1} \\ A^{(1)}_{2,-2} \\ \vdots \\ A^{(1)}_{L,L} \\ A^{(2)}_{1,-1} \\ A^{(2)}_{1,0} \\ A^{(2)}_{1,1} \\ A^{(2)}_{2,-2} \\ \vdots \\ A^{(N)}_{L,L} \end{pmatrix} \quad \text{and} \tag{3.1.4}$$

$$\mathbf{C} = \begin{pmatrix} \bar{C}^{(1)}_{0,0} + \tilde{C}^{(1)}_{0} & \bar{C}^{(1)}_{0,1} & \dots & \bar{C}^{(1)}_{0,M-1} & \hat{C}^{(2,1)}_{0,0} & \dots & \hat{C}^{(N,1)}_{0,M-1} \\ \bar{C}^{(1)}_{1,0} & \bar{C}^{(1)}_{1,1} + \tilde{C}^{(1)}_{1} & \dots & \bar{C}^{(1)}_{1,M-1} & \hat{C}^{(1,0)}_{2,1} & \dots & \hat{C}^{(N,1)}_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \bar{C}^{(1)}_{M-1,0} & \bar{C}^{(1)}_{M-1,1} & \dots & \bar{C}^{(1)}_{M-1,M-1} + \tilde{C}^{(1)}_{M-1} & \hat{C}^{(2,1)}_{M-1,0} & \dots & \hat{C}^{(N,1)}_{M-1,M-1} \\ \hat{C}^{(1,2)}_{0,0} & \hat{C}^{(1,2)}_{0,1} & \dots & \hat{C}^{(1,2)}_{0,M-1} & \bar{C}^{(2)}_{0,0} + \tilde{C}^{(2)}_{0} & \dots & \hat{C}^{(N,2)}_{0,M-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hat{C}^{(1,N)}_{M-1,0} & \hat{C}^{(1,N)}_{M-1,1} & \dots & \hat{C}^{(1,N)}_{M-1,M-1} & \hat{C}^{(2,N)}_{M-1,0} & \dots & \bar{C}^{(N)}_{M-1,M-1} + \tilde{C}^{(N)}_{M-1} \end{pmatrix}. \tag{3.1.5}$$

There are a total of $NM = N(L+1)^2 - N$ unknown coefficients $A^{(i)}_k$ to solve for. The dimensions of the vectors are thus NM, and the vectors are here indexed by k. The matrix $\mathbf{C}$ has dimensions $NM \times NM$, and its rows are here indexed by k and columns by l. Using integer division, div, and its remainder, mod, which satisfy

$$x = y(x \operatorname{div} y) + (x \bmod y), \tag{3.1.6}$$

j, i, and their corresponding k' and l' can be found from the indices k and l

$$j(k) = 1 + k \operatorname{div} M, \qquad i(l) = 1 + l \operatorname{div} M, \tag{3.1.7a}$$

$$k' = k \bmod M, \qquad l' = l \bmod M. \tag{3.1.7b}$$

The elements of $\mathbf{C}$ hence become

$$C_{k,l} = \tilde{C}^{(j(k))}_{k'} \delta_{kl} + \begin{cases} \bar{C}^{(j(k))}_{k',l'}, & j(k) = i(l), \\ \hat{C}^{(i(l),j(k))}_{k',l'}, & j(k) \neq i(l), \end{cases} \tag{3.1.8}$$

and the elements of $\mathbf{A}$

$$A_k = A^{(j(k))}_{l_j(k'),m_j(k')}, \tag{3.1.9}$$

and $\mathbf{b}$

$$b_k = -b_{1,m_j(k')}\delta_{1,l_j(k')}. \tag{3.1.10}$$

The vector $\mathbf{A}$ can then be found from solving Eq. (3.1.3) with values from Eqs. (3.1.8)–(3.1.10).

3.1.2 Truncating the system of equations for an infinite lattice

For a lattice based on a unit cell repeated periodically infinitely many times as described in Sec. 2.2.9, the same truncations made in Sec. 3.1.1 still apply for the matrix accounting for the contributions from the unit cell at lattice point $(0,0)$, $\mathbf{C}^{(0,0)}$, in Eq. (2.2.83). For the matrix accounting for the rest of the unit cells, $\check{\mathbf{C}}$, the contributions will be truncated by the same truncation as the contributions from the neighbouring spheres in $\mathbf{C}^{(0,0)}$, $L_\parallel$

$$b_k = \tilde{C}_k^{(j)}A_k^{(j)} + \sum_{l=0}^{(L_\perp+1)^2-2}\bar{C}_{k,l}^{(j)}A_l^{(j)} + \sum_{i\neq j}\sum_{l=0}^{(L_\parallel+1)^2-2}\hat{C}_{k,l}^{(i,j)}A_l^{(i)} + \sum_{i=1}^{N}\sum_{l=0}^{(L_\parallel+1)^2-2}\check{C}_{k,l}^{(i,j)}A_l^{(i)}. \tag{3.1.11}$$

Furthermore, the summation over lattice points in $\sum_{i',l'}'$ can't include an infinite number of lattice

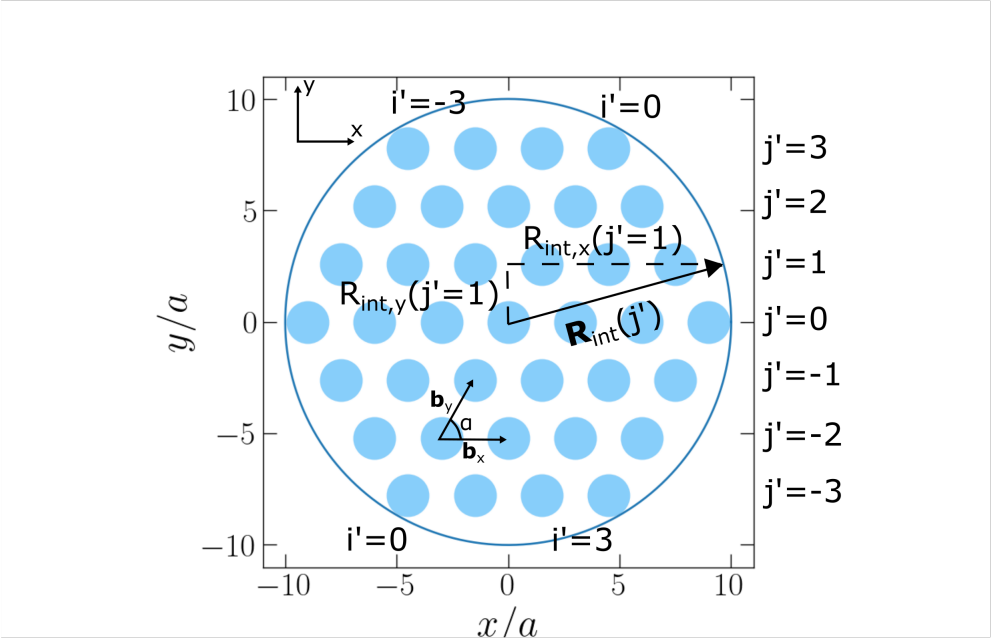


FIGURE 3.1: An example of a limited section of spheres with radius a from an infinite periodic lattice. The section is limited by the distance of interaction, R_{int}, which forms a circle in the xy-plane. The components of the vector $\mathbf{R}_{\text{int}}(j')$ depend on which lattice coordinate j' the vector ends at. The periodicity of the lattice is given by the lattice vectors $\mathbf{b}_x$ and $\mathbf{b}_y$. The distance b_x is the spacing between the spheres sharing the same lattice coordinate j', and similarly for the distance b_y and the lattice coordinate i'. In this example the distance of interaction is set to, $10a$, and the lattice spacings are $b_x = b_y = 3a$. The lattice vector $\mathbf{b}_x$ is parallel to the x-axis, and $\mathbf{b}_y$ is oriented an angle $\alpha = 60°$ up from the x-axis.

points numerically. Instead, only the unit cells with a distance $\Delta R^{(i',j')}$ smaller than a given interaction distance R_{int} are included as illustrated by Fig. 3.1. To determine the limits for the sums over the lattice coordinates i' and j', the components of the interaction vector $\mathbf{R}_{\text{int}}(j')$, which

has the constant interaction distance R_{int} as length, can be used. The interaction vector is defined to start at lattice site $(0,0)$ and end towards right on the x-axis at lattice coordinate j'. Consequently, the included lattice sites are contained within a circle of radius R_{int} with centre in lattice site $(0,0)$. The two components of the interaction vector are

$$\begin{aligned} R_{\text{int,y}}(j') &= j' b_y \sin\alpha, \\ R_{\text{int,x}}(j') &= \sqrt{R_{\text{int}}^2 - (j' b_y \sin\alpha)^2}, \end{aligned} \tag{3.1.12}$$

where $R_{\text{int,x}}(j')$ is found from the Pythagorean theorem. Due to the translational symmetry of the lattice and the interaction vector starting at lattice point $(0,0)$, the limits for the sum over the lattice coordinate j' are anti-symmetric $j'_{\text{end}} = -j'_{\text{start}} \equiv N_{j'}$. The limit can be found from the largest j' such that $R_{\text{int,y}}(j') \leq R_{\text{int}}$. By using the integer division from Eq. (3.1.6), the limit can be written as

$$N_{j'} = R_{\text{int}} \text{ div } (b_y \sin\alpha). \tag{3.1.13}$$

For the limits for the sum over the lattice coordinate i' on the other hand, the symmetry breaks down for $\alpha \neq 90°$. These limits also depend on the lattice coordinate j' of the row. As for $N_{j'}$, the start limit for the sum over the lattice coordinate i' at row j', $i'_{\text{start}}(j')$, can be found from the smallest i' such that $\Delta R_x^{(i',j')} \geq -R_{\text{int,x}}(j')$. Likewise, the end limit for the sum over the lattice coordinate i' at row j', $i'_{\text{end}}(j')$, can be found from the greatest i' such that $\Delta R_x^{(i',j')} \leq R_{\text{int,x}}(j')$. Applying the identities for $\Delta\mathbf{R}^{(i',j')}$ and $\mathbf{R}_{\text{int}}(j')$ from Eqs. (2.2.61) and (3.1.12), respectively, leads to the inequalities

$$\begin{aligned} i'_{\text{start}}(j') b_x + j' b_y \cos\alpha &\geq -\sqrt{R_{\text{int}}^2 - (j' b_y \sin\alpha)^2}, \\ i'_{\text{end}}(j') b_x + j' b_y \cos\alpha &\leq \sqrt{R_{\text{int}}^2 - (j' b_y \sin\alpha)^2}. \end{aligned} \tag{3.1.14}$$

With the floor function from Eq. (2.2.54), and its complement, the ceiling function $\lceil\ \rceil$ returning the smallest integer greater than or equal to its argument, the limits for the sum over the lattice coordinate i' at row j' become

$$\begin{aligned} i'_{\text{start}}(j') &= \left\lceil \frac{1}{b_x}\left(-\sqrt{R_{\text{int}}^2 - (j' b_y \sin\alpha)^2} - j' b_y \cos\alpha\right)\right\rceil, \\ i'_{\text{end}}(j') &= \left\lfloor \frac{1}{b_x}\left(\sqrt{R_{\text{int}}^2 - (j' b_y \sin\alpha)^2} - j' b_y \cos\alpha\right)\right\rfloor. \end{aligned} \tag{3.1.15}$$

The lattice point $(0,0)$ is still excluded from the summation. The C coefficients then take the form

$$\begin{aligned} \check{C}_{k,l}^{(i,j)} = \sum_{\substack{j'=-N_{j'} \\ (i',j')\neq(0,0)}}^{N_{j'}} \sum_{i'=-i'_{\text{start}}(j')}^{i'_{\text{end}}(j')} & \mathrm{e}^{\mathrm{i}\mathbf{k}_\parallel\cdot\Delta\mathbf{R}^{(i',j')}} H\left(l_j(k), m_j(k) \middle| l_i(l), m_i(l)\right) u(L_\parallel - l_i(l)) \\ \times \left[\frac{Y_{l_i(l)+l_j(k)}^{m_i(l)-m_j(k)}\left(\theta_{ij}^{(i',j')}, \phi_{ij}^{(i',j')}\right)}{\left(R_{ij}^{(i',j')}\right)^{l_j(k)+l_i(l)+1}} + (-1)^{l_i(l)+m_i(l)} \beta \frac{Y_{l_i(l)+l_j(k)}^{m_i(l)-m_j(k)}\left(\theta_{\bar{i}\bar{j}}^{(i',j')}, \phi_{\bar{i}\bar{j}}^{(i',j')}\right)}{\left(R_{\bar{i}\bar{j}}^{(i',j')}\right)^{l_j(k)+l_i(l)+1}} \right]. \end{aligned} \tag{3.1.16}$$

The finite $\check{\mathbf{C}}$ matrix accounting for the contributions from the unit cells within a distance of R_{int} around the unit cell located at lattice point $(0,0)$ written out is

$$\check{\mathbf{C}} \equiv \begin{pmatrix} \check{C}_{0,0}^{(1,1)} & \check{C}_{0,1}^{(1,1)} & \dots & \check{C}_{0,M-1}^{(1,1)} & \check{C}_{0,0}^{(2,1)} & \dots & \check{C}_{0,M-1}^{(N,1)} \\ \check{C}_{1,0}^{(1,1)} & \check{C}_{1,1}^{(1,1)} & \dots & \check{C}_{1,M-1}^{(1,1)} & \check{C}_{1,0}^{(2,1)} & \dots & \check{C}_{1,M-1}^{(N,1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \check{C}_{M-1,0}^{(1,1)} & \check{C}_{M-1,1}^{(1,1)} & \dots & \check{C}_{M-1,M-1}^{(1,1)} & \check{C}_{M-1,0}^{(2,1)} & \dots & \check{C}_{M-1,M-1}^{(N,1)} \\ \check{C}_{0,0}^{(1,2)} & \check{C}_{0,1}^{(1,2)} & \dots & \check{C}_{0,M-1}^{(1,2)} & \check{C}_{0,0}^{(2,2)} & \dots & \check{C}_{0,M-1}^{(N,2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \check{C}_{M-1,0}^{(1,N)} & \check{C}_{M-1,1}^{(1,N)} & \dots & \check{C}_{M-1,M-1}^{(1,N)} & \check{C}_{M-1,0}^{(2,N)} & \dots & \check{C}_{M-1,M-1}^{(N,N)} \end{pmatrix}. \tag{3.1.17}$$

The matrix $\check{\mathbf{C}}$ still has dimensions $NM \times NM$, even if $L_{\parallel} < L$, in order to execute the matrix addition in Eq. (2.2.83). Consequently, the same conventions from Eqs. (3.1.7a) and (3.1.7b) can be used to find the elements of the matrix

$$\check{C}_{k,l} = \check{C}_{k',l'}^{(i(l),j(k))}. \tag{3.1.18}$$

Furthermore, the length of the component of the wave vector parallel to the surface, from Eq. (2.2.66), is assumed to be zero for the solution of the Laplace equation [3]

$$|\mathbf{k}_{\parallel}| = 0. \tag{3.1.19}$$

The phase factor in the Bloch-Floquet theorem from Eq. (2.2.65) is then neglected.

3.2 DDSCAT

3.2.1 Choice of method for the discrete dipole approximation

The choice of the method for the discrete dipole approximation was between the two open source programs DDSCAT written in Fortran by B. T. Draine and P. J. Flatau [7, 8, 9] and ADDA written in C by M. A. Yurkin and A. G. Hoekstra [23]. The advantage of ADDA over DDSCAT is its "surface mode" allowing for scattering by particles located above a plane interface [24]. However, ADDA is limited to finite systems of scatterers whereas DDSCAT allows for infinitely periodic targets [8]. As it makes more sense to compute the reflectance of an infinitely periodic metasurface without a substrate than one of a finite set of scatterers above a substrate, the choice fell on DDSCAT. Furthermore, with the high order of inter-particle interaction calculated in the multipole expansion in Sec. 2.2, the focus on the interaction with a substrate is not as high as for GranFilm. Both DDSCAT and ADDA require the interdipole separation, d, to be small compared to the structural lengths of the targets. Moreover, DDSCAT [25] requires $|n|kd < 1$ and $|n-1| \lesssim 3$, while ADDA [23] requires $|n|kd < \pi/5$ and $|n-1| < 2$, with k as the wave number. The refractive index, n, for Ag provided by the Sopra database [6], in the interval $[2.7\,\text{eV}, 3.7\,\text{eV}]$, has a maximum value $\max(|n-1|) \approx 2.7$, which only satisfies the criteria from DDSCAT. With the largest values for $|n|$ and k, the product $|n|kd$ ranges from approximately 0.1 to 0.01 depending on the interdipole separation, d. Consequently, the other criteria is satisfied for both methods.

3.2.2 Truncating the system of equations

When numerically solving for the polarisations in Eq. (2.4.11), the number of lattice sites to be summed over has to be finite as for the approach for the multipole expansion in Sec. 3.1.2. The truncation is performed by only summing over the point dipoles within the distance $R_{jk}^{(m,n)} \leq 2/(\gamma k_0)$, which is denoted by the prime over the sum

$$\bar{\mathbf{G}}(\mathbf{R}_{jk}) \approx \sum_{m,n}{}' \mathbf{G}(\mathbf{R}_{jk}^{(m,n)}) \exp\Big[\mathrm{i}(m\mathbf{k}_0 \cdot \mathbf{L}_u + n\mathbf{k}_0 \cdot \mathbf{L}_v) - (\gamma k_0 R_{jk}^{(m,n)})^4\Big]. \tag{3.2.1}$$

A suppression factor $\exp\Big[-(\gamma k_0 R_{jk}^{(m,n)})^4\Big]$ is added to make the truncation smoother [8]. In the limit $R_{jk}^{(m,n)} \approx 2/(\gamma k_0)$, the suppression factor becomes $\exp\Big[-(\gamma k_0 R_{jk}^{(m,n)})^4\Big] \approx \mathrm{e}^{-16}$. The parameter γ is chosen to be small enough for the summation over the lattice sites (m, n) to yield representative results, but large enough to prevent too long computation time, as the computation time is inversely proportional to gamma squared $\sim \gamma^{-2}$.

3.2.3 Application of discrete Fourier transform to speed up computation

An iterative algorithm to solve Eq. (2.4.8) and (2.4.11) can be used, such as the conjugate-gradient (CG) algorithm. The algorithm starts with an initial guess for the polarisations, $\mathbf{p}_j$, and determines the next step based on the error, $\mathbf{E}_0(\mathbf{R}_j) + \sum_{k=1}^{N} \mathbf{G}(\mathbf{R}_{jk})\mathbf{p}_k$, until the steps have converged monotonically towards a result with sufficiently small error. B. T. Draine and P. J. Flatau together with J. J. Goodman [26] accelerated the computation of this sum, for point dipoles located on a cubic lattice with spacing d and dimensions N_x, N_y, N_z. Using a three dimensional point dipole indexing

$$\mathbf{i} \equiv (i_x, i_y, i_z), \tag{3.2.2}$$

and $\mathbf{r}_0$ as an arbitrary origin, the relative position vector becomes

$$\mathbf{R_{ji}} = \mathbf{R_j} - \mathbf{R_i} = (j_x d, j_y d, j_z d) + \mathbf{r}_0 - \big((i_x d, i_y d, i_z d) + \mathbf{r}_0\big) = d(j_x - i_x, j_y - i_y, j_z - i_z). \tag{3.2.3}$$

Thus, the interaction matrix $\mathbf{G}(\mathbf{R_{ji}})$ only depends on the difference in the indices. Consequently, the interaction matrix can be written as a 2-level Block-Toeplitz matrix [20]

$$\mathbf{G'_{j-i}} \equiv \begin{cases} \mathbf{G}(\mathbf{R_{ji}}), & \mathbf{j} \neq \mathbf{i} \\ -\alpha_j^{-1}, & \mathbf{j} = \mathbf{i} \end{cases}. \tag{3.2.4}$$

Equation (2.4.8) can then be transformed into a discrete convolution

$$-\mathbf{E}_0(\mathbf{R_j}) = \sum_{i=1}^{N} \mathbf{G}(\mathbf{R}_{ji})\mathbf{p}_i = \sum_{\mathbf{i}=(1,1,1)}^{(N_x,N_y,N_z)} \mathbf{G'_{j-i}}\mathbf{p_i} = \sum_{\mathbf{i}=(1,1,1)}^{(2N_x,2N_y,2N_z)} \mathbf{G'_{j-i}}\mathbf{p'_i}, \tag{3.2.5}$$

where the polarisations are only non-zero in the occupied lattice sites

$$\mathbf{p'_i} \equiv \begin{cases} \mathbf{p_i}, & \text{site } \mathbf{i} \text{ is occupied} \\ 0, & \text{site } \mathbf{i} \text{ is unoccupied} \end{cases}. \tag{3.2.6}$$

Both $\mathbf{G}'_{\mathbf{j}-\mathbf{i}}$ and $\mathbf{p}'_{\mathbf{i}}$ are doubled and periodic in every dimension, with periodicity of twice the dimensions of the lattice

$$\begin{aligned}\mathbf{p}'_{(i_x,i_y,i_z)} &= \mathbf{p}'_{(i_x\pm 2N_x,i_y\pm 2N_y,i_z\pm 2N_z)},\\ \mathbf{G}'_{(j_x-i_x,j_y-i_y,j_z-i_z)} &= \mathbf{G}'_{(j_x-i_x\pm 2N_x,j_y-i_y\pm 2N_y,j_z-i_z\pm 2N_z)}.\end{aligned} \tag{3.2.7}$$

The polarisations $\mathbf{p}'_{\mathbf{i}}$ are thus zero for all the lattice site coordinates $\mathbf{i}$ with components $N_\mu < i_\mu \leq 2N_\mu$. The Fourier transform of a convolution of two functions is the element-wise product of the Fourier transforms of the functions, denoted by $\circ$

$$-\hat{\mathbf{E}}_0(\mathbf{R_n}) = \hat{\mathbf{G}}'_{\mathbf{n}} \circ \hat{\mathbf{p}}_{\mathbf{n}}, \tag{3.2.8}$$

with $\hat{\mathbf{E}}_0$, $\hat{\mathbf{G}}'$ and $\hat{\mathbf{p}}'$ as the discrete Fourier transform of $\mathbf{E}_0$, $\mathbf{G}'$ and $\mathbf{p}'$, respectively

$$\hat{\mathbf{E}}_0(\mathbf{R_n}) \equiv \sum_{\mathbf{i}} \mathbf{E}_0(\mathbf{R_i}) \exp\left[-2\pi \mathrm{i}\left(\frac{n_x i_x}{2N_x} + \frac{n_y i_y}{2N_y} + \frac{n_z i_z}{2N_z}\right)\right]. \tag{3.2.9}$$

The computation of the sum in Eq. (2.4.8) can thus be found from the inverse discrete Fourier transform of the right-hand-side in Eq. (3.2.8)

$$\sum_{j=1}^{N} \mathbf{G}(\mathbf{R}_{\mathbf{i}j})\mathbf{p}_j = \frac{1}{8N_xN_yN_z} \sum_{\mathbf{n}} \hat{\mathbf{G}}'_{\mathbf{n}} \circ \hat{\mathbf{p}}_{\mathbf{n}} \exp\left[2\pi \mathrm{i}\left(\frac{n_x i_x}{2N_x} + \frac{n_y i_y}{2N_y} + \frac{n_z i_z}{2N_z}\right)\right], \tag{3.2.10}$$

which for all $\mathbf{n}$ point dipole positions can be evaluated in $O((N_xN_yN_z)\ln(N_xN_yN_z))$ operations, instead of the $O(N^2)$ it would have taken otherwise. If N_x, N_y and N_z aren't very factorisable, the amount of operations can be as high as $\approx O((N_xN_yN_z)^{\frac{4}{3}})$ [26]. The FFT method can also be used for the summation in Eq. (2.4.12) when calculating the electric near field [9].

3.2.4 The parameter file

Appendix B.3 presents the parameter file, "ddscat.par", where DDSCAT reads its input arguments. Instructions for the parameter file are found in the DDSCAT user guide [27]. The preliminaries in the parameter file include five parameters. The first one, 'NOTORQ', tells DDSCAT to skip radiative torque calculations. Then comes the iterative solution algorithm, where 'PBCGS2' is the BiConjugate Gradient with Stabilisation. Third is the Fast Fourier Transform algorithm for the computation described in Sec. 3.2.3, and 'FFTMKL' uses the Intel® oneAPI Math Kernel Library. Next, the approximation for the polarizabilities used in Eq. (2.4.7) is decided. The option 'LATTDR' is the lattice dispersion relation approximation described in Eq. (2.4.15), and 'GKLDR' is the corrected lattice dispersion relation approximation described in Eq. (2.4.16). Lastly, the preliminary 'NOTBIN' tells DDSCAT to not write the output files in binary.

The memory allocation for target generation is specified by the maximum number of point dipole sites, N_x, N_y, and N_z, in the three dimensions. The periodic target geometry with the target unit cell read from a target file is specified by the option 'FRMFILPBC'. In order to specify the periodic boundary conditions, the polarisation states and the wave vector of the incident light must be specified in the target frame (TF) and the lab frame (LF). The target frame has unit vectors $\hat{\mathbf{x}}_{\mathrm{TF}}$, $\hat{\mathbf{y}}_{\mathrm{TF}}$ and $\hat{\mathbf{z}}_{\mathrm{TF}}$. The four parameters for the 'FRMFILPBC' target is the periodicity in $\hat{\mathbf{y}}_{\mathrm{TF}}$ and $\hat{\mathbf{z}}_{\mathrm{TF}}$, respectively, in the units of the inter-dipole distance, d. The next parameter is a number

from 1 to 6, where only 1 is allowed for periodic target. This number specifies two orthogonal unit vectors $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$, which will be used for orientation of the target frame in the lab frame. The number 1 specifies $\hat{\mathbf{a}}_1 \parallel \hat{\mathbf{x}}_{\mathrm{TF}}$ and $\hat{\mathbf{a}}_2 \parallel \hat{\mathbf{y}}_{\mathrm{TF}}$. The third unit vector $\hat{\mathbf{a}}_3$ is found from the cross product $\hat{\mathbf{a}}_3 = \hat{\mathbf{a}}_1 \times \hat{\mathbf{a}}_2$. The fourth parameter is the file name of the target file.

Appendix B.4 is an example of a snippet of a target file for a sphere generated by the nanoHUB tool DDSCAT Shape Generator [10]. The second line in the target file specifies the number of point dipoles, and the next two lines specify the two unit vectors $\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$ in the target file. The fourth line is the relative spacing of point dipoles in the $\hat{\mathbf{x}}_{\mathrm{TF}}$, $\hat{\mathbf{y}}_{\mathrm{TF}}$ and $\hat{\mathbf{z}}_{\mathrm{TF}}$ directions, and the fifth line describes the target frame coordinates of the centre of the scattering object. Then, all the relative point dipole positions in arbitrary units in the target frame and the corresponding dielectric files for the relative permittivities in the three dimensions are listed. The number of dielectric files is specified below the target parameters, and then the file names for the corresponding dielectric files are listed. Appendix B.5 shows an example of a dielectric file made by the function make_dielectrics_for_DDSCAT in App. B.1, from a dielectric file from the SOPRA database illustrated in App. A.2. The latter file has a header including 4 numbers. The first number is a flag for whether the refractive indices are given linearly spaced for wave energies (1) or wave lengths (2). The next two numbers are the start and end energies or lengths in eV or µm, respectively. At last is the total number of refractive indices minus one. Then, the real and imaginary parts of the refractive indices are listed. The structure in the dielectric file used by DDSCAT is a bit different. The first and third lines are just comments not read as input. The second line is five numbers indicating what is stored in the corresponding five columns after the third line. The number 1 represents the wave lengths. Number 2 and 3 are the real and imaginary parts of the refractive indices, and 4 and 5 are the real and imaginary parts of the relative permittivities. As only either of the refractive indices and the relative permittivities are used, the two numbers not used are set to zero instead.

The near field parameter, NRFLD, can be set to 0 to skip the additional near field calculations. If only the near field calculations for the electric field are desired, then NRFLD is set to 1. Lastly, if the near field calculations for both the electric and magnetic fields are desired, NRFLD is set to 2. The next line consists of six non-negative numbers describing the fractional extensions of the original volume, in which the near field calculations shall take place. The six numbers correspond to both directions in each of the three dimensions. After the computation is done, a binary .E1 file containing the values for the electric field is made. The DDSCAT routine ddpostprocess creates a text file, "ddpostprocess.out", from the binary file. The routine reads from another parameter file called "ddpostprocess.par" as viewed in App. B.6. The first line in the post processing parameter file tells which binary file to read the electric near field values from. The second line is the name of the VTR output file to be made. The third line determines if the VTR output file is to be made, 1 for true and 0 for false. Similarly, the fourth line is whether or not the text file "ddpostprocess.out" shall be made. If so, the coordinates at which the electric nearfield is evaluated is decided by the rest of the lines. Each of the following lines consist of seven parameters. The first three are the x, y and z coordinates of a starting point. Similarly for the next three parameters and an ending point. The last parameter then determines how many points spaced linearly between the start and end point at which the electric field will be evaluated. The function DDPOSTPROCESS_file in App. B.1 modifies such a parameter file to include all the points on a certain grid, and the function Nearfield processes the output text file "ddpostprocess.out".

The error tolerance for the iterative solver to satisfy is set by the parameter TOL, and it must be reached within MXITER iterations or else the computation is aborted. GAMMA is the interaction cutoff parameter, γ, from Sec. 3.2.2. The angular resolution is only used for computation of radiative torques and the average scattering angle, so the parameter ETASCA is not of significance here. The

wave lengths for the incident light are given by four parameters. The first two parameters are the start and end wave lengths given in microns. Then, the number of wave lengths, and lastly, the spacing of the wave lengths. The spacing can be linear, inversely linear or logarithmic. The refractive index of the surrounding ambient medium is set by NAMBIENT. The lattice spacing, d, is not given as input. Instead, the effective radius, a_{eff}, from Eq. (2.4.21) is given. The lattice spacing can then be found by supplementing Eq. (2.4.22) with the number of point dipoles, N, from the target file. The parameters for the effective radius are given in the same manner as for the incident wave lengths.

The incident wave vector is parallel to the x-axis in the lab frame, $\hat{\mathbf{x}}_{\text{LF}}$. The first polarisation state to do calculations for is specified by the parentheses $(0,0)$ $(\text{Re}(\mathbf{e}_{y,\text{LF}}), \text{Im}(\mathbf{e}_{y,\text{LF}}))$ $(\text{Re}(\mathbf{e}_{z,\text{LF}}), \text{Im}(\mathbf{e}_{z,\text{LF}}))$, where $\mathbf{e}_{\text{LF}}$ is the polarisation vector in the lab frame. In order to compute the Mueller matrix elements, the polarisation state orthogonal to both the incident wave vector and the polarisation vector specified by the parentheses must be computed for. This is done by setting the parameter IORTH to 2. Otherwise, the parameter must be set to 1. The functions in App. B.1 making arrays of cross sections and Mueller matrix elements read from the .sca and .avg files. For these files to be made, the parameter IWRKSC must be set to 1. Then, the DDSCAT output files are generated for each wave length, orientation and effective radius that are computed for. This effectively works as a saving mechanism and an indication of computational progress.

The angles β, θ and ϕ describe the orientation of the target frame relative to the lab frame. The angle θ is the polar angle of the unit vector $\hat{\mathbf{a}}_1$ relative to the incident wave vector, $\hat{\mathbf{x}}_{\text{LF}}$. Similarly, the angle ϕ is the azimuthal angle of $\hat{\mathbf{a}}_1$ with respect to $\hat{\mathbf{x}}_{\text{LF}}$. Lastly, the angle β is the rotation of $\hat{\mathbf{a}}_2$ around $\hat{\mathbf{a}}_1$. The unit vectors $\hat{\mathbf{a}}_1$, $\hat{\mathbf{a}}_2$ and $\hat{\mathbf{a}}_3$ can be expressed in terms of the unit vectors of the lab frame

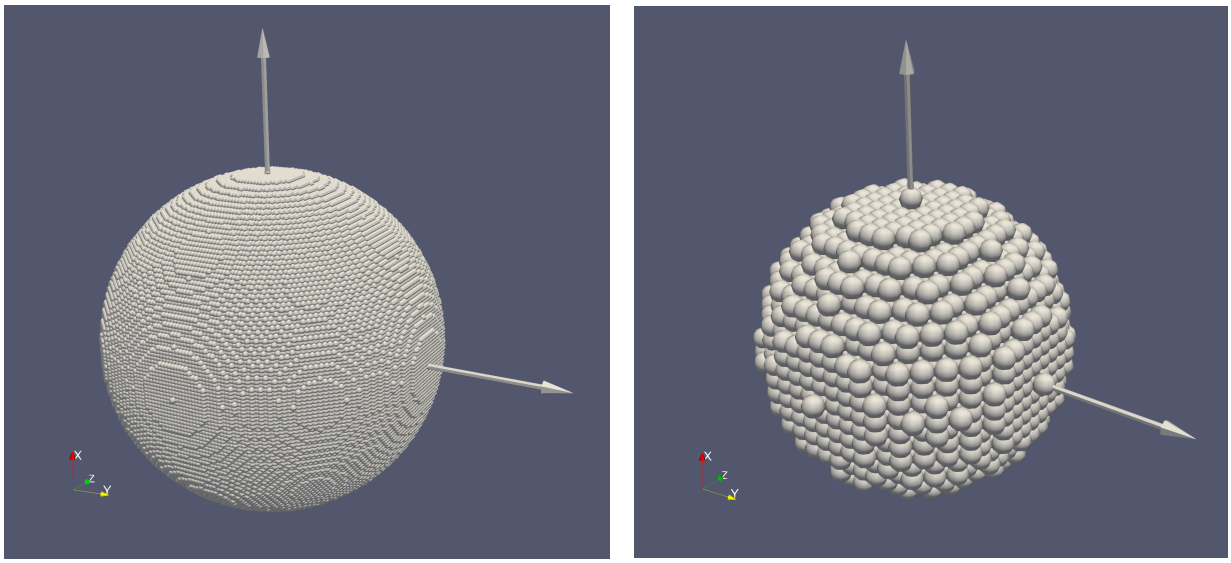
$$\begin{pmatrix} \hat{\mathbf{a}}_1 \\ \hat{\mathbf{a}}_2 \\ \hat{\mathbf{a}}_3 \end{pmatrix} = \begin{pmatrix} \cos\Theta & \sin\Theta\cos\Phi & \sin\Theta\sin\phi \\ -\sin\Theta\cos\beta & (\cos\Theta\cos\beta\cos\Phi - \sin\beta\sin\Phi) & (\cos\Theta\cos\beta\sin\Phi + \sin\beta\cos\Phi) \\ \sin\Theta\sin\beta & -(\cos\Theta\sin\beta\cos\Phi + \cos\beta\sin\Phi) & -(\cos\Theta\sin\beta\sin\Phi - \cos\beta\cos\Phi) \end{pmatrix} \begin{pmatrix} \hat{\mathbf{x}}_{\text{LF}} \\ \hat{\mathbf{y}}_{\text{LF}} \\ \hat{\mathbf{z}}_{\text{LF}} \end{pmatrix}. \tag{3.2.11}$$

The angles have each their parameters for start angle, end angle and total number of angles. The starting point for the incident wave lengths IWAV, effective radii, IRAD, and target orientations, IORI, can be chosen to be different from the first ones specified above. Otherwise, the number zero indicates to start in normal order. The number of Mueller matrix elements to print is specified by NSMELTS, and the next line contains the matrix indices for those Mueller matrix elements. The scattered directions must be specified by the target frame, 'TFRAME', for targets with periodic boundary conditions. At last is the number of diffraction orders from Eq. 2.4.33, and the the corresponding orders (M, N) are listed below. For the quasi-static limit, only the order $(0,0)$ will be valid.

3.2.5 The discretisation

The discretisation of the scattering objects can be visualised by displaying the point dipoles as small touching spheres in the open data analysis and visualisation application, ParaView [11]. DDSCAT has a routine called vtrconvert, which creates .pvd files from the target files as in App. B.4. The nanoHUB tool DDSCAT Shape Generator [10] can be used to generate a target file for a pseudosphere, which is a discretisation of a regular sphere. The target file can then be modified by the python function cut_in_half in App. B.1 to represent a half pseudosphere, or by two_spheres to represent a dimer. The full and half pseuospheres are shown in Figs. 3.2–3.3, respectively, and the dimer in Fig 3.4. Furthermore, the directions of the unit vectors specified in the parameter file,

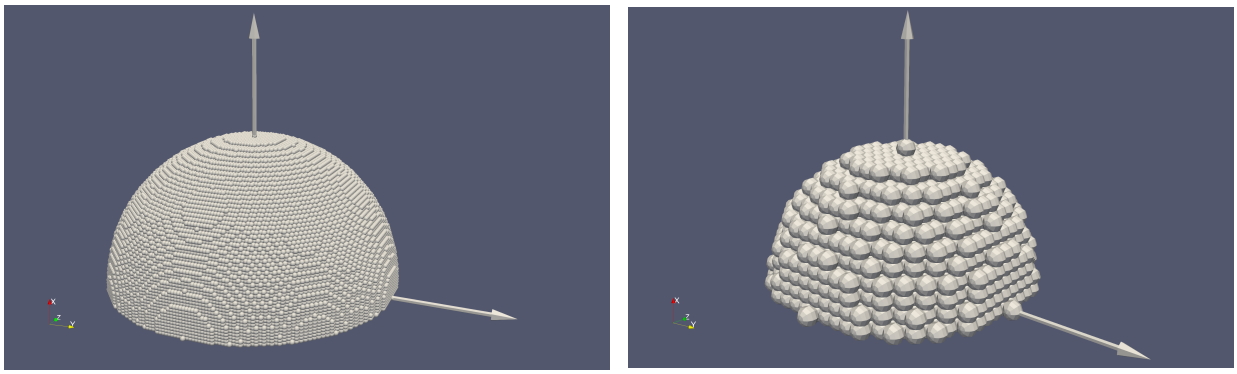
$\hat{\mathbf{a}}_1$ and $\hat{\mathbf{a}}_2$, are illustrated as arrows. The pseudosphere created by the DDSCAT Shape Generator consists of an odd number of point dipole planes. Hence, the half sphere remaining after cut in half contains one plane of point dipoles more than the half sphere removed. Consequently, the number of point dipoles in Fig. 3.3 are slightly greater than half of the number of point dipoles in Fig. 3.2.



(a) A pseudosphere made of 523 305 point dipoles.

(b) A pseudosphere made of 4169 point dipoles.

FIGURE 3.2: Two spheres discretized by point dipoles visualised as small spheres. The arrow pointing upwards is parallel to the unit vector $\hat{\mathbf{a}}_1$, and the other one is parallel to the unit vector $\hat{\mathbf{a}}_2$. The unit vectors are parallel to the two vectors in the target frame, $\hat{\mathbf{x}}_{\mathrm{TF}}$ and $\hat{\mathbf{y}}_{\mathrm{TF}}$, respectively.



(a) A half pseudosphere made of 265 575 point dipoles.

(b) A half pseudosphere made of 2243 point dipoles.

FIGURE 3.3: Two half spheres discretized by point dipoles visualised as small spheres. The arrow pointing upwards is parallel to the unit vector $\hat{\mathbf{a}}_1$, and the other one is parallel to the unit vector $\hat{\mathbf{a}}_2$. The unit vectors are parallel to the two vectors in the target frame, $\hat{\mathbf{x}}_{\mathrm{TF}}$ and $\hat{\mathbf{y}}_{\mathrm{TF}}$, respectively.

3.3 Implementation

To solve the system of equation in Eq. (3.1.3), the python script in App. A.1 is developed. To avoid overflow in the computation of the binomial coefficients used in $H(l_j, m_j|l_i, m_i)$ from Eq. (2.2.42),

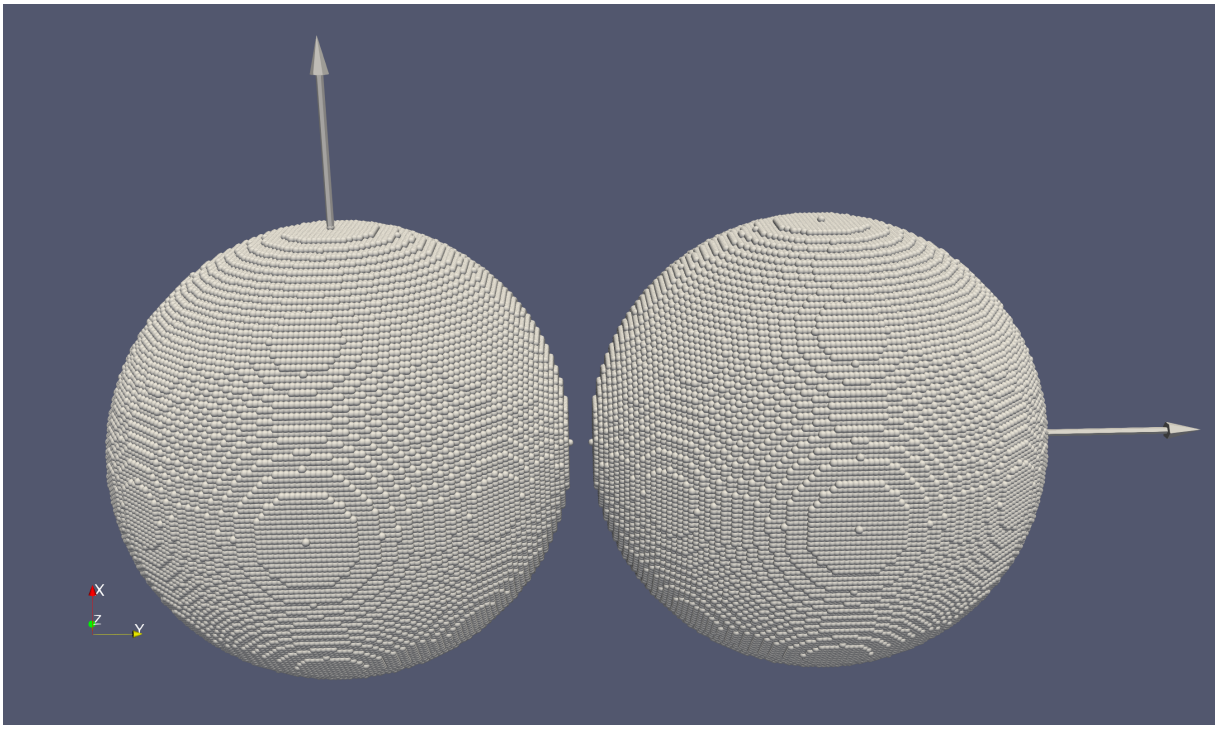


FIGURE 3.4: A dimer consisting of two pseudospheres of 523 305 point dipoles each. The spheres are separated by a distance of 5 lattice sites or a tenth of the radius of the spheres. The arrow pointing upwards is parallel to the unit vector $\hat{\mathbf{a}}_1$, and the other one is parallel to the unit vector $\hat{\mathbf{a}}_2$. The unit vectors are parallel to the two vectors in the target frame, $\hat{\mathbf{x}}_{\mathrm{TF}}$ and $\hat{\mathbf{y}}_{\mathrm{TF}}$, respectively.

Scipy's function binom from the Special functions library is used. The same library also offers a function to compute spherical harmonics, but the function is limited to $l < 86$, which limits $l_j < 43$. Instead, the spherical harmonics are computed using the freely available software, SHTOOLS [5]. The software for the python language is called PYSHTOOLS [5], and the function used is the spharm from the library Expand. The function uses the standard three- term recursion formula with the scaling approach of S. Holmes and W. A. Featherstone, and the results are accurate to about degree 2800 [5]. Creating the $\mathbf{C}$ matrix is sped up with the just-in-time compiler, Numba [4]. The compiler translates code to machine level, and it allows for parallelisation of loops with the function prange. The asymptotic time complexity of creating the $\mathbf{C}$ matrix on an Intel® Core™ i7-10700 processor for a finite set of N spheres is $9 \cdot 10^{-9}(MN)^2$s. To solve for $\mathbf{A}$ in Eq. (3.1.3), the iterative solver gmres from the Scipy library Sparse linear algebra is utilised. The function uses generalised minimal residual iterations with a start guess based on the solution found from the previous energy of the incident field. The asymptotic time complexity of the routine gmres is $6 \cdot 10^{-9}(MN)^2$s on the same processor when the tolerance chosen is 10^{-9}. An alternative iterative solver is the bicgstab from the same library, which uses biconjugate gradient stabilised iterations. The iterativ solver bicgstab has a slightly higher asymptotic time complexity at $7 \cdot 10^{-9}(MN)^2$s. For the first energy, there is no previous solution to use as a starting iteration. The solver used then is the standard LU factorising solve from the Numpy library Linear algebra, which has an asymptotic time complexity of $7 \cdot 10^{-12}(MN)^3\mathrm{s} + 2 \cdot 10^{-9}(MN)^2$s. The asymptotic space complexity of the python script in App. A.1 is dominated by the matrix $\mathbf{C}$, which has an asymptotic space complexity of $16(MN)^2$B. For an infinite lattice, the space complexity for the $\mathbf{C}$ matrix and the time complexity for solving for the A coefficients remain unchanged from the complexities for the finite system, as only the

coefficients for one unit cell are solved for. The time complexity for building the $\mathbf{C}$ matrix on the other hand increases considerably with the interaction distance squared, $\sim R_{\mathrm{int}}^2$.

The Fortran software DDSCAT is compiled using the (classic) Intel® Fortran compiler version 2021.2.0 in order to take advantage of the Intel® oneAPI Math Kernel Library version 2021.2.0 when executing the FFT computations. This is specified in the Makefile as illustrated in App. B.2. The computations are parallelised by the use of OpenMP. There is an option for using the Intel® MPI Library as well, but this only allows for multiple target orientations to be computed for simultaneously. As only two target orientations are computed for here, which are normal incidence and an angle of incidence at 45°, the different orientations are run on different computers separately. The analytical Mie result of the absorption cross section is obtained from the code BHMIE [12] translated into python by H. Kaiser [13].

Chapter 4

Results and discussion

In this chapter the results obtained on the basis of the formulation presented in Ch. 2 is presented. The chapter is divided into three sections. The first section regards systems of finite number of particles. The second section regards the transition to systems of particles repeated infinitely on a lattice. The third section regards the comparison of DDSCAT to the multipole expansion method and GranFilm. The aim is to verify the results obtained from App. A.1 and compare them to the results obtained from the software DDSCAT.

4.1 Finite systems

In order to quality check the implementations of App. A.1, the results from Figs. 2–5 from Ref. [2] are reproduced in Figs. 4.1–4.4. The dipole moments from Eq. (2.2.89) are only computed for three incident electric fields, each parallel to one of the three Cartesian axes. Any incident field will be a linear combination of those three and likewise for the corresponding dipole moments. Furthermore, the shifts in energy positions of the resonances are studied along with the near fields. The near field obtained from the multipole expansion method is compared to the near field obtained from DDSCAT.

4.1.1 Dimensionless dipole moments

Figure 4.1 presents the dimensionless dipole moment for a single Drude sphere with a height $h = 0.05a$ above a substrate with different dielectric functions in the three subplots. The first function equals the one of the ambient medium, $\varepsilon_- = \varepsilon_+ = 1$. The resonance occurs at $\hbar\omega = \sqrt{3}$ eV $\approx 1.73\,\text{eV}$, which agrees with the Mie result from Sec. 2.3.1. The dipole moment is also completely independent of the direction of the incident electric field. Next, when increasing the dielectric function of the substrate, the resonance peaks are red shifted. The dipole moment now depends on the azimuthal angle of the incident electric field, ϕ_0. The amplitudes of the peaks decrease, and more peaks occur with the stronger substrate interaction. The dipole moment red shifts more in the case of an incident electric field parallel to the z-axis than for the x-axis. This can be understood in terms of the results presented in Sec. 2.1.5. When the incident field is orthogonal to the substrate, the dipole moments from the sphere and the substrate line up in series and thus add in strength as presented in Fig. 2.1. In the case where the incident electric field is parallel to the substrate on the other hand, the dipole moments from the sphere and the substrate end up pointing in opposite directions. The field strength is then reduced to quadrupole order.

In Fig. 4.2 another sphere is added on the x-axis, separated by a distance $d = 0.1a$ from the

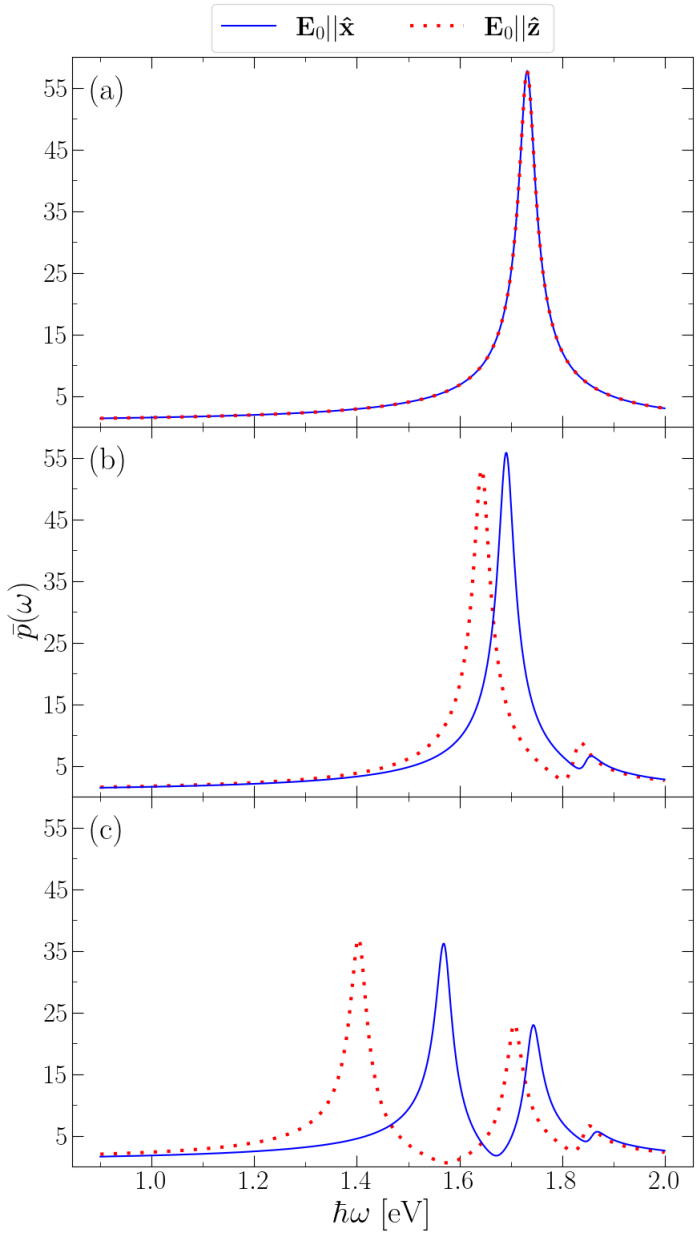


FIGURE 4.1: The dimensionless dipole moment, $\bar{p}(\omega)$, for a single nanosphere of radius a, placed a height $h = 0.05a$ above a substrate with dielectric function (a) $\varepsilon_- = 1$, (b) $\varepsilon_- = 2$ and (c) $\varepsilon_- = 10$. The ambient medium surrounding the sphere is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric function of the sphere follows the Drude model with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. The orthogonal sum truncation in the multipole expansion used is $L_\perp = 50$.

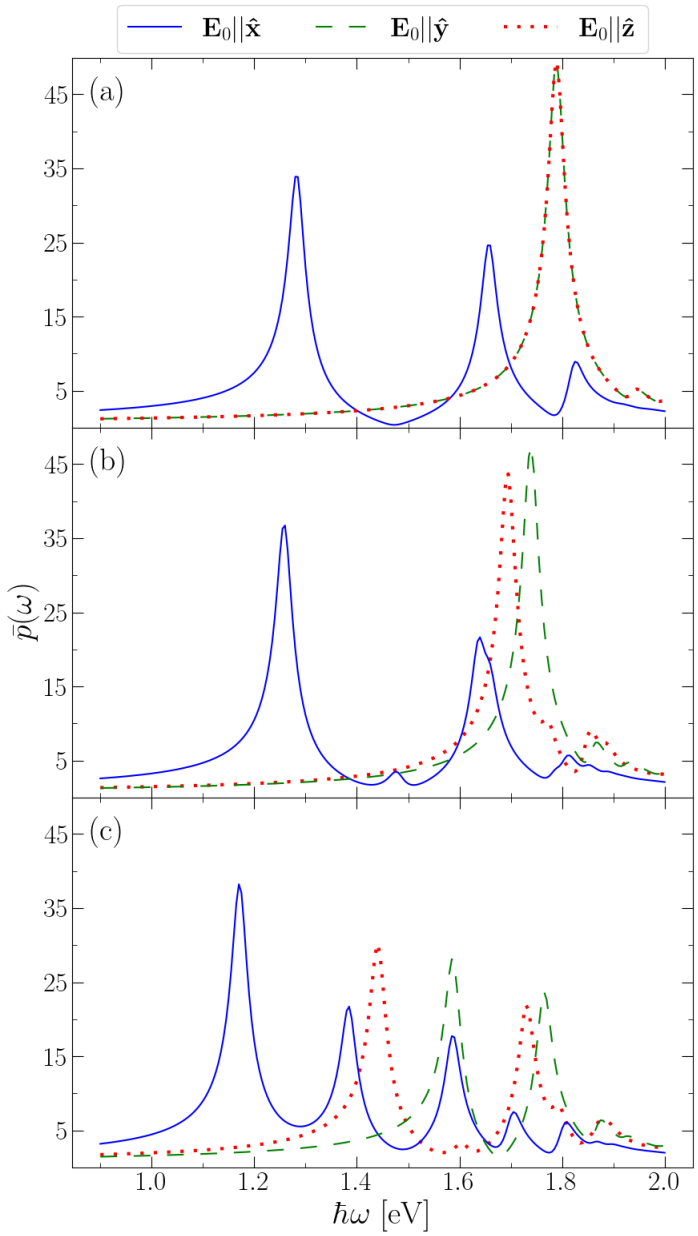


FIGURE 4.2: The dimensionless dipole moment, $\bar{p}(\omega)$, for two nanospheres of radius a, separated by a distance $d = 0.1a$ along the x-axis and placed a height $h = 0.05a$ above a substrate with dielectric function (a) $\varepsilon_- = 1$, (b) $\varepsilon_- = 2$ and (c) $\varepsilon_- = 10$. The ambient medium surrounding the spheres is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric functions of the spheres are modelled by the Drude form with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. Both the orthogonal and parallel sum truncations in the multipole expansion used are $L = 50$.

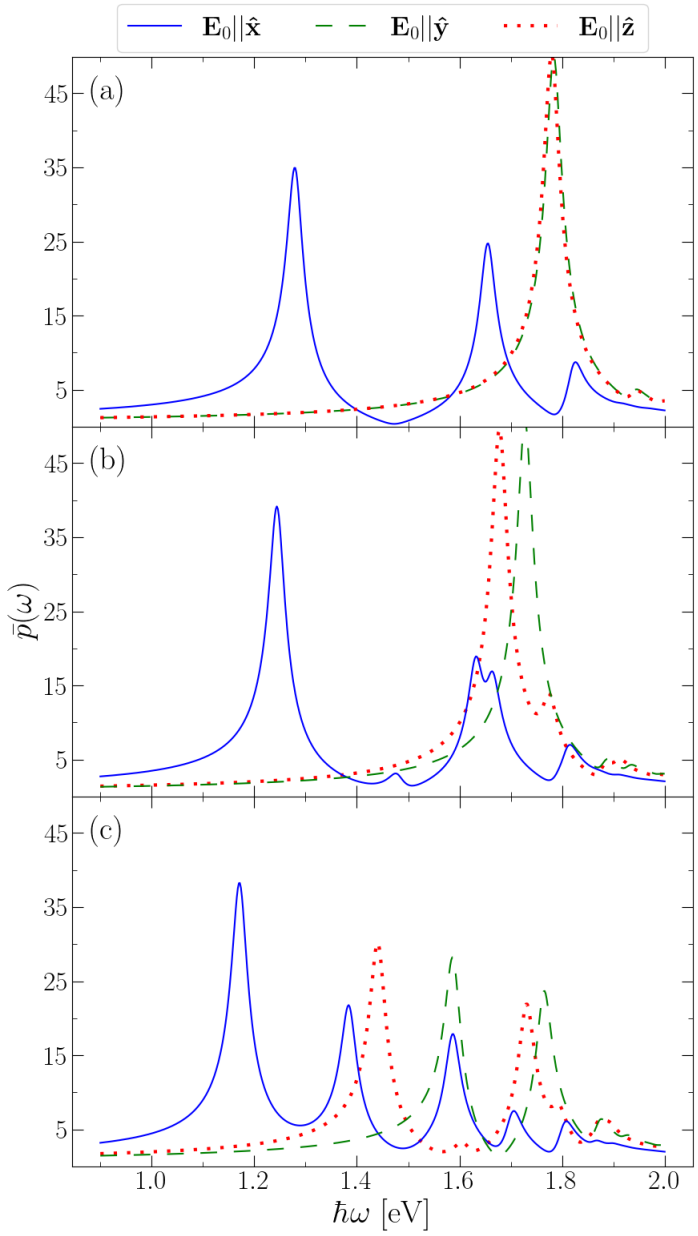


FIGURE 4.3: The dimensionless dipole moment, $\bar{p}(\omega)$, plotted against energies of the incident wave for two nanospheres of radius a, separated by a distance $d = 0.1a$ along the x-axis and placed a height (a) $h = 2a$, (b) $h = 0.3a$ and (c) $h = 0.05a$ above a substrate with dielectric function $\varepsilon_- = 10$. The ambient medium surrounding the spheres is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric functions of the spheres are modelled by the Drude form with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. Both the orthogonal and parallel sum truncations in the multipole expansion used are $L = 30$.

one in Fig. 4.1. The choice of these positions results in the distance between a sphere's multipole and its image multipole matching the distance between a sphere's multipole and its neighbour's multipole, $d = 2h$. This time, the no substrate case is only symmetric around the x-axis. The resonance energy position is not located at $\hbar\omega = \sqrt{3}\text{eV} \approx 1.73\,\text{eV}$ as previously, but instead blue shifted to $\hbar\omega \approx 1.79\,\text{eV}$. For all three subplots in Fig. 4.2, the resonance peaks, when the incident electric field is parallel to either the z- or y-axis, are quite similar to Fig. 4.1, but they are blue shifted slightly and decreased in amplitude. This is caused by the induced charge concentrations in the neighbour sphere being symmetric instead of anti-symmetric and thus having the same sign as illustrated later in Sec. 4.1.4. Consequently, the local field enhancements would be smaller than for the single sphere case. The Coulomb force from the neighbour's charge concentrations will then act as a restoring force. When the incident electric field is parallel to the x-axis on the other hand, the red shift is much larger and more peaks appear. The higher dielectric function of the substrate, the larger red shift and more peaks occur. The dipole moments for the spheres here line up in series as they did in Fig. 2.2. Hence they somewhat double in strength.

The red shifts in the resonance peaks follow from stronger interactions with either the substrate or with the other sphere. When the dipole moments line up in series the interactions are at their strongest. Similarly, they're at their weakest when they face opposite directions, as they turn to quadrupole order in strength. The interactions result in higher induced charge concentrations, which again may lead to a stronger local electric field than the external incident electric field. Hence, with a stronger electric force from the local electric field, the resonance can occur for lower wave energies of the incident plane wave. The reason for the interaction between the spheres to be greater than the interaction with the substrate comes from the image charges not being equal to the ones of the spheres. The image charges depend on the dielectric function of the substrate, as the boundary condition on the surface of the substrate must be satisfied. Hence, the two dipole moments for the spheres add up to more than for the sphere and the substrate, and are consequently red shifted more even though they are separated by the same distance.

The substrates in Fig. 4.3 have all dielectric functions $\varepsilon_- = 10$, but the heights of the spheres above the substrate vary. When $h = 2a$, the dipole moments are almost identical to the ones with no substrate, but very slightly red shifted. For $h = 0.3a$, the dipole moments with incident electric field along the z- and y-axis, separately, are still quite similar to Fig. 4.2(b), but also slightly red shifted. For the height $h = 0.05a$, the case is the same as for Fig. 4.2(c), but computed to order $L = 30$ instead of 50. As the plots seem identical, the dipole moment has converged for eye precision.

The origin of the leftmost resonance peak in Figs. 4.2(a) and 4.3(a), where there is little to no substrate interaction, may be assumed to be the inter-particle interaction. The peak remains as the leftmost peak, even when increasing the substrate interaction to Figs. 4.2(c) and 4.3(c). During this increase, a new peak begins to arise in Figs. 4.2(b) and 4.3(b) next to the leftmost peak. The peak may be assumed to originate from the substrate interaction and can be related to the leftmost peak in Fig 4.1(c). The rest of the peaks in Figs. 4.2 and 4.3 are more difficult to determine. The resonances are spaced more closely such that their evolutions coincide when increasing the substrate interaction as for the two peaks in Fig. 4.3(b). The left one is fading away and the right one is growing larger for increasing substrate interaction. As the same peak is not split into two yet in Fig. 4.3(b), the substrate interaction may be weaker there than in Fig. 4.2(b). This agrees with the red shifts of the peaks with incident electric field parallel to the other two axes being greater in Fig. 4.3(b) than in Fig. 4.2(b).

4.1.2 Visualisation of the red shifts

Figure 4.4 shows the energy positions of the lowest energy resonances for the dipole moments with incident electric field orthogonal to the substrate when varying the height h. For the lowest truncation, $L = 1$, the energy positions are greater than the other ones at all heights. At the second truncation order, the resonances occur at the same energies as the other orders, except for $L = 1$, for large values of h, but then they go astray. The higher the truncation order, the lower values of h are required to separate the energy positions from even higher orders of truncation. The plot is quite informative for detecting when a certain order of truncation will definitely not yield converged results.

The dipole moments from Fig. 4.1(c) are computed for 50 heights, h, and presented as a contour plot by Fig. 4.5. The heights are logarithmically spaced from $0.05a$ to a. As h decreases, the resonance peaks are red shifted, become narrower and of smaller amplitude. For $h < 0.15a$, a second resonance peak appears. Similarly, the contour plot shown by Fig. 4.6 includes the dipole moments from Fig.4.2(c) computed for 50 distances d logarithmically spaced from $0.1a$ to $2a$. For the incident electric field parallel to the y- and z-axis, separately, the resonance peaks demonstrate the blue shifts from comparing Fig. 4.2 (a) to Fig. 4.1 (a). For the field parallel to the x-axis on the other hand, the resonance peaks red shift significantly. The second peak ceases around $d = 0.6a$, where a new peak appears to the left of the first one. For $d < 0.2a$ up to four peaks are present.

A major contribution to the red shifts of the resonance peaks in Figs. 4.5 and 4.6(b) can be found from writing out the denominators for the coefficients $A_{1,m}$ at dipole order. For a single sphere above a substrate, this is done in Sec. 2.3.2. Figure 2.6 presents the required energies for the real part of these denominators to become zero for different heights above the substrate. The

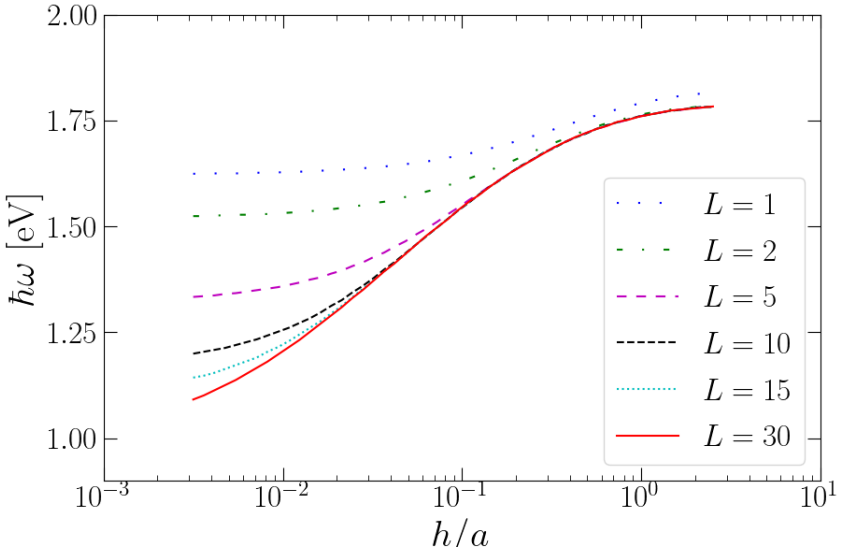


FIGURE 4.4: The lowest resonance energy positions of the dimensionless dipole moment, $\bar{p}(\omega)$, for two nanospheres of radius a separated by a distance $d = 0.1a$ along the x-axis. The energy positions are plotted for the height, h, above a substrate with dielectric function $\varepsilon_- = 10$. The incident electric field is parallel to the z-axis, and the ambient medium surrounding the spheres is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric functions of the spheres are modelled by the Drude form with plasma frequency $\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\gamma = 0.03\,\text{eV}$. The orthogonal and parallel sum truncations in the multipole expansion are set equal and are denoted by L.

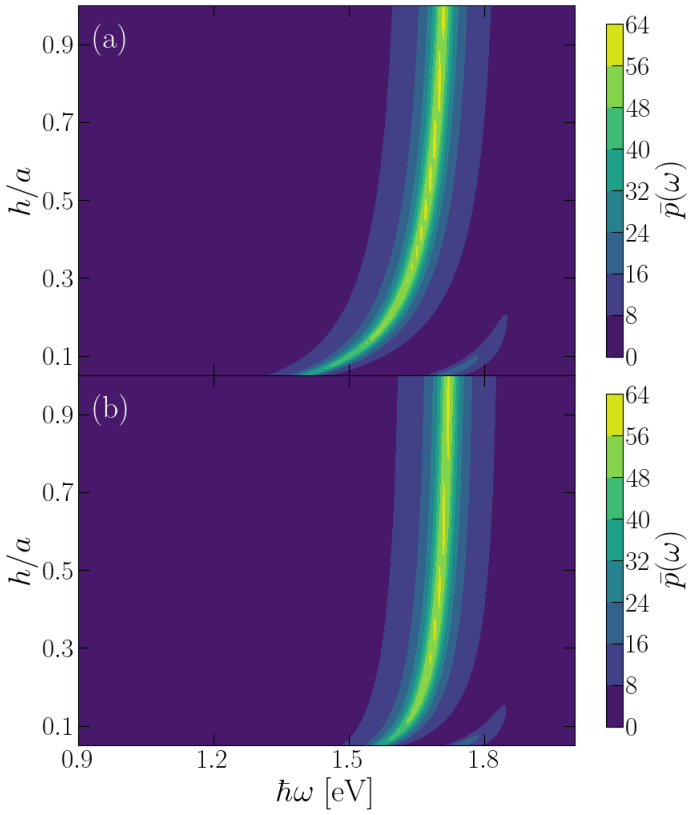


FIGURE 4.5: The dimensionless dipole moment, $\bar{p}(\omega)$, of a single nanosphere of radius a plotted at the first axis against energies of the incident wave and at the second axis against heights h above a substrate with dielectric function $\varepsilon_- = 10$. The ambient medium surrounding the sphere is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric function of the sphere is modelled by the Drude form with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. The direction of the incident electric field is parallel to (a) the z-axis and (b) the x-axis. The orthogonal sum truncation in the multipole expansion used is $L_\perp = 70$.

dielectric parameters used are the same as in Fig. 4.5, and the red shift from decreasing the height is evident. The resonance peaks for the single sphere in Fig. 4.5 closely follow the resonance energy positions in Fig. 2.6 for large values of h. When decreasing h, the resonance peaks computed at high order red shift more than the analytical values at dipole order. This may be a consequence of the dipole order dominating the far field region, and the higher order strengthen more at shorter distances as shown in Sec. 2.1.5. When the incident electric field is orthogonal to the substrate, the resonance peak in Fig. 4.5(a) follows closest to the one found from the denominator of $A_{1,0}$. They are very close until the height becomes smaller than $0.4a$. Similarly for when the incident field is parallel to the substrate, the resonance peak in Fig. 4.5(b) follows closest to the one found from $A_{1,\pm1}$, and they are very close until the height becomes smaller than $0.3a$. This agrees well with the Cartesian components of the dipole moment, as the dipole moment in z-direction only depends on $A_{1,0}$, and for the x- and y-direction, the dipole moments depend on both $A_{1,1}$ and $A_{1,-1}$. The dipole order approximation holding for lower heights when the incident electric field is parallel to

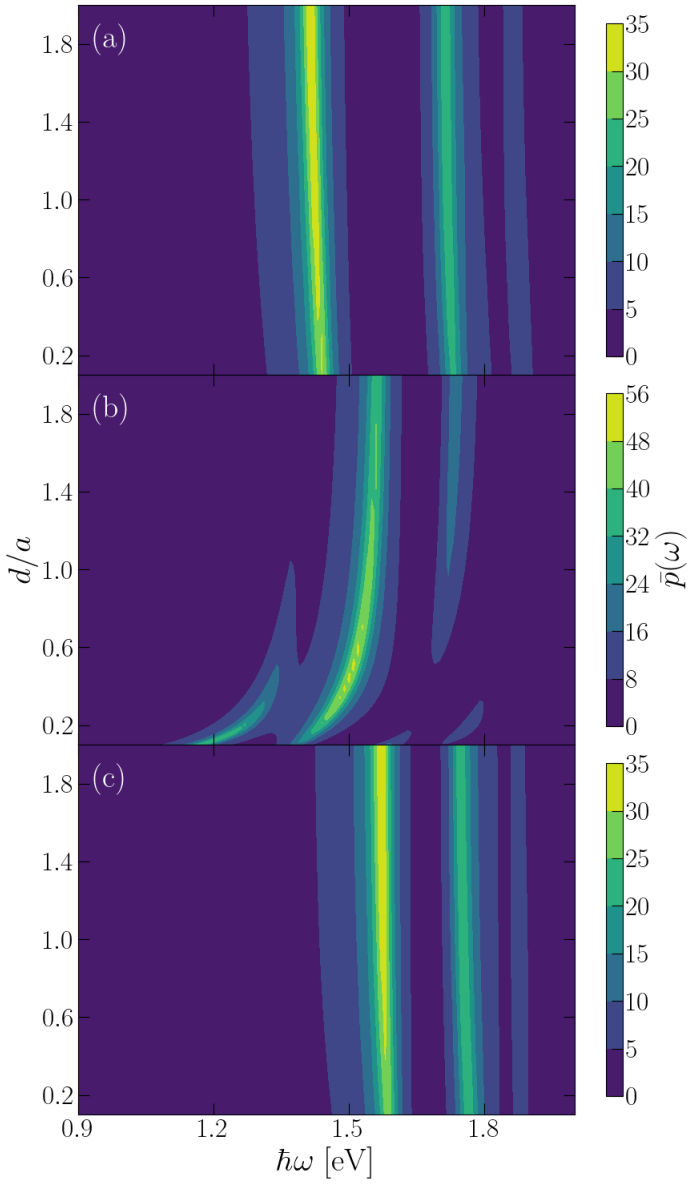


FIGURE 4.6: The dimensionless dipole moment, $\bar{p}(\omega)$, of two nanospheres of radius a plotted at the fist axis against energies of the incident wave and at the second axis against the distance d separating the spheres along the x-axis. They are placed a height $h = 0.05a$ above a substrate with dielectric function $\varepsilon_- = 10$. The ambient medium surrounding the spheres is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric functions of the spheres are modelled by the Drude form with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. The direction of the incident electric field is parallel to (a) the z-axis, (b) the x-axis and (c) the y-axis. The orthogonal and parallel sum truncations in the multipole expansion are set equal at $L = 60$.

the substrate also agrees with the substrate interactions being weaker than the ones for the incident electric field orthogonal to the substrate.

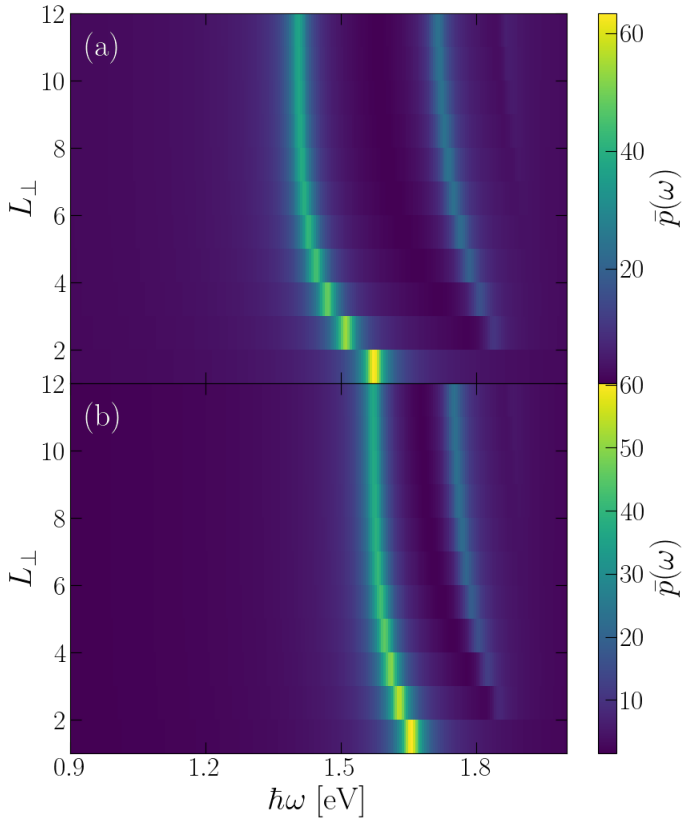


FIGURE 4.7: The dimensionless dipole moment, $\bar{p}(\omega)$, of a single nanosphere of radius a plotted at the first axis against energies of the incident wave and on the second axis against the orthogonal sum truncation in the multipole expansion. The sphere is placed a height $h = 0.05a$ above a substrate with dielectric function $\varepsilon_- = 10$. The ambient medium surrounding the sphere is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric function of the sphere is modelled by the Drude form with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. The direction of the incident field is parallel to (a) the z-axis and (b) the x-axis.

For two spheres placed a height $h = 0.05a$ above the substrate as in Fig. 4.6(b), the denominator of the coefficients $A_{1,m}$ is written out at dipole order in Sec. 2.3.3. The coefficients all share the same denominator this time, and it has two minimums causing resonances. The energy positions of these resonances are plotted in Fig. 2.7 as a function of the distance d separating the spheres along the x-axis. The dashed line lies close to the solution for the real part of the denominator being zero. The drawn line however is far away from the solution for the imaginary part being zero. In Fig. 4.6(b), the largest resonance peak starts following the drawn line from Fig. 2.7. For lower values of d, the peak in Fig. 4.6(b) red shifts more as in the case with a single sphere. The dashed line however misses the second resonance peak in Fig. 4.6(b) for all distances d. This could be caused by the strong interaction with the substrate, as the spheres are placed $h = 0.05a$ above it. The dipole order hence does not give the same representation even when the spheres are far

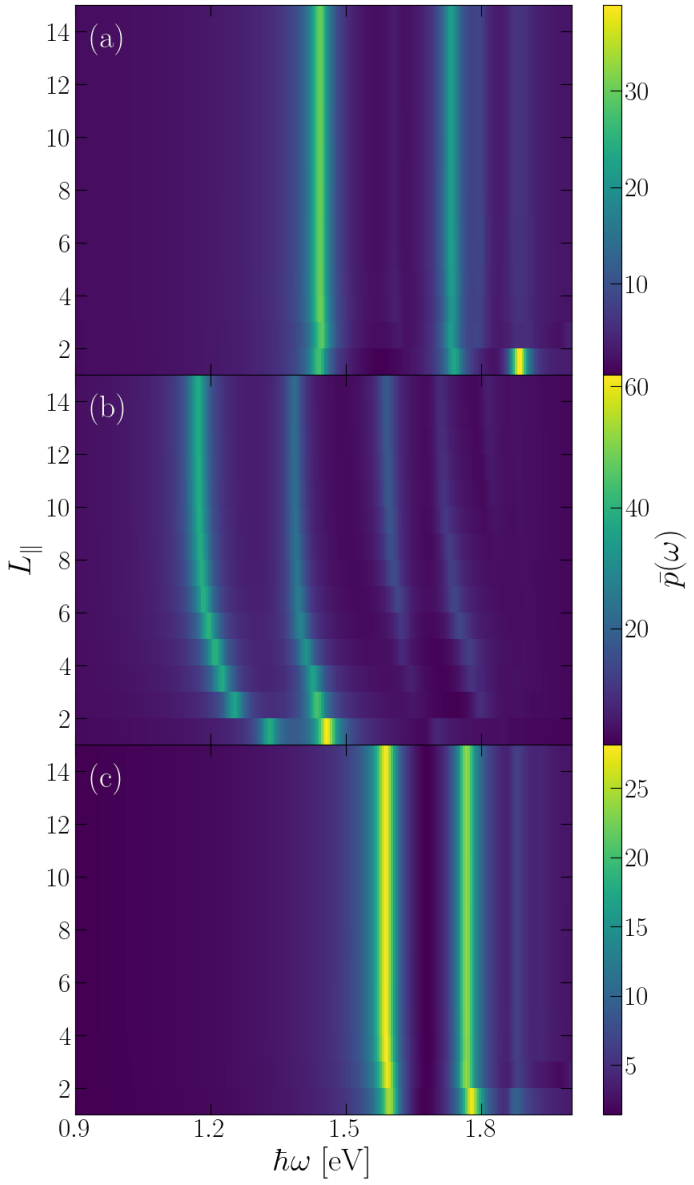


FIGURE 4.8: The dimensionless dipole moment, $\bar{p}(\omega)$, of two nanospheres of radius a plotted at the first axis against energies of the incident wave and on the second against the parallel sum truncation in the multipole expansion. The spheres are placed a height $h = 0.05a$ above a substrate with dielectric function $\varepsilon_- = 10$, and are separated by a distance $d = 0.1a$ along the x-axis. The ambient medium surrounding the spheres is vacuum with a dielectric function $\varepsilon_+ = 1$. The dielectric functions of the spheres are modelled by the Drude form with plasma frequency $\hbar\omega_p = 3\,\text{eV}$ and the inverse of the free carrier relaxation time $\hbar\gamma = 0.03\,\text{eV}$. The direction of the incident field is parallel to (a) the z-axis, (b) the x-axis and (c) the y-axis. The orthogonal sum truncation in the multipole expansion is $L_\perp = 30$.

apart from each other. With dipole order in the interactions between the two spheres, increasing the multipole order of the substrate interactions could be used to determine where the dipole order approximation in parallel truncation breaks down. For the case in Fig. 4.6(b), that would have been at $d \approx 0.5a$, which in a hexagonal lattice would correspond to a surface density of $\approx 58\%$. The new peaks appearing for low values of d are coming from the higher order terms. Only the dipole moment with the incident electric field parallel to the axis through the centres of the two spheres appears to be affected by the distance separating the spheres. This shows the interaction between the spheres are only significant when their dipole moments line in series. The neighbour interactions for the other two directions of incidence compared to the substrate interactions are quite weak and induce the same blue shift as discussed in Sec. 4.1.1 when comparing Figs. 4.2–4.3 to Fig. 4.1.

Figure 4.7 presents the dipole moments from Fig. 4.1(c) computed for various truncations $L_\perp$ in the orthogonal multipole expansion. The resonance peaks red shift when the order increases, and the red shifts occur faster the earlier the sum is truncated. When the slopes of the curves connecting the resonance peaks become vertical and thus very little shifted from the last truncation, the dipole moment is close to the convergence. Similarly, Fig. 4.8 presents the dipole moments from Fig. 4.2(c) computed for various truncations $L_\parallel$ in the parallel multipole expansion with the orthogonal truncation, $L_\perp = 30$. For the incident electric field parallel to the y- and z-axis, separately, the resonance peaks stabilise for very low parallel truncations, $L_\parallel$. For the electric field incident along the y-axis, there is some red shift for the lowest truncations. For incident electric field parallel to the z-axis on the other hand, there appear to some blue shift for the lowest truncations. Lastly, for the incident electric field parallel to the x-axis, there is a strong red shift all the way up to $L_\parallel = 8$. The dipole moment does not seem to converge until $L_\parallel = 13$ because of the resonance peak on the right appearing after $L_\parallel = 10$. There is not much physics related to which way these peaks shift as function of truncation. It is the rates at which the peaks shift that are of importance, as they give a hint of how fast the solution converges.

4.1.3 Ring structures

The spheres can be placed to form a ring in the xy-plane such as illustrated in Fig. 4.10(a). There, 24 spheres are placed with a centre to centre distance of $2.1a$. The ring structure has a rotational symmetry around the z-axis, which intercepts the centre of the ring. In order to obtain a split ring, one of the spheres are removed as in Fig. 4.10(b). If the ring is oriented such that two spheres are placed on each axis, and one of those four are removed, there will still be a mirror symmetry with respect to the axis where the sphere is removed from. In Fig. 4.10(b) the sphere is removed from the x-axis at the right end. The dimensionless dipole moment for the spheres in Fig. 4.10 are presented in Fig. 4.9. For the full ring in Fig. 4.9(a), it is apparent that the dipole moments are linear combinations of the the two dipole moments with incident electric field parallel to the x-axis and to the y-axis. For the other directions of incidence the dipole moments are contained within those two. Furthermore, there are only seven unique outcomes, as the dimensionless dipole moments are invariant under a 180° rotation of the incident electric field around any axis. The dimensionless dipole moments for the spheres mirrored by either the x- or the y-axis are hence identical as indicated by the colours of the spheres in Fig. 4.10. The largest and leftmost peak is caused by the interaction with the closest neighbours. Consequently, the resonance is strongest when the incident electric field is parallel to the axis through the centres of the two neighbouring spheres, and thus the resonance is weakest when the incident electric field is perpendicular to this axis.

When removing one of the spheres at the x-axis in Fig. 4.9(b)–(c), the mirror symmetry with respect to the y-axis breaks down as illustrated by the colours of the spheres in Fig. 4.10. However,

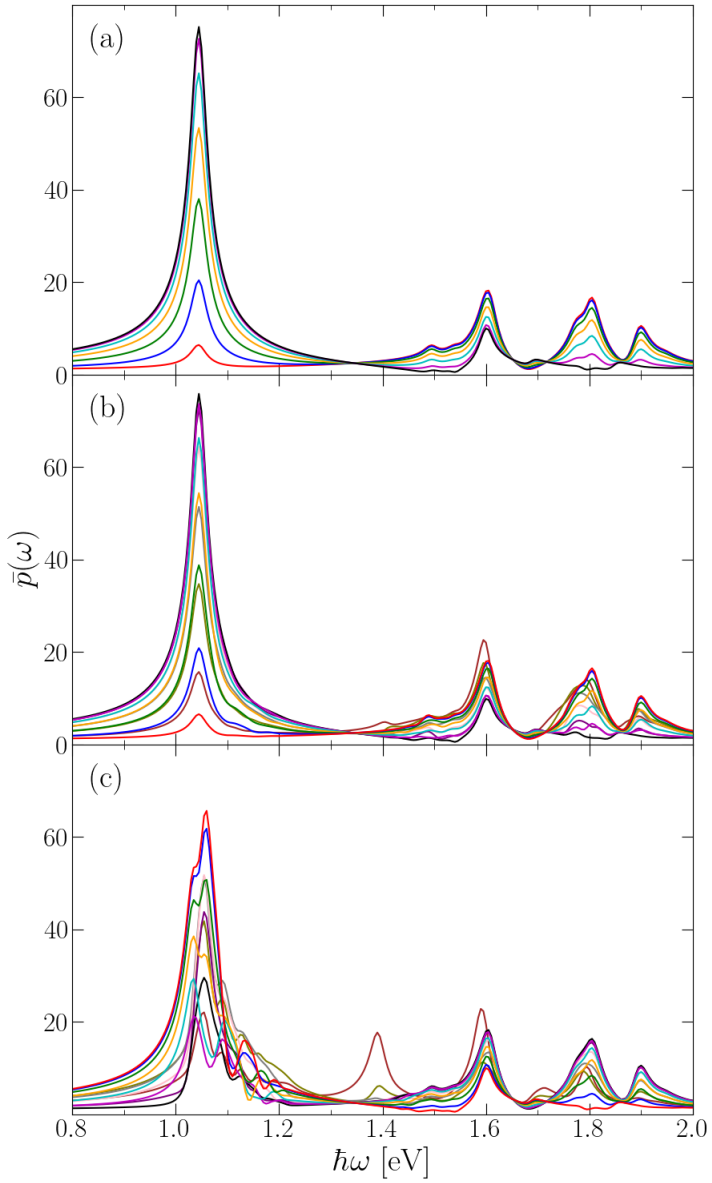


FIGURE 4.9: The dimensionless dipole moment, $\bar{p}(\omega)$, plotted for incident wave energies for each sphere in (a) a full ring and in (b)–(c) a split ring. The incident electric field is parallel to the x-axis in (a)–(b) and the y-axis in (c). The rings are placed in vacuum a height $h = 0.05a$ above a substrate with dielectric function $\varepsilon_- = 10$, where a is the radius of the spheres. There are 24 spheres in the full ring and 23 in the split ring. Both structures have centre in $(x, y) = (0, 0)$, and the spheres are separated by a distance $d = 0.1a$. The dielectric function of the spheres follows the Drude model with $\hbar\omega_p = 3\,\text{eV}$ and $\hbar\gamma = 0.03\,\text{eV}$. The split ring is a full ring without one of the spheres at $y = 0$. The multipole order used is $L = 30$. The colours of dipole moments correspond to the colours of the spheres in Fig. 4.10.

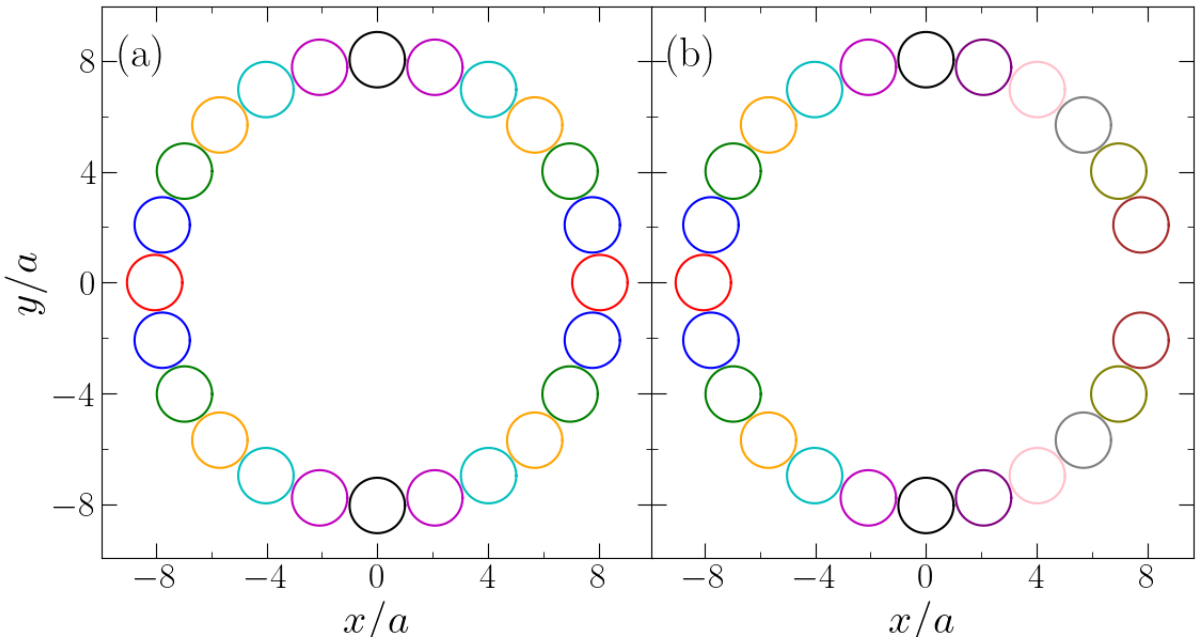


FIGURE 4.10: The structure of (a) a full ring and (b) a split ring. The structures consist of (a) 24 and (b) 23 spheres with radius a. The centre of the structures are located at $(x, y) = (0, 0)$. The spheres are separated by a distance $d = 0.1a$. The split ring is equal to the full ring except for missing one of the two spheres located at the x-axis. The spheres with same colours share the same dimensionless dipole moment for incident electric field parallel to either the x- or the y-axis.

there is still a mirror symmetry with respect to the x-axis resulting in 12 unique outcomes contrary to 23. When the incident electric field is parallel to the x-axis in Fig. 4.9(b), the dipole moments are quite similar to the ones for the full ring. The asymmetry is present in the slight variations in the pairs of spheres mirrored by the y-axis. The dipole moment of the leftmost sphere is barely affected at all, as it is the furthest one away from the gap. When the spheres are closer to the gap, the difference with respect to Fig. 4.9(a) increases. The resonance peak second from the right is where the dipole moments are changed the most. This suggests the peak is caused by interactions along the diameter of the ring. When the electric field is incident along the y-axis as in Fig. 4.9(b), the leftmost resonance peak is completely distorted. Even the dipole moment for the sphere furthest away from the gap is drastically changed from Fig. 4.9(a). The gap interrupting the neighbour coupling at its strongest point affects the rest of the ring much more than it did at its weakest point as one would expect. This demonstrates how the expansion coefficients are strongly coupled to each other.

4.1.4 Near field calculations

The near fields for the system in Figs. 4.2(c) and 4.3(c), with the incident wave energy $\hbar\omega = 1.35\,\text{eV}$ and the incident electric field parallel to the x-axis, are presented in Fig. 4.11. The electric field enhancements from Eqs. (2.2.30)–(2.2.32) are in Fig. 4.11(a) a recreation of Fig. 7(a) in Ref. [2] for verification. The numerical values appear to agree well, especially in the hot spots between the spheres and the substrate and between the spheres. The electric field is discontinuous at the surface of the spheres and of the substrate as expected from Eq. (2.1.4). The electric field has a mirror

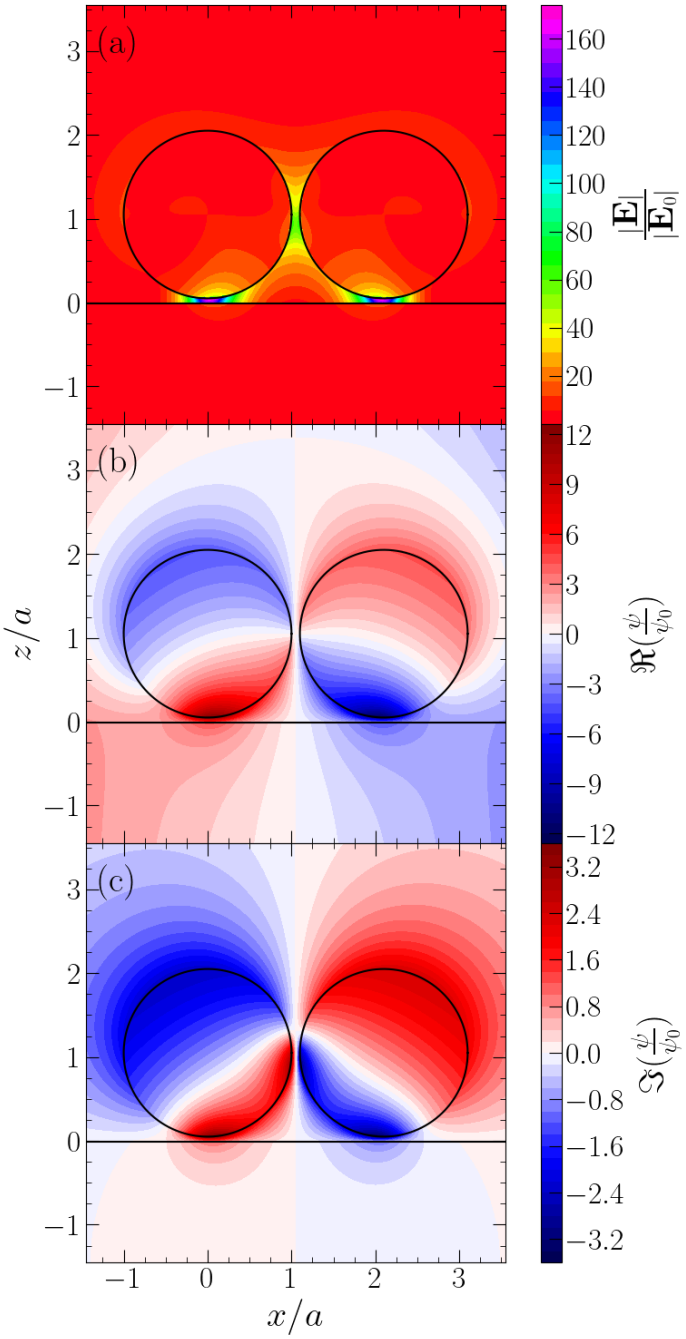


FIGURE 4.11: The (a) field enhancement, the (b) real and (c) imaginary part of the potential enhancement of a supported dimer. The dielectric function of the spheres follows the Drude model with $\hbar\omega_p = 3\,\text{eV}$ and $\hbar\gamma = 0.03\,\text{eV}$, and the dielectric function of the substrate is $\varepsilon_- = 10$. The spheres are separated by a distance of $d = 0.1a$ and placed a height $h = 0.05a$ above the substrate, where a is the radius of the spheres. The substrate is located at $z = 0$, and the centre of the left sphere is located at $x = 0$. The centres of the two spheres lie parallel to the x-axis as per the electric field from the incident plane wave. The ambient medium is vacuum and the incident wave energy is $\hbar\omega = 1.35\,\text{eV}$. The multipole order used is $L = 30$.

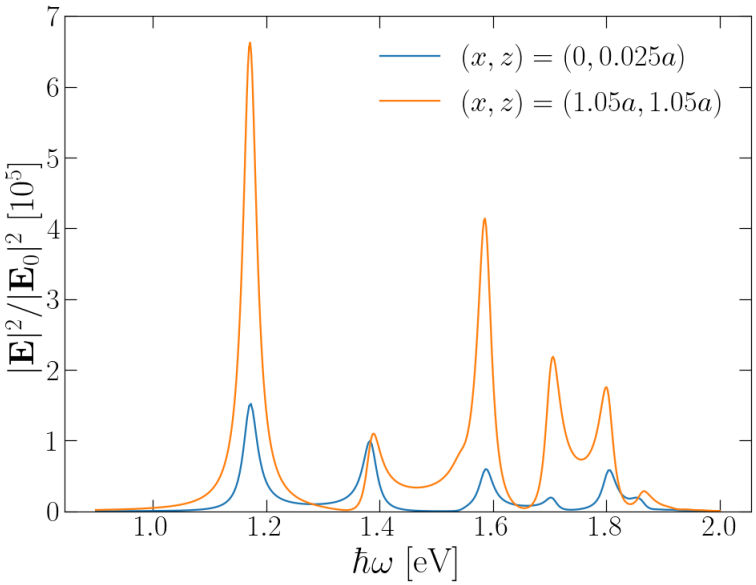


FIGURE 4.12: The squared field enhancement of a supported dimer scanned for incident wave energies. The orange line is evaluated at the middle point between the centres of the two spheres and the blue line is evaluated at the middle point between one sphere and the substrate. The dielectric function of the spheres follows the Drude model with $\hbar\omega_p = 3\,\text{eV}$ and $\hbar\gamma = 0.03\,\text{eV}$, and the dielectric function of the substrate is $\varepsilon_- = 10$. The spheres are separated by a distance $d = 0.1a$ and placed a height $h = 0.05a$ above the substrate, where a is the radius of the spheres. The substrate is located at $z = 0$, and the centre of the left sphere is located at $x = 0$. The centres of the two spheres lie parallel to the x-axis as per the electric field from the incident plane wave. The ambient medium is vacuum, and the multipole order used is $L = 30$.

symmetry with respect to the plane $x = 1.05a$, where the distance to each sphere is equal.

When scanning the electric field enhancements squared, between the two spheres and between a sphere and the substrate, for an energy spectre as in Fig. 4.12, not many incident wave energies result in greater substrate induced field enhancement than the enhancement between the spheres. This was pointed out in Ref. [2], where the figure is recreated from Fig. 6. A such scanning allows for more precise verification, as a contour plot is not as easy to compare precisely. The two figures are indistinguishable. The field enhancements can also be used to further determine the origin of the resonances in the dipole moments. As previously, the most left peak originates from the inter-particle interaction, and the second left peak originates from the substrate interaction. The third and fourth peak appear to originate from the inter-particle interaction, but the last one is still difficult to determine. Even though the field enhancement between the spheres is greatest, it may still be a result of the substrate interaction, and vice versa.

The real and imaginary parts of the potential enhancement from Eqs. (2.2.10), (2.2.22)–(2.2.23), for the same system as in Fig. 4.11(a), are shown in Figs. 4.11(b)–(c), respectively. The potential is continuous everywhere as expected from the boundary conditions in Eqs. (2.2.33)–(2.2.35). Moreover, the the potential is completely anti-symmetric with respect to the separating plane, $x = 1.05a$, where the zero potential is defined. The anti-symmetric potential is a consequence of the incident electric field being parallel to the axis through the centres of the spheres. For the incident electric field parallel to the other two Cartesian axes, the potential enhancements become completely symmetric as illustrated in Fig. 4.13. When the incident electric field is parallel to the z-axis in

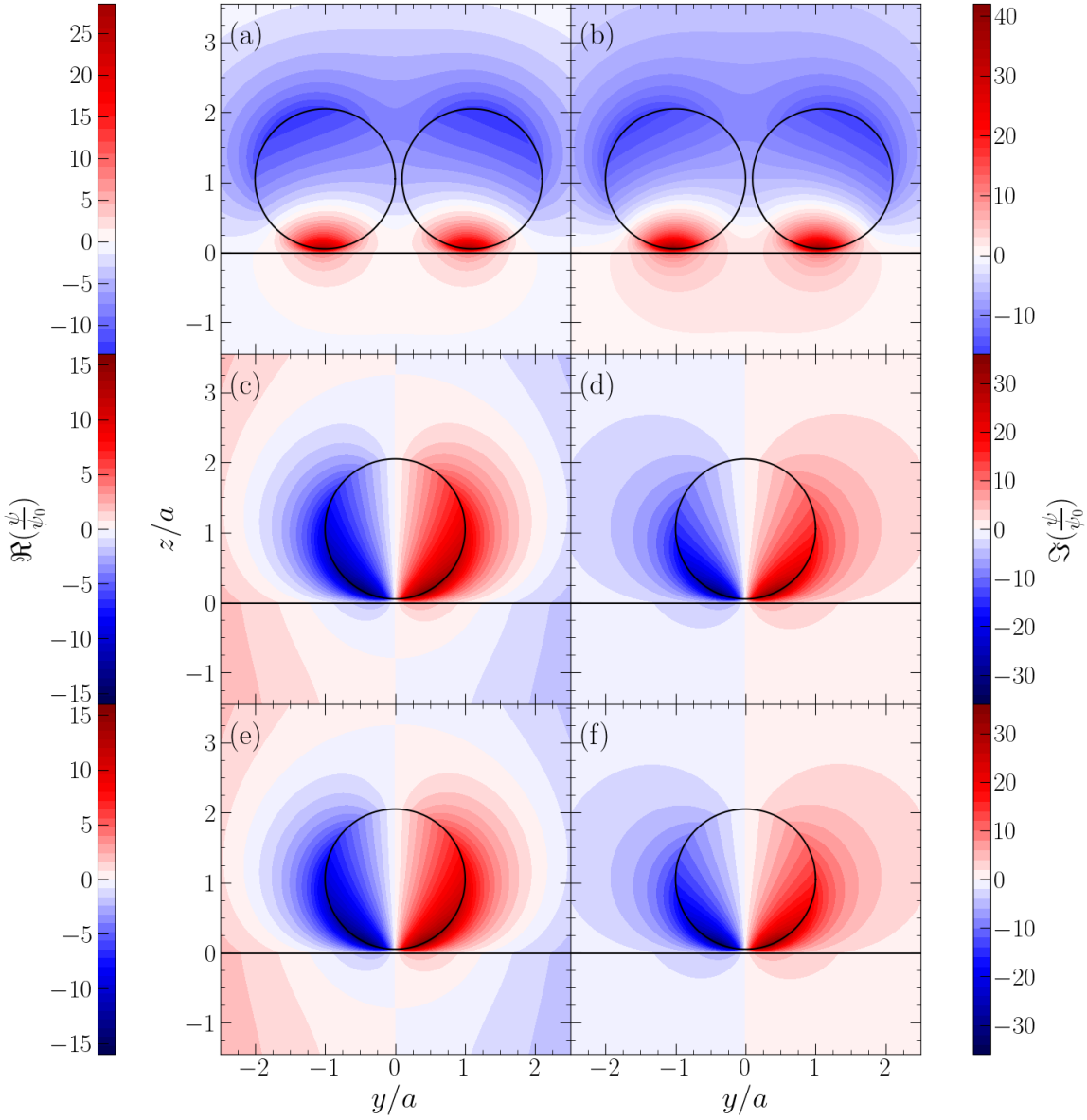


FIGURE 4.13: The potential enhancements for a supported dimer positioned on (a)–(b) the y-axis and (c)–(f) the x-axis. The real parts of the potential enhancements are presented in (a), (c) and (e), while the imaginary parts are presented in (b), (d), and (f). The cross sections in (a)–(d) are positioned at $x = 0$, and the cross sections in (e)–(f) are positioned at $x = 2.1a$. The incident electric field is parallel to the z-axis in (a)–(b) and to the y-axis in (c)–(f). The energy of the incident wave in (a)–(b) is $\hbar\omega = 1.43\,\text{eV}$ and in (c)–(f) $\hbar\omega = 1.58\,\text{eV}$. The dielectric function of the spheres follows the Drude model with $\hbar\omega_p = 3\,\text{eV}$ and $\hbar\gamma = 0.03\,\text{eV}$, and the dielectric function of the substrate is $\varepsilon_- = 10$. The spheres are separated by a distance $d = 0.1a$ and placed a height $h = 0.05a$ above the substrate, where a is the radius of the spheres. The ambient medium is vacuum, and the multipole order used is $L = 30$. In (a)–(b) the centres of the spheres are located at $(x, y, z) = (0, -1.05a, 1.05a)$ and $(x, y, z) = (0, 1.05a, 1.05a)$, and in (c)–(f) at $(x, y, z) = (0, 0, 1.05a)$ and $(x, y, z) = (0, 2.1a, 1.05a)$.

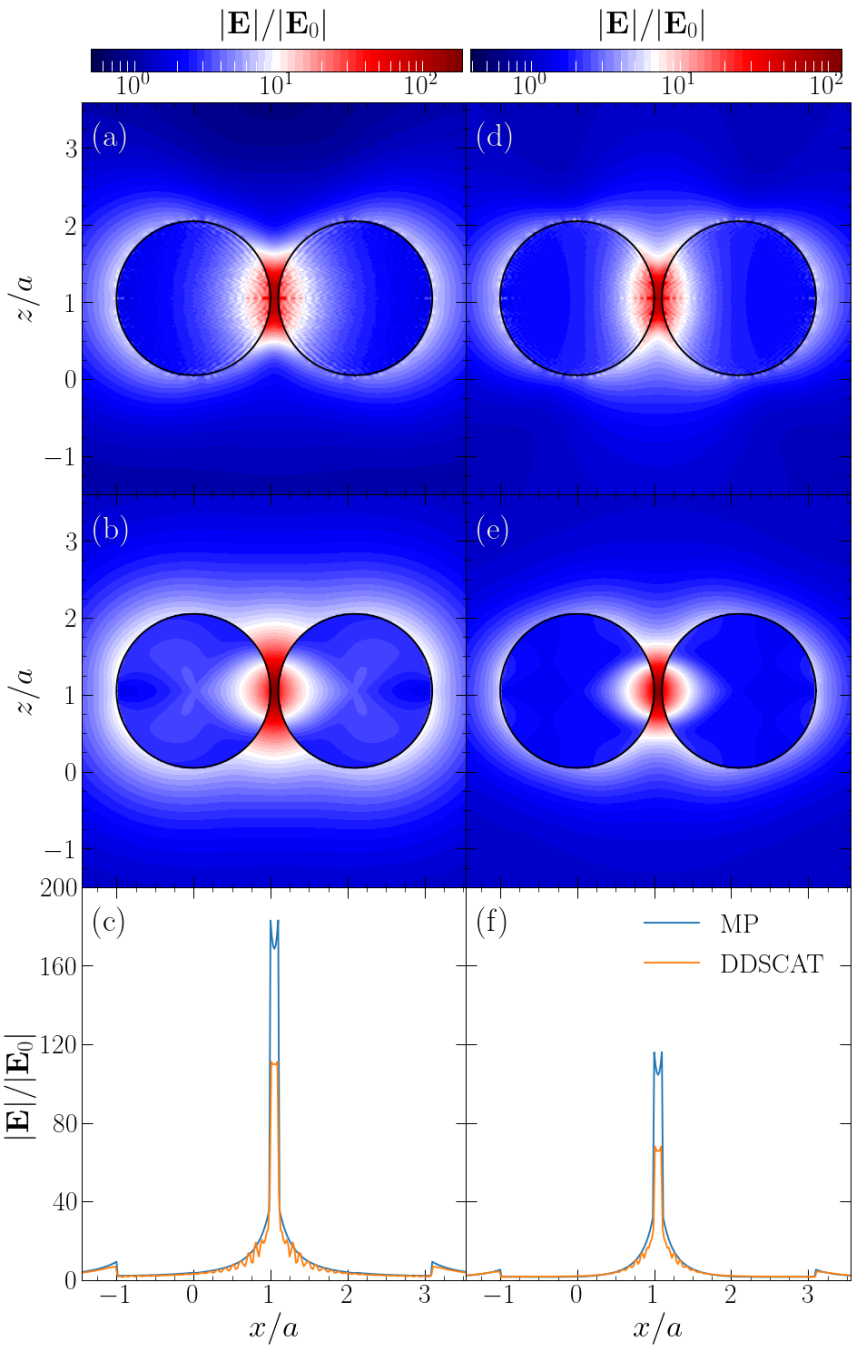


FIGURE 4.14: The field enhancement of a freestanding Ag dimer in vacuum, $\varepsilon_+ = \varepsilon_- = 1$. The dielectric function of the spheres are provided from the Sopra database, and the spheres are separated by a distance $d = 0.1a$, with a as the radius. The incident wave energy for (a)–(c) is $\hbar\omega = 2.96\,\text{eV}$ and for (d)–(f) is $\hbar\omega = 3.2\,\text{eV}$. The left colour bar corresponds to both the contour plots (a) and (b), and likewise for the right colour bar and the contour plots (d) and (e). The results in (a) and (d) are obtained from DDSCAT, while the results in (b) and (e) are obtained from the multipole expansion. For DDSCAT, each sphere is discretised by 523 305 point dipoles and the spheres are separated by 5 point dipoles. The centre of the left sphere is located at $x = 1.05a$, and the axis through the centre of the spheres is parallel to the x-axis. Similarly, the electric field of the incident plane wave is also parallel to the x-axis, and in the results obtained from DDSCAT the incident wave vector is parallel to the z-axis. Lastly, the field enhancements in (c) and (f) are scanned along the axis $(x, z) = (0, 1.09a)$.

Figs. 4.13(a)–(b), the potential enhancements are mirrored by the separation plane $y = 0$. The potential enhancements are largest on the side of the spheres facing the substrate as in Figs. 4.11(b)–(c), due to the resonances originating mainly from the substrate interactions. However, the hot spots closest to the substrate are not attracted to each other in the same manner. The potential enhancements in Figs. 4.13(c)–(d) are identical to the potential enhancements in Figs. 4.13(e)–(f), as the potential is symmetric when mirrored by the plane $x = 1.05a$, which is placed equally far away from the two spheres. The hot spots are now positioned on the vertical sides of the spheres, but shifted a bit towards the substrate.

The electric field enhancement for an Ag dimer in vacuum is presented in Fig. 4.14. As opposed to the dimer in Fig. 4.11, there is no substrate present due to the limitations of DDSCAT. The electric field enhancements are calculated for incident energies $\hbar\omega = 2.96\,\text{eV}$ in Figs. 4.14(a)–(c) and for $\hbar\omega = 3.2\,\text{eV}$ in Figs. 4.14(d)–(f). The first incident energy corresponds to the global maximum of the dipole moment for the two spheres, while the second energy corresponds to a local minimum. Consequently, the field enhancements in Figs. 4.14(a)–(c) are greater than the ones in Figs. 4.14(d)–(f), especially in the vicinity around the spheres and near the hot spot between the spheres. The incident electric field is parallel to the x-axis as in the near field calculations from Fig. 4.11. Due to the absence of a substrate, the electric field enhancements obtained from the multipole expansion in Figs. 4.14(b) and 4.14(e) are symmetric around the axis through the centres of the two spheres. The near field enhancements obtained from DDSCAT specifies the incident wave vector parallel to the z-axis as well. Hence, the symmetry for the electric field enhancements in Figs. 4.14(a) and 4.14(d) are reduced to a mirror symmetry with respect to the plane $x = 1.05a$ as in Fig. 4.11(a). The break in symmetry is caused by DDSCAT including retardation in the calculations.

The discretisation of the dimer used for the DDSCAT near field calculations in Figs. 4.14(a) and 4.14(d) is displayed in Fig. 3.4. DDSCAT is limited to a constant inter-dipole spacing for each axis. Consequently, the point dipole resolution in the areas with the greatest field enhancements is limited to the overall resolution. The electric field enhancements obtained from DDSCAT are thus not as accurate in the hot spots as the enhancements obtained from the multipole expansion. This is particularly illustrated when scanning through the axis $(y, z) = (0, 1.09a)$. The difference in enhancements increases when approaching the gap between the spheres, and in the gap there is more than a third in relative difference. The numerical noise for DDSCAT is much greater for $\hbar\omega = 2.96\,\text{eV}$ than for $\hbar\omega = 3.2\,\text{eV}$. The choice of the axis $(y, z) = (0, 1.09a)$ over the axis through the centres of the spheres was to reduce this numerical noise considerably. Overall, the two methods agree well except for in the hot spot.

4.2 Infinite systems

So far the systems consisting of a few spheres have been studied. In this section, systems made of an infinite number of supported particles will be considered. Such systems can come in many forms, for instance, as a random or regular (periodic) array of spheres. Here the focus will mostly be on systems consisting of a regular array of spheres, both square and hexagonal lattices. To quality check the implementations of App. A.1 regarding systems of infinite lattices, the results from Figs. 2 and 4–5 from Ref. [3] are reproduced in Figs. 4.15 and 4.18. Moreover, the validity of the quadrupole order of inter-particle interaction is studied for various surface densities.

4.2.1 Dimensionless dipole moments

The dimensionless dipole moments from Fig. 2 in Ref. [3] are recreated as Fig. 4.15, and the results are indistinguishable, which testifies to the correctness of the implementation. The material of the

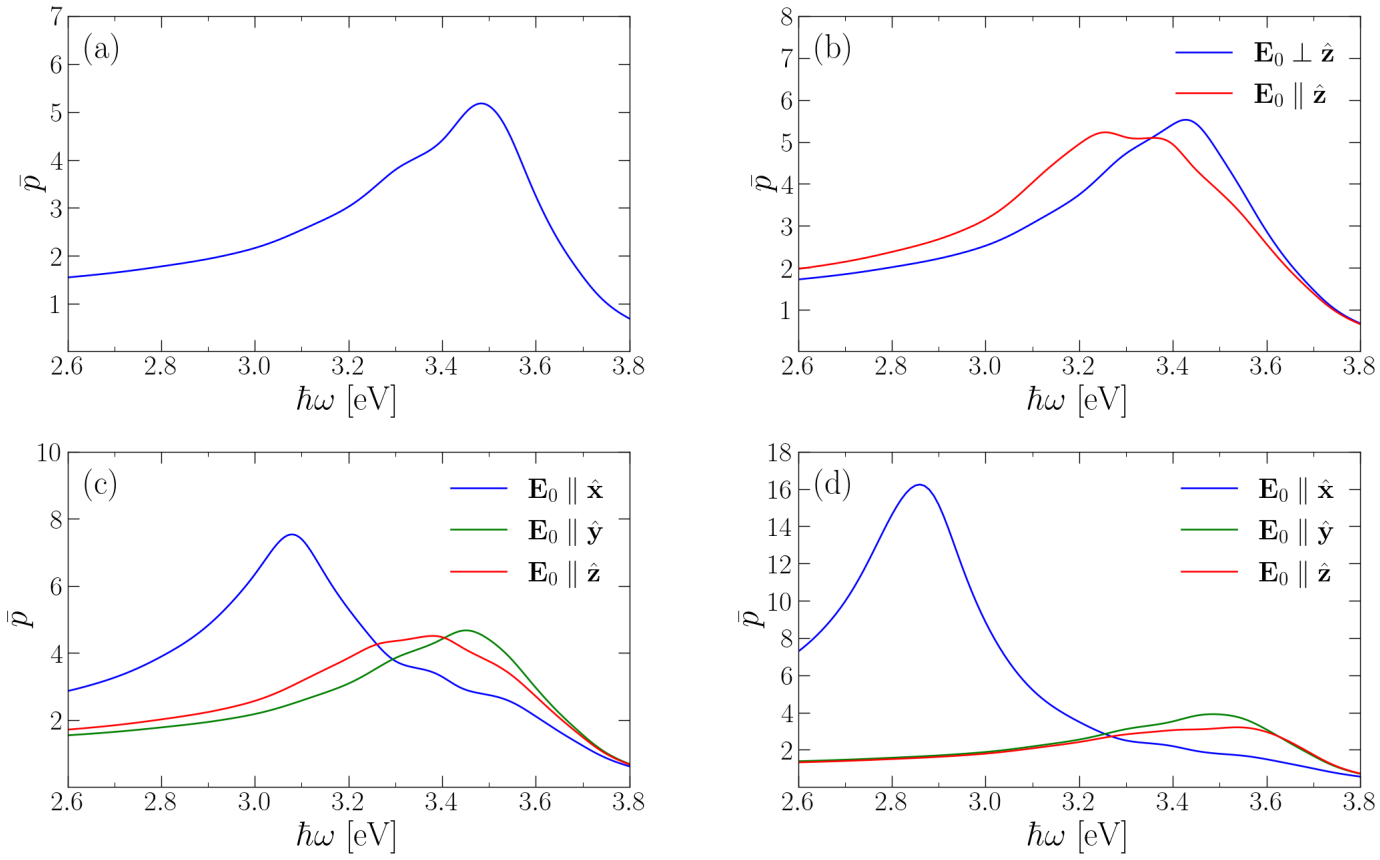


FIGURE 4.15: The dimensionless dipole moment, $\bar{p}(\omega)$, as a function of incident wave energies for (a) a single freestanding nanosphere, (b) a single supported nanosphere, (c) a supported dimer and (d) an infinite lattice of nanospheres. All the spheres are made of Ag and have the same radius a. The supported spheres in (b)–(d) are placed a height $h = 0.01a$ above an Al_2O_3 substrate with dielectric function $\varepsilon_- = 2.76$. The dimer in (c) is separated by a distance $d = 0.2a$, and the lattice in (d) has lattice constants $b_x = 2.2a$ and $b_y = 4b_x$. The dielectric function for Ag is provided from the Sopra database, and the ambient medium is vacuum. The multipole order used is $L = 30$. The lattice sites within an interaction distance $R_{\text{int}} = 30a$ are included in the calculations.

spheres is Ag, and the dielectric function is provided by the Sopra database [6]. The experimental data for the dielectric function does not result in as smooth curves as the Drude model did in Figs. 4.1–4.3. The isolated sphere in Fig. 4.15(a) reaches the Mie resonance at $\hbar = 3.5\,\text{eV}$. Supporting the sphere with an Al_2O_3 substrate induces a slightly greater red shift in the dipole moment when the incident field is parallel to the z-axis, than in Fig. 4.1(b) where a substrate with a dielectric function $\varepsilon_- = 2$ was used. The greater red shift agrees with the larger dielectric function of the substrate at $\varepsilon_- \approx 2.76$ as illustrated in Fig. 4.16. The sphere is placed closer to the substrate as well. The deformation of the resonance peak at $\hbar\omega \approx 3.3\,\text{eV}$ can be explained from the distortion of the dielectric function of Ag provided by the Sopra database in Fig. 4.16. For comparison, the Drude model for Ag, with $\hbar\omega_p = 9.17\,\text{eV}$ and $\hbar\gamma = 0.018\,\text{eV}$, and the Drude model from Secs. 4.1.1-4.1.3 are included. Clearly, the Drude model for Ag with the given parameters is not very accurate.

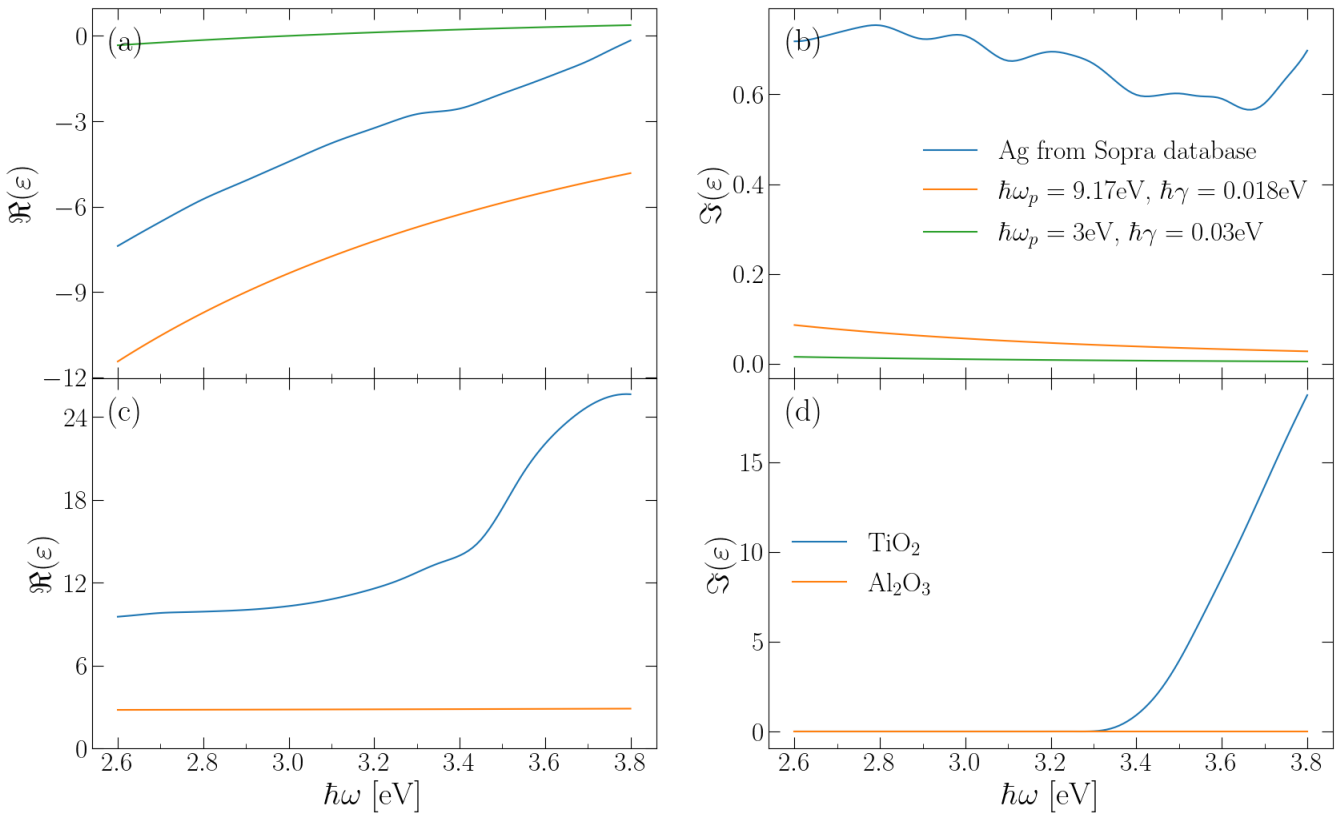


FIGURE 4.16: The dielectric functions for (a) and (b) Ag provided by Sopra and by the Drude model, respectively, and for (c) and (d) Al_2O_3 and TiO_2 provided by Sopra, respectively. The parameters for the Ag Drude model used are $\hbar\omega_p = 9.01\,\text{eV}$ and $\hbar\gamma = 0.018\,\text{eV}$. The other Drude model used has $\omega_p = 3\,\text{eV}$ and $\hbar\gamma = 0.03\,\text{eV}$. The real values are displayed in (a) and (c), and the complex ones in (b) and (d). The dielectric functions are plotted for incident wave energies.

The dimer in Fig. 4.15(c) induces a red shift of the resonance peak, for the incident field parallel to the x-axis, a bit weaker than the red shift in Figs. 4.2(b) and 4.3(b), which agrees with the longer separation distance $d = 0.2a$ between the spheres. Moreover, there is only one resonance peak present. The high damping caused by the imaginary part of the dielectric function for Ag in Fig. 4.16(b) may suppress the resonances such that only the strongest one makes it through. The large imaginary part of the dielectric function is also the reason for the lower amplitudes observed in Fig. 4.15. The amplitudes for the dipole moments with incident electric field parallel to the other axes decrease in the transition from a single sphere to a dimer as for Figs. 4.1–4.3. Again, the presence of another sphere appears to disturb the enhancements when the incident field is orthogonal to the axis through the centres of the spheres.

The lattice in Fig. 4.15(d) has the same separation distance between the spheres along the x-axis as for the dimer in Fig. 4.15(c), $b_x = 2.2a$. Along the y-axis on the other hand, the inter-particle distance, from centre to centre, is four times the one along the x-axis, $b_y = 4b_x$. As expected, the dipole moment for incident electric field parallel to the x-axis has a very pronounced red shifted

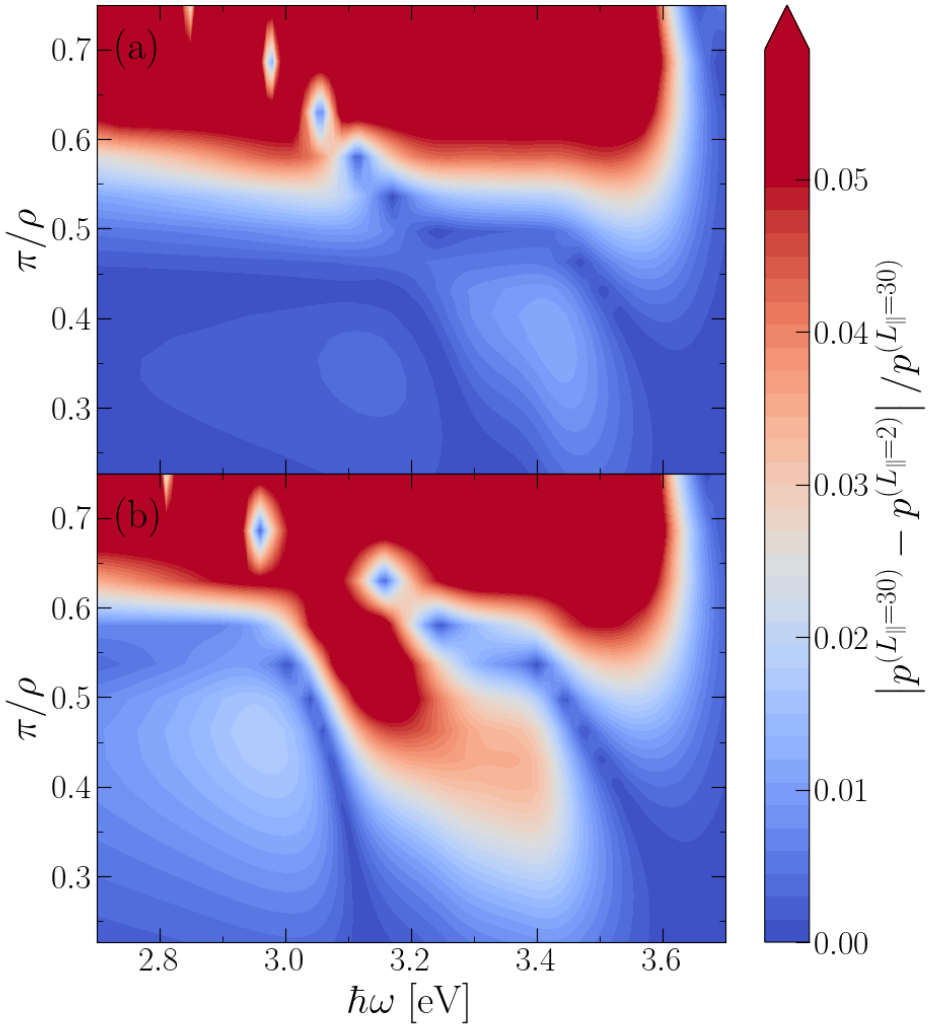


FIGURE 4.17: The relative difference in dimensionless dipole moment between multipole orders in parallel direction $L_\parallel = 2$ and $L_\parallel = 30$. The dimensionless dipole moments are computed for a supported hexagonal Ag lattice in vacuum. The Ag spheres are placed a height $h = 0.01a$ above a substrate made of (a) Al_2O_3 and (b) TiO_2, where a is the radius of the spheres. The differences are plotted for the incident wave energies along the first axis and for the surface density π/ρ along the second axis, where ρ is the area per sphere in units of a^2. The contour plot presents all the relative differences above 5% as the same colour. The dielectric functions of the spheres and the substrates are provided by the Sopra database. Only the lattice sites within an interaction distance $R_{\text{int}} = 30$ from the centre site are included in the calculations. The orthogonal multipole order in all cases is $L_\perp = 30$. The incident electric field is parallel to one of the lattice vectors.

resonance peak of larger amplitude. The incident electric field induces a collective resonance along the arrays of spheres. Along the y-axis on the other hand, the spheres are separated sufficiently to not change the dipole moment much relative to the result presented in Fig. 4.15(c). The dipole

moment for the incident field parallel to the z-axis is flattened more. The Coulomb forces appear to be more restoring when the dipole moments of the spheres align in series with the image dipole moments.

The GranFilm software is limited to quadrupole order in the multipole expansion regarding interaction with neighbouring spheres, $L_{\parallel} = 2$. However, for the interaction with the substrate, the multipole expansion can be truncated at an arbitrary order $L_{\perp}$ similar to what is done for the methodology presented in this thesis. To gain more insight in which surface density the quadrupole approximation in the parallel multipole expansion is valid for, the relative difference in dipole moment from one calculated with a high multipole order, $L_{\parallel} = 30$, is presented in Fig. 4.18 for a range of surface densities. The multipole order regarding interaction with the substrate is kept at $L_{\perp} = 30$ in both cases. The dipole moments are calculated for a hexagonal Ag lattice with lattice vectors $\mathbf{b}_x$ and $\mathbf{b}_y$. The vectors are of equal length, b, and the angle between them is 60°. The area per sphere thus becomes $\rho = |\mathbf{b}_x \times \mathbf{b}_y| = \sqrt{3}b^2/2$. The surface density is then obtained from dividing the area of one sphere by the area per sphere, and in units of radii the area of a sphere is π.

Such a hexagonal lattice of Ag spheres is placed a height $h = 0.01a$ above an Al_2O_3 substrate in Fig. 4.18(a) and a TiO_2 substrate in Fig. 4.18(b). The dielectric functions of the substrates are shown in Fig. 4.16(c) and (d), where the dielectric function for TiO_2 is much greater than for Al_2O_3. Consequently, the surface density allowed before reaching a relative difference of 5% is higher for an Al_2O_3 substrate than a TiO_2 substrate. For Al_2O_3 the surface density can be at 55% and still yield decent results. For TiO_2 on the other hand, the relative difference will be much higher around the resonance. The surface density limit for TiO_2 is at 45%, and still the relative difference around the resonance will be near 5%. The colour bar used presents all relative differences greater than 5% as the same colour in order to gain better resolution in the interval of interest, $[0\%, 5\%]$. The red shifts of the resonance peaks are also visualised as the inter-particle distance decreases. The relative difference for TiO_2 around the resonance decreases evenly as the surface density decreases. The case is different for Al_2O_3 around $\hbar\omega \approx 3.2\,\text{eV}$. Here the two dipole moments intercept and give a somewhat misrepresenting image. The low relative differences from the interceptions are especially unrepresentative for the surface densities above 60%.

4.2.2 Reflectivities

The reflectivity from Eq. (2.2.94) is presented for an Ag lattice in Fig. 4.18, which is recreated from Figs. 4–5 in Ref. [3]. The blue lines are the reflectivities for s-polarised light based on the amplitudes from Eq. (2.2.103), and the red lines are the reflectivities for p-polarised light based on the amplitudes from Eq. (2.2.105). The black line is where the reflectivities for s- and p-polarised light coincide. The lattice is placed a height $h = 0.01a$ above an Al_2O_3 substrate, but the lattice structure in Figs. 4.18(a)–(b) differs from the structure in Figs. 4.18(c)–(d). In Figs. 4.18(a)–(b) the lattice has a square structure with inter-particle distance $b_x = b_y = 2.2a$. As expected, the reflectivities for the s- and p-polarised light coincide for the normal incidence in Fig. 4.18(a). Moreover, the reflectivity for s-polarised light is greater than for p-polarised light for all incident wave energies in Fig. 4.18(b), where the incidence is $\theta = 45°$. The peak of the s-polarised light is also slightly more red shifted than for the p-polarised light. The electric field from the s-polarised light is parallel to the lattice vector $\mathbf{b}_y$ and will thus be able to induce collective resonances along the lattice vector. In contrast, the electric field from the p-polarised light has only a component with $\sqrt{2}/2$ of the one from the s-polarised light along the surface dividing plane. For both directions of incidence, the peaks of the reflectivities occur approximately at the resonance energy for the dipole moment with incident electric field along the x-axis in Fig. 4.15(d) as anticipated.

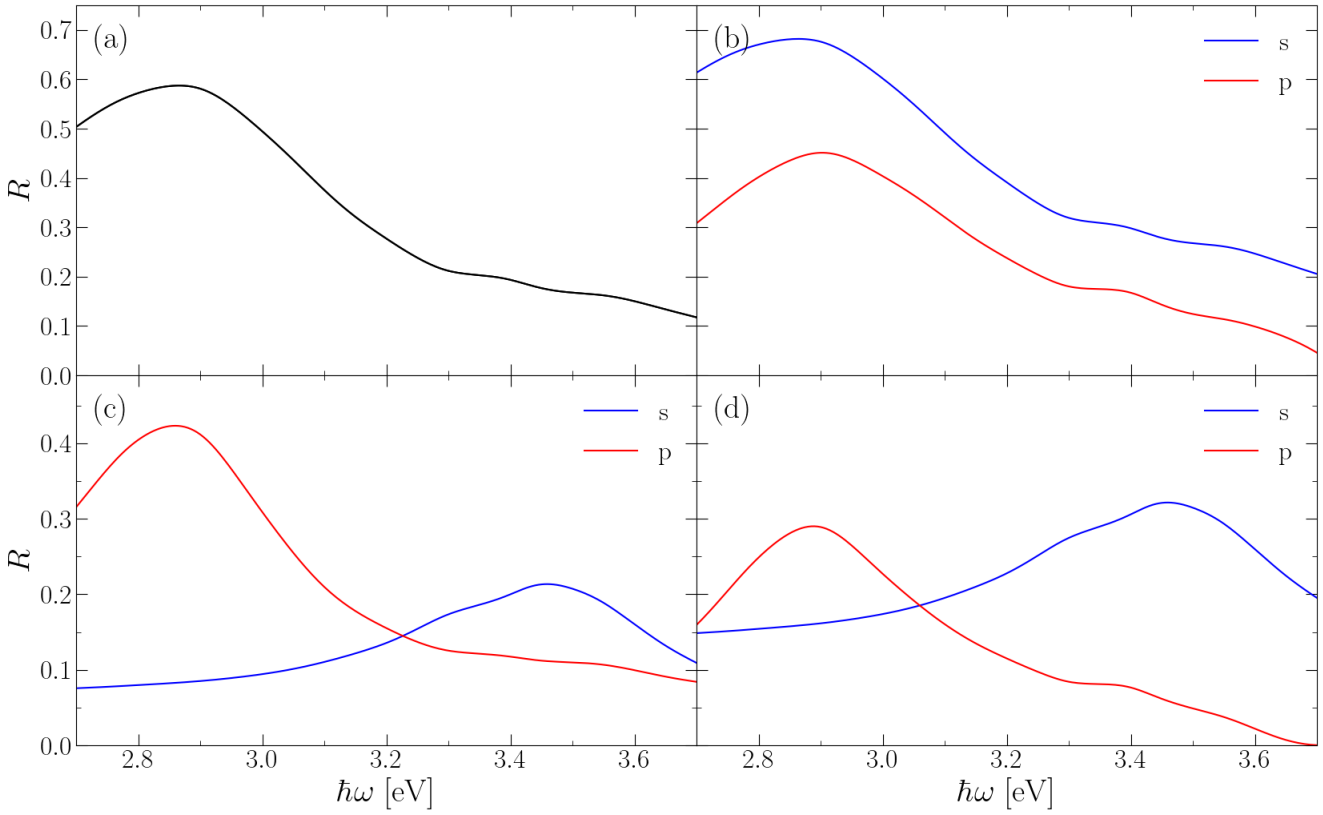


FIGURE 4.18: The reflectivities as functions of incident wave energies for supported Ag nanospheres in (a) and (b) a square lattice with lattice constants $b_x = b_y = 2.2a$ and (c) and (d) a rectangular lattice with lattice constants $b_x = 2.2a$ and $b_y = 2b_x$. The dielectric function of Ag is obtained from the Sopra database. The spheres are placed a height $h = 0.01a$ above a Al_2O_3 substrate with dielectric function $\varepsilon_- = 2.76$, with $a = 10\,\text{nm}$ as the radius of the spheres. The angles of the incident wave vector are for (a) and (c) $(\theta, \phi) = (0°, 0°)$, and for (b) and (d) $(\theta, \phi) = (45°, 0°)$, where s-polarised light is defined to be polarised along the y-axis. The ambient medium is vacuum, and the multipole order used is $L = 30$. The lattice sites within an interaction distance $R_{\text{int}} = 30a$ are included in the calculations.

The lattice structure in Figs. 4.18(c)–(d) is rectangular with $b_x = 2.2a$ and $b_y = 2b_x$. The reflectivities no longer coincide for the normal incidence in Fig. 4.18(c), as the four fold symmetry of the lattice is reduced to two fold. The shape of the reflectivity is still quite similar for the p-polarised light, but the amplitude is noticeably lower with the lower surface density. For the s-polarised light, the shape of the reflectivity resembles more of the dipole moment with the incident electric field parallel to the y-axis in Fig. 4.15(d). The resemblance indicates the inter-particle distance is sufficiently large enough for dampening the interaction between the spheres along the y-axis. When changing the incidence to $\theta = 45°$, the reflectivity increases for the s-polarised light and decreases for the p-polarised light. For the p-polarised light, the component of the incident electric field along the densely packed x-axis is decreased by a portion $1 - \sqrt{2}/2$. For the s-polarised light, the increase in reflectivity is caused by the general increase from the substrate when increasing the

polar angle of the incident wave vector, θ. The results from the paper [3] differ slightly from the ones obtained in Fig. 4.18, where the reflectivities in Fig. 4.18 are generally higher. The paper used another approach for determining the reflectivities than through the susceptibilities in Eqs. (2.2.98) and (2.2.97) proposed by D. Bedeaux and J. Vlieger [1]. Instead, Ref. [3] based the reflectivities on the effective displacement field in Eq. (2.2.92).

4.3 The Discrete Dipole Approximation Method

In the last section of this chapter, the results obtained from the software DDSCAT is tested against two analytical solutions. The first is the Mie theory for the absorption efficiency factor, and the second is the reflectivity of a finite, thin film in vacuum. Lastly, the reflectivities for various unsupported square Ag lattices obtained from DDSCAT are compared to the ones obtained from the multipole expansion implemented here and grom GranFilm.

4.3.1 Verification

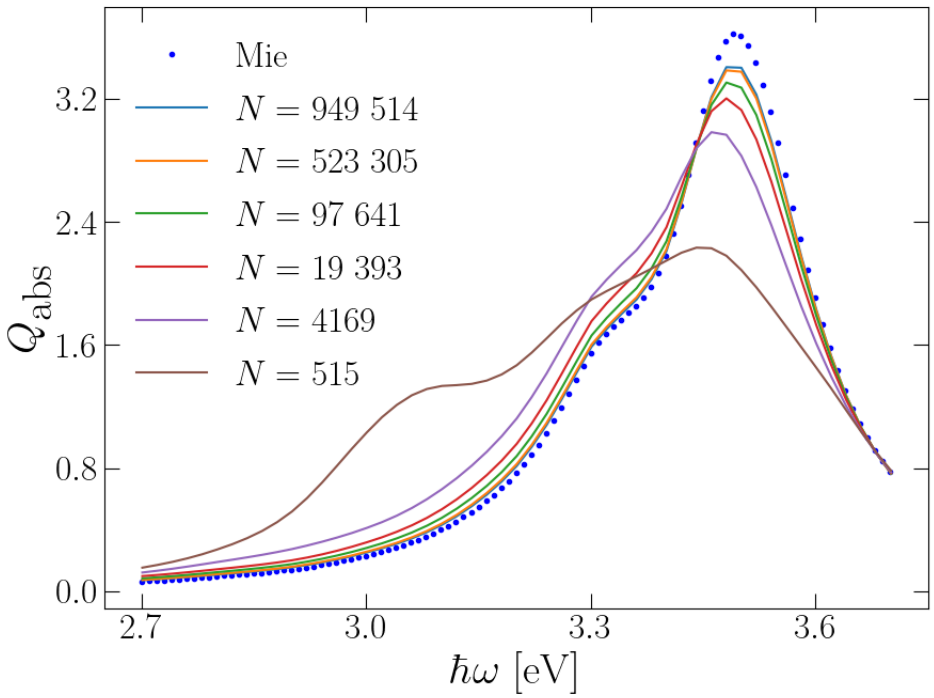


Figure 4.19: The absorption efficiency factor for an Ag sphere with radius $a = 10\,\text{nm}$ in vacuum plotted for incident wave energies. The blue dots are the Mie result, and the other lines are obtained from DDSCAT calculations with various number of point dipoles.

To verify the results from DDSCAT, the absorption efficiency factor, Q_{abs}, in Eqs. (2.4.18) and (2.4.23) for an Ag sphere in vacuum is compared to the exact Mie theory result in Eqs. (2.4.46) and (2.4.20). The discretisation of the sphere with radius $a = 10\,\text{nm}$ ranges from $N = 515$ to $N = 949\,514$ point dipoles and is visualised in Fig. 3.2 for $N = 4169$ and $N = 523\,305$. The efficiency factor calculated on the basis of $N = 515$ point dipoles is quite far away from the Mie result.

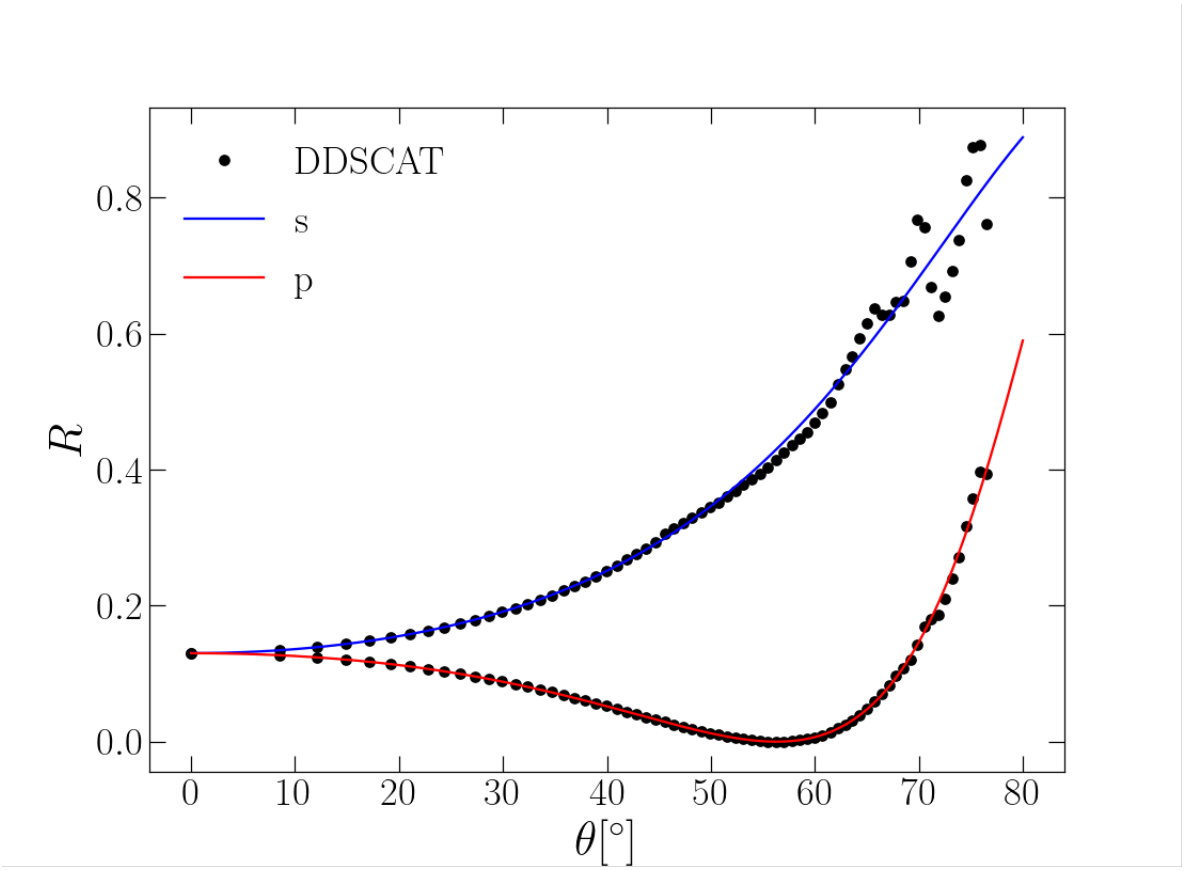


FIGURE 4.20: The reflectivity of a thin, finite film of thickness $d = 100\,\text{nm}$ in vacuum plotted for the angle of incidence. The incident wave energy is $\hbar\omega = 2.48\,\text{eV}$ and the refractive index of the film is $n = 1.5 + 0.02\text{i}$. In the DDSCAT calculations, the film is discretised by 20 point dipoles in thickness. The interaction cutoff parameter used is $\gamma = 0.01$.

At $N = 4169$ point dipoles, the shape of the curve is much more similar to the Mie result, but still somewhat far off. As the discretisation resolution increases, the results converge towards the analytic theory. However, the convergence slows down as the number of point dipole increases, and there is barely any difference between $N = 523\,305$ and $N = 949\,514$ point dipoles, except for almost twice as long simulation time for the latter case compared to the former. For those dipole densities, the absorption efficiency factor is very close to the Mie result except for at the resonance. When the field enhancements are at their greatest, the dipole densities are insufficient for an accurate result. Outside the resonance, the number of point dipoles $N = 523\,305$ suffices.

To verify the reflectivity obtained from DDSCAT in Eq. (2.4.30), the reflectivity of a thin film in vacuum is compared to the analytic reflectivity from Eq. (2.4.50) in Fig. 4.20. The reflectivities are calculated for incident angles ranging from $\theta = 0°$ to almost $80°$. The unit cell of the film is an array of 20 point dipoles along the z-axis. The point dipoles in the unit cell form the thickness of the film, $d = 100\,\text{nm}$. The unit cell is then repeated periodically in the xy-plane, with the same inter-dipole spacing as within the unit cell, and the interaction parameter $\gamma = 0.01$ is used. The results start out accurate, but as θ increases, the results oscillate around the analytic result. The greater the angle of incidence, the more the results deviate from the solution. As the numerical solver used is iterative, the computation time depends on how close the solution is to the previous result. The rapid oscillation causes large variations and thus long computation times, which is why

the data points stop before $\theta = 80°$. The reflectivity for s-polarised light deviates more and earlier than for p-polarised light. However, for both polarisations, the results obtained from DDSCAT are still rather accurate at $\theta = 45°$.

4.3.2 DDSCAT compared to multipole expansion and GranFilm

The reflectivities obtained for a square Ag lattice with the multipole expansion are compared to the corresponding results obtained from DDSCAT in Figs. 4.21(a)–(c) under normal incidence and in Figs. 4.22(a)–(c) with an angle of incidence $\theta = 45°$. The discretisation of the spheres is demonstrated in Fig. 3.2. Due to the limitations of DDSCAT, there is no substrate present. Apart from the substrate, the systems assumed to produce the results in Figs. 4.21(a) and 4.22(a) are identical to those assumed in obtaining the results in Figs. 4.18(a)–(b), respectively. The reflectivities without substrate are generally lower, but at the resonance the differences are not as noticeable. The reflectivity caused by the substrate is relatively small compared to the reflectivity caused by the spheres at resonance. Nevertheless, the substrate induces a stronger resonance and a slightly larger red shift of the resonance energy positions. The computation time for DDSCAT was two weeks for the 11 data points for the reflectivities in Fig. 4.22(a), with $N = 523\,305$ and $\gamma = 0.01$ on an Intel® Core™ i7-8700K processor. This was the longest computation time for the DDSCAT simulations, as they depend on the periodicity of the lattice and the direction of the incident plane wave. The reflectivities obtained by the multipole method in the same figure took a half hour on the same processor for $L = 30$ and $R_{\text{int}} = 30a$. The interaction cutoff parameter γ could probably have been a bit higher without much loss of accuracy.

The lattice spacing in Figs. 4.21(a)–(c) and 4.22(a)–(c) are $b_x = b_y = 2.2a$, $3a$ and $4a$. The data points obtained from DDSCAT distant from the resonances fit very well with the results from the multipole expansion. Around the resonance on the other hand, DDSCAT struggles to achieve high enough reflectivity as it did with the absorption efficiency factor in Fig. 4.19. As the discretisation of the sphere at $N = 523\,305$ point dipoles is not sufficient to obtain a highly accurate result, the reflectivities obtained from a discretisation of $N = 4169$ point dipoles are included as well to indicate the degree of convergence. The greater the difference between the reflectivities obtained for the two discretisations, the further away are the results from convergence. As the lattice spacing increases, the resonance weakens and becomes narrower in the energy spectrum. Consequently, the data points from DDSCAT fit better with the results from the multipole expansion. Furthermore, the data points fit best for p-polarised light at an incidence $\theta = 45°$, second best for normal incidence and last comes the s-polarised light at incidence $\theta = 45°$. The deviation being greater for s-polarised light than p-polarised light agrees with the evolution in Fig. 4.20. The smaller discrepancy for p-polarised light at incidence $\theta = 45°$ than the discrepancy for normal light indicates the electric field being directed completely along a lattice vector has greater impact on the accuracy than the incidence of the wave vector. The data points from DDSCAT appear to be slightly above the data points obtained from the multipole expansion to the left of the resonance peak and vice versa on the right side. Hence, there appears to be a minuscule red shift of the DDSCAT data points compared to the ones from the multipole expansion.

The reflectivities for a lattice of Ag half spheres in vacuum are presented in Figs. 4.21(d)–(f) and 4.22(d)–(f). The lattice vectors are the same as for Figs. 4.21(a)–(c) and 4.22(a)–(c). The half spheres are oriented with the dome facing upwards from the xy-plane as if they were spheres intercepted by a substrate. The discretisation of the half spheres are illustrated in Fig. 3.3. The reflectivities for the half spheres are red shifted a lot compared to the full spheres. Furthermore, the amplitudes are smaller for the half spheres. As the scattering object is smaller and its shape is truncated at the middle, the resonance may be assumed to be weaker. In Figs. 4.21(e)–(f) and 4.22(e)–(f),

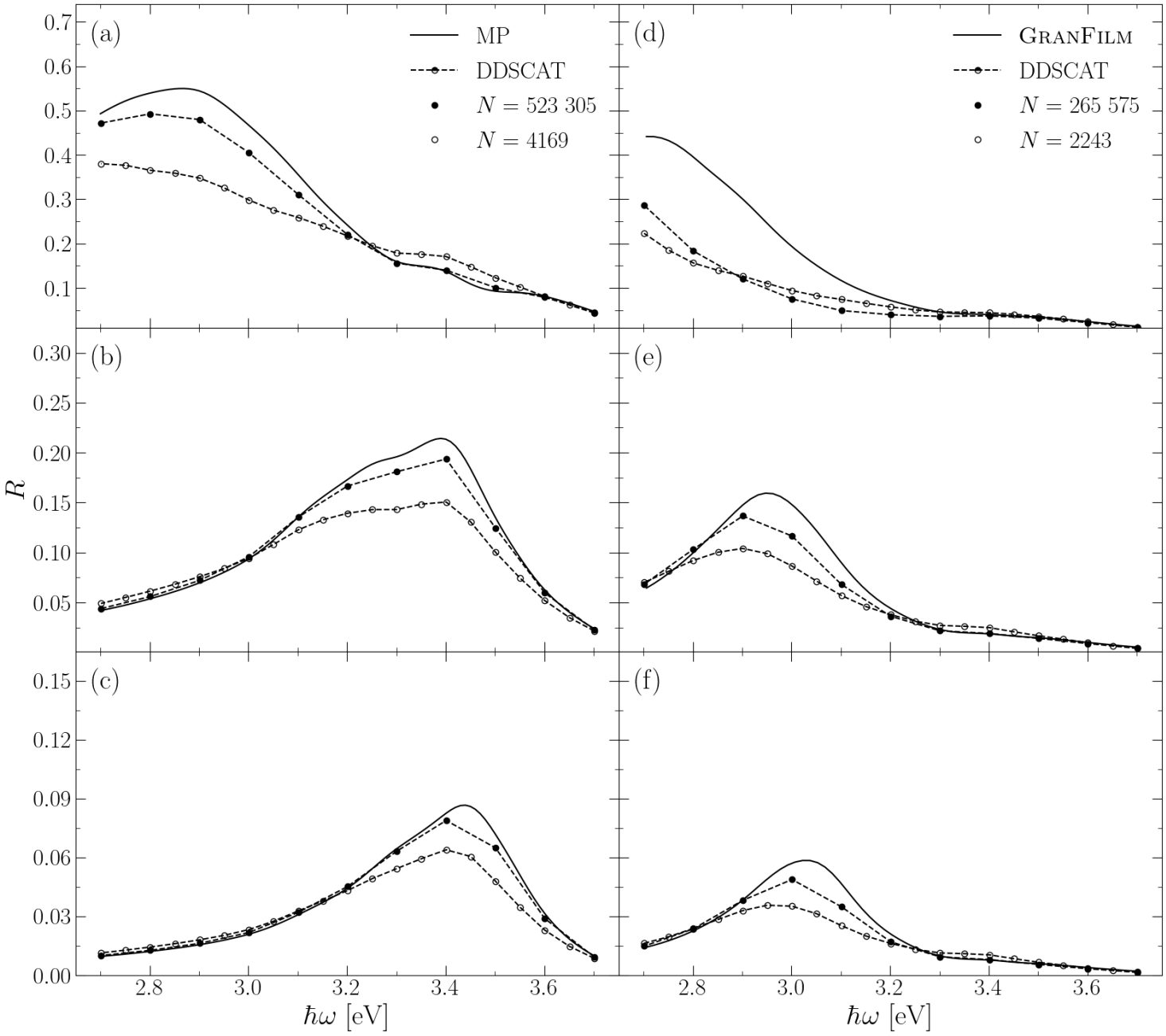


FIGURE 4.21: The reflectivities as a function of incident wave energies, for square lattices of Ag (a)–(c) spheres and (d)–(f) half spheres hovering in vacuum. The radius of the full and half spheres is $a = 10\,\text{nm}$, and the lattice constants are for (a) and (d) $b_x = b_y = 2.2a$, for (b) and (e) $b_x = b_y = 3a$ and for (c) and (f) $b_x = b_y = 4a$. The dielectric function for Ag is provided by the Sopra database. In the multipole calculations, the truncation order is $L = 30$, and the lattice sites included are within the interaction distance $R_{\text{int}} = 30a$. For DDSCAT the cutoff parameter is $\gamma = 0.01$. The GranFilm calculations are done for a substrate with dielectric function $\varepsilon_- = 1.0201$. The truncation in parallel multipole expansion for GranFilm is at quadrupole order, $L_\parallel = 2$, and the truncation in orthogonal multipole expansion is $L_\perp = 32$. The angles of incidence are $(\theta, \phi) = (0°, 0°)$.

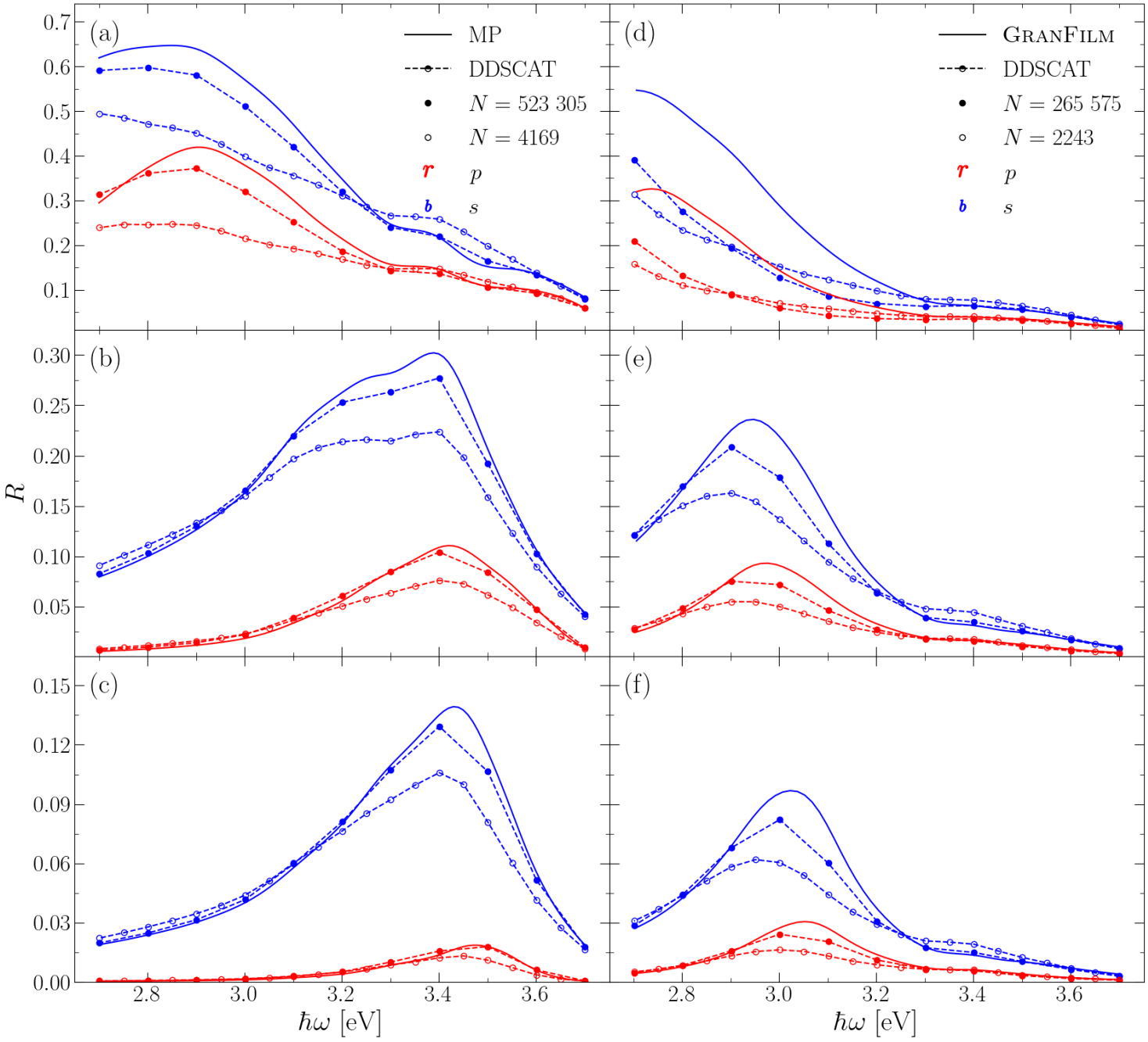


FIGURE 4.22: The reflectivities as a function of incident wave energies, for square lattices of Ag (a)–(c) spheres and (d)–(f) half spheres hovering in vacuum. The radius of the full and half spheres is $a = 10\,\text{nm}$, and the lattice constants are for (a) and (d) $b_x = b_y = 2.2a$, for (b) and (e) $b_x = b_y = 3a$ and for (c) and (f) $b_x = b_y = 4a$. The dielectric function for Ag is provided by the Sopra database. In the multipole calculations, the truncation order is $L = 30$, and the lattice sites included are within the interaction distance $R_{\text{int}} = 30a$. For DDSCAT the cutoff parameter is $\gamma = 0.01$. The GranFilm calculations are done for a substrate with dielectric function $\varepsilon_- = 1.0201$. The truncation in parallel multipole expansion for GranFilm is at quadrupole order, $L_\parallel = 2$, and the truncation in orthogonal multipole expansion is $L_\perp = 32$. The angles of incidence are $(\theta, \phi) = (45°, 0°)$.

the reflectivities obtained from GranFilm [14] behave very similarly to the reflectivities obtained from the multipole method when compared to DDSCAT. For GranFilm, the reflectivities appear slightly blue shifted, and near the resonances the amplitude is greater. The differences in amplitudes are even greater than between DDSCAT and the multipole expansion. The total number of point dipoles is halved, which may result in less accurate resonances even though the inter-dipole spacing is the same. The discretisation of the half spheres in Fig. 3.3 aren't perfectly halved either, as the full spheres consists of an odd number of point dipole planes. The software GranFilm being limited to quadrupole order in interaction with neighbouring islands could be a minuscule factor, but the surface densities for sub plots (e) and (f) are 35% and 20%, respectively. Recalling from Fig. 4.17, the quadrupole approximation in interaction with neighbouring spheres was valid for at least up to 50% surface density with less than one percent relative error, even with an Al_2O_3 substrate. For Figs. 4.21(d) and 4.22(d) on the other hand, the surface density is at 65%, where the quadrupole approximation probably is insufficient to properly account for the interaction between particles. Consequently, the resonances obtained from GranFilm are much more blue shifted compared to DDSCAT. However, outside of the resonance region the data points fit better than anticipated. The general red shift of the resonances obtained with DDSCAT compared to the multipole expansion and GranFilm may be speculated to be due to DDSCAT including retardation in the calculations.

Chapter 5

Conclusion

A code (see App. A.1) is implemented based on a multipole expansion of the scalar potential, for performing simulations of dimensionless dipole moments, electric field enhancements, potential enhancements and reflectivities for s- and p-polarised light. The the code is validated by comparison with multiple references.

The expansion coefficients determined from the multipole expansion for a finite and infinite set of spheres are verified through the comparison of the dimensionless dipole moments computed in Secs. 4.1.1 and 4.2.1 and Refs. [2, 3], respectively. The validity of the results is thus supported by the work of P.A. Letnes, I. Simonsen and D. L. Mills. The observations made include stronger interactions causing stronger red shifts of the energy position of the resonance. This is particularly visualised in Sec. 4.1.2. The largest red shift occurs from the inter-particle interaction, but only when the incident electric field is parallel to a short lattice vector or the axis through the centres of the dimer. For the other Cartesian axes, the potential in the spheres becomes symmetric instead of anti-symmetric, which results in the Coulomb force acting as a restoring force. Hence, the electric field enhancements weaken resulting in a slight blue shift in the resonance energy positions. The substrate interactions on the other hand, induce red shifts for all polar angles of incidence of the electric field, and the largest substrate induced red shift occurs for normal incidence of the electric field.

The near field calculations of the electric field enhancements in Sec. 4.1.4 is verified against the results presented in Ref. [2], which again supports the calculations of the potential enhancements. The potential enhancements are continuous everywhere and anti-symmetric for the incident electric field parallel to the axis through the centres of the dimer, but symmetric for the other two Cartesian axes. The near field calculations for DDSCAT are more characterised by numerical noise, and the deviations from the results obtained from the multipole expansion increase more the closer they are evaluated to the hot spot. The same trend occurs for the verification of the absorption efficiency factor in Sec. 4.3.1. Near the resonance, and thus largest field enhancements, the discretisation of the sphere at $N = 523\,305$ point dipoles is insufficient for great accuracy.

The reflectivities are only halfway verified in Sec. 4.2.2 when compared to the results from Ref. [3], as another approach is used there. Consequently, the reflectivities in Sec. 4.2.2 are somewhat higher. However, when compared to the results obtained with DDSCAT in Sec. 4.3.2, the data points fit exceptionally well outside of the resonances. Again, the discretisation for the spheres at $N = 523\,305$ point dipoles appears to be insufficient when the reflectivities are evaluated near the resonances. Furthermore, the reflectivites obtained for a lattice of half spheres by GranFilm behave very similarly to the reflectivities obtained from the multipole expansion of full spheres when compared to DDSCAT. The only differences when the surface density is below 40% are slightly

greater blue shifts of the GranFilm results when compared to the DDSCAT results and slightly greater amplitudes around the resonances. For the surface density at 65% on the other hand, the quadrupole approximation in inter-particle interaction breaks down even without a substrate. In Sec. 4.2.1, the surface density limit for the quadrupole approximation is determined as 55% supported by an Al_2O_3 substrate and as 45% by an TiO_2 substrate.

The storage of the matrix $\mathbf{C}$ from Eq. (3.1.5) during the computation of the dipole moments for 24 spheres at multipole order $L = 30$ in Sec. 4.1.3 required 8 GB of RAM based on the asymptotic space complexity $16(MN)^2$B. The python code in App. A.1 is thus not suitable for computation of such large unit cells on computers with 8 GB of RAM or less. The total execution time for 221 incident wave energies was almost 40 hours on an Intel® Core™ i7-8700K processor with 6 cores. For an infinite lattice, the space complexity for the $\mathbf{C}$ matrix and the time complexity for solving for the A coefficients remain unchanged from the complexities for the finite system, as only the coefficients for one unit cell are solved for. The time complexity for building the $\mathbf{C}$ matrix on the other hand increases considerably with the interaction distance squared, $\sim R_{\text{int}}^2$.

Further improvements could be implementing the "surface mode" from ADDA to DDSCAT. The alternative would be to implement periodic boundary conditions for ADDA. As the method developed here and GranFilm are mainly focused on supported particles, the "surface mode" would allow for more relevant comparisons. Another useful implementation would be to vary the inter-dipole spacing for higher resolutions near the hot spots. The near field results obtained from the multipole expansion could be a useful initial guess for the discretisation of the spheres. Lastly, a DDSCAT-simulation with more point dipoles could be run on a computer with more cores for a higher interaction cutoff parameter γ for only the data points near the resonance. Perhaps then the results could be accurate also for energies in the immediate vicinity of the resonance.

Bibliography

[1] D. Bedeaux and J. Vlieger. *Optical properties of surfaces*. World Scientific Publishing, 2014.

[2] P. A. Letnes, I. Simonsen, and D. L. Mills. Substrate influence on the plasmonic response of clusters of spherical nanoparticles. *Phys. Rev. B*, 83(7):075426, Feb 2011.

[3] P. A. Letnes, I. Simonsen, and D. L. Mills. Plasmonic resonances at interfaces patterned by nanoparticle lattices. unpublished, Aug 2012.

[4] S. K. Lam, A. Pitrou, and S. Seibert. Numba: A LLVM-based python JIT compiler. In *LLVM '15: Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, New York, NY, USA, Nov 2015. Association for Computing Machinery.

[5] M. A. Wieczorek and M. Meschede. Shtools: Tools for working with spherical harmonics. *Geochem. Geophy. Geosy.*, 19(8):2574–2592, Aug 2018.

[6] Sopra database. `http://www.sspectra.com/sopra.html`. Accessed: 2021-06-07.

[7] B. T. Draine and P. J. Flatau. Discrete-dipole approximation for scattering calculations. *J. Opt. Soc. Am. A*, 11(4):1491–1499, Apr 1994.

[8] B. T. Draine and P. J. Flatau. Discrete-dipole approximation for periodic targets: theory and tests. *J. Opt. Soc. Am. A*, 25(11):2693–2703, Nov 2008.

[9] P. J. Flatau and B. T. Draine. Fast near field calculations in the discrete dipole approximation for regular rectilinear grids. *Opt. Express*, 20(2):1247–1252, Jan 2012.

[10] M. Seeram, G. T. Forcherio, and D. K. Roper. Shape generator for DDSCAT. `https://nanohub.org/resources/22758`, Mar 2016.

[11] U. Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, 2015.

[12] C. F. Bohren and D. R. Huffman. *Absorption and Scattering of Light by Small Particles*. John Wiley & Sons, Ltd, 1998.

[13] BHMIE. `http://scatterlib.wdfiles.com/local--files/codes/bhmie.py`. Accessed: 2021-06-09.

[14] R. Lazzari and I. Simonsen. Granfilm: a software for calculating thin-layer dielectric properties and fresnel coefficients. *Thin Solid Films*, 419(1):124–136, Jul 2002.

[15] E. Aursand. Optical properties of truncated and coated spheroidal nanoparticles on a substrate. Master thesis, NTNU, 2012.

[16] D. J. Griffiths. *Introduction to electrodynamics; 4th ed.* Pearson, 2013.

[17] Spherical harmonics. `https://brilliant.org/wiki/spherical-harmonics/`. Accessed: 2021-02-11.

[18] W. Bosch. On the computation of derivatives of legendre functions. *Phys. Chem. Earth Pt. A*, 25(9):655–659, Dec 2000.

[19] D. Bedeaux and J. Vlieger. A phenomenological theory of the dielectric properties of thin films. *Physica*, 67(1):55–73, Jul 1973.

[20] M. A. Yurkin and A. G. Hoekstra. The discrete dipole approximation: An overview and recent developments. *J. Quant. Spectrosc. Ra.*, 106(1):558–589, Jul 2007.

[21] D. Gutkowicz-Krusin and B. T. Draine. Propagation of electromagnetic waves on a rectangular lattice of polarizable points. unpublished, Apr 2004.

[22] O. S. Heavens. Optical properties of thin films. *Rep. Prog. Phys*, 23(1):1–65, Jan 1960.

[23] M. A. Yurkin and A. G. Hoekstra. The discrete-dipole-approximation code adda: Capabilities and known limitations. *J. Quant. Spectrosc. Ra.*, 112(13):2234–2247, Mar 2011.

[24] M. A. Yurkin and M. Huntemann. Rigorous and fast discrete dipole approximation for particles near a plane interface. *J. Phys. Chem. C*, 119(52):29088–29094, Dec 2015.

[25] B. T. Draine and J. Goodman. Beyond Clausius-Mossotti: Wave Propagation on a Polarizable Point Lattice and the Discrete Dipole Approximation. *Astrophys. J.*, 405(2):685–697, Mar 1993.

[26] J. J. Goodman, B. T. Draine, and P. J. Flatau. Application of fast-fourier-transform techniques to the discrete-dipole approximation. *Opt. Lett.*, 16(15):1198–1200, Aug 1991.

[27] B. T. Draine and P. J. Flatau. User guide for the discrete dipole approximation code DDSCAT 7.3. `https://arxiv.org/pdf/1305.6497.pdf`, Jul 2020.

Appendix A

Multipole expansion method

A.1 Python script

```
#
# The MultiPole Expansion Software (MPES)
#
#              by
#
#        Fredrik Knapskog
#
# Trondheim, 28-May-2021
#
#     --- o0o ---
#
# Routines:  get_spherical_coords
#            get_l
#            get_m
#            H
#            get_C
#            get_b
#            get_k
#            get_B
#            get_B_0
#            get_p
#            get_i1
#            get_j1
#            get_length
#            E_angles
#            susceptibilities
#            s_polarization
#            p_polarization
#            is_outside
#            coord_table
#            spherical_harm
#            potential_grid
```

```
#            field_grid
#
#            main
#


# ======================================================================
#  Importing libraries
# ======================================================================

import numpy as np
import scipy
from matplotlib import pyplot as plt
from scipy.special import sph_harm, binom
from scipy import constants
from scipy.sparse.linalg import gmres #Iterative solver
import warnings
from time import *
from pyshtools import expand
from numba import jit, prange
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
from IPython.display import clear_output
from pysopra import EpsilonSOPRA, micron2eV, eV2micron

# ----------------------------------------------------------------------
# --- end importing libraries
# ----------------------------------------------------------------------

# ======================================================================
#  Implementing functions
# ======================================================================

#Returns five coordinates. The first two are the polar and azimuthal
#angles of point p_2 relative to point p_1. Third is the polar angle
#of p_2s image multipole relative to p_1. Fourth is the distance between
#p_1 and p_2 and fifth is the distance between p_2s image multipole
#and p_1.
@jit(nopython=True)
def get_spherical_coords(p_1, p_2):
    polar_angle = np.arctan2(np.sqrt((p_2[0]-p_1[0])**2 + (p_2[1]-p_1[1])**2),
    ↪  p_2[2]-p_1[2])
    azimuthal_angle = np.arctan2(p_2[1]-p_1[1], p_2[0]-p_1[0])
    image_polar_angle = np.arctan2(np.sqrt((p_2[0]-p_1[0])**2 +
    ↪  (p_2[1]-p_1[1])**2), -p_2[2]-p_1[2])
    R = np.sqrt((p_2[0]-p_1[0])**2 + (p_2[1]-p_1[1])**2 + (p_2[2]-p_1[2])**2)
    image_R = np.sqrt((p_2[0]-p_1[0])**2 + (p_2[1]-p_1[1])**2 + (p_2[2]+p_1[2])**2)
    return polar_angle, azimuthal_angle, image_polar_angle, R, image_R
```

```
79
80  # ------------------------------------------------------------------------
81
82  #Returns the index l from matrix index ind (with numba)
83  @jit(nopython=True)
84  def get_l(ind):
85      return np.sqrt(ind+1)//1
86
87  # ------------------------------------------------------------------------
88
89  #Returns the index m from matrix index ind (with numba)
90  @jit(nopython=True)
91  def get_m(ind):
92      l = get_l(ind)
93      return ind + 1 -l*(l+1)
94
95  # ------------------------------------------------------------------------
96
97  #Returns the index l from matrix index ind (without numba)
98  def get_l1(ind):
99      return np.sqrt(ind+1)//1
100
101 # ------------------------------------------------------------------------
102
103 #Returns the index m from matrix index ind (without numba)
104 def get_m1(ind):
105     l = get_l(ind)
106     return ind + 1 -l*(l+1)
107
108 # ------------------------------------------------------------------------
109
110 #Computes H(l_j, m_j | l_i, m_i) for an array of indices k
111 def H(k, M):
112     k1 = k//M
113     l1 = k%M
114     l_i, m_i, l_j, m_j = get_l1(l1), get_m1(l1), get_l1(k1), get_m1(k1)
115     l = l_i + l_j
116     m = m_i - m_j
117     return np.sqrt(4*np.pi)*(-1)**(l_i+m_j)*np.sqrt((2*l_i+1)/((2*l_j+1)*(2*l+1)))
        ↪  * np.sqrt(binom(l+m, l_i+m_i)*binom(l-m, l_j+m_j))
118
119 # ------------------------------------------------------------------------
120
121 #Computes H(0, 0 | l_i, m_i) for an array of indices k
122 def H1(k):
123     l_j, m_j = 0, 0
124     l_i, m_i = get_l1(k), get_m1(k)
125     l = l_i + l_j
```

```
126     m = m_i - m_j
127     return np.sqrt(4*np.pi)*(-1)**(l_i+m_j)*np.sqrt((2*l_i+1)/((2*l_j+1)*(2*l+1)))
        ↪  * np.sqrt(binom(l+m, l_i+m_i)*binom(l-m, l_j+m_j))
128
129 # ----------------------------------------------------------------------------
130
131 #Builds C matrix
132 @jit(nopython=True, parallel = True)
133 def get_C(C, L_parallel, L_orthogonal, M, N_cell, lattice_points, centre_point,
    ↪  lattice_vectors, epsilon, r, beta, epsilon_plus, coords, spherical_harmonics,
    ↪  image_spherical_harmonics, Hs, k_vec, b_x, b_y, alpha):
134     M_parallel = (L_parallel+1)**2 - 1
135     for k in range(N_cell*M):
136         j, k1 = k//M, k%M
137         l_j, m_j = get_l(k1), get_m(k1)
138         C[k,k] += r[j]**(-2*l_j-1)*(l_j*epsilon[j] +
            ↪  epsilon_plus*(l_j+1))/(l_j*(epsilon[j] - epsilon_plus))
139         for point in range(lattice_points):
140             phase_factor = np.exp(1j*(k_vec[0]*lattice_vectors[point, 0] +
                ↪  k_vec[1]*lattice_vectors[point, 1]))
141             if point != centre_point:
142                 for l in prange(N_cell*M_parallel):
143                     i, l1 = l//M_parallel, l%M_parallel
144                     l_i, m_i = get_l(l1), get_m(l1)
145                     l2, m2 = l_i+l_j, m_i-m_j
146                     ind = int(l2*(l2+1)/2+np.abs(m2))
147                     element1 = Hs[k1,l1]/coords[j,i,point,3]**(l2+1)
148                     element2 = Hs[k1,l1]/coords[j,i,point,4]**(l2+1)
149                     if m2 >= 0:
150                         C[k,M*i+l1] += phase_factor *
                            ↪  (element1*spherical_harmonics[j,i,point,ind] +
                            ↪  element2*(-1)**(l_i+m_i)*beta *
                            ↪  image_spherical_harmonics[j,i,point,ind])
151                     else:
152                         C[k,M*i+l1] += phase_factor * (element1*(-1)**m2 *
                            ↪  np.conj(spherical_harmonics[j,i,point,ind]) +
                            ↪  element2*(-1)**(l_i+m_i +m2)*beta *
                            ↪  np.conj(image_spherical_harmonics[j,i,point,ind]))
153             else:
154                 for l in prange(N_cell*M):
155                     i, l1 = l//M, l%M
156                     l_i, m_i = get_l(l1), get_m(l1)
157                     if i != j:
158                         if l_i <= L_parallel:
159                             l2, m2 = l_i+l_j, m_i-m_j
160                             ind = int(l2*(l2+1)/2+np.abs(m2))
161                             element1 = Hs[k1,l1]/coords[j,i,point,3]**(l2+1)
162                             element2 = Hs[k1,l1]/coords[j,i,point,4]**(l2+1)
```

```
                                if m2 >= 0:
                                    C[k,l] += element1 *
                                    ↪  spherical_harmonics[j,i,point,ind] +
                                    ↪  element2*(-1)**(l_i+m_i)*beta *
                                    ↪  image_spherical_harmonics[j,i,point,ind]
                                else:
                                    C[k,l] += element1*(-1)**m2 *
                                    ↪  np.conj(spherical_harmonics[j,i,point,ind]) +
                                    ↪  element2*(-1)**(l_i+m_i +m2)*beta *
                                    ↪  np.conj(image_spherical_harmonics[j,i,point,ind])
                        else:
                            if l_i <= L_orthogonal and m_j == m_i:
                                l2, m2 = l_i+l_j, m_i-m_j
                                ind = int(l2*(l2+1)/2)
                                C[k,l] += Hs[k1,l1] / coords[j,i,point,4]**(l2+1) *
                                ↪  beta*(-1)**(l_i+m_i) *
                                ↪  image_spherical_harmonics[j,i,point,ind]
    return C

# ----------------------------------------------------------------------------

#Creates b-vector from electric field direction
def get_b(theta_0, phi_0, M, N):
    b = np.zeros(M, dtype = np.complex128)
    b[0] = np.sqrt(2*np.pi/3)*np.sin(theta_0)*np.exp(1j*phi_0)
    b[1] = np.sqrt(4*np.pi/3)*np.cos(theta_0)
    b[2] = -np.sqrt(2*np.pi/3)*np.sin(theta_0)*np.exp(-1j*phi_0)
    return np.tile(b,N)

# ----------------------------------------------------------------------------

#Creates k-vector from incident wave direction and wave number
def get_k(omega, theta_0, phi_0, length_scale):
    HC = 2 * np.pi * 0.1973269631
    return
    ↪  2*np.pi/(HC/omega)*length_scale*1e3*np.sin(theta_0)*np.array([np.cos(phi_0),
    ↪  np.sin(phi_0), 0])*0


# ----------------------------------------------------------------------------

#Computes B coefficients from solved A coefficients
@jit(nopython=True, parallel = True)
def get_B(M, N, A, b, beta, r, Hs, coords, spherical_harmonics,
↪  image_spherical_harmonics, L_parallel, L_orthogonal):
    B = np.zeros(M*N, dtype = np.complex128)
    for k in prange(M*N):
        j, k1 = k//M, k%M
```

```
        l_j, m_j = get_l(k1), get_m(k1)
        B[k] += -r[j]**(1-l_j)*b[k] + A[k]*r[j]**(-2*l_j-1)
        for l in prange(M*N):
            i, l1 = l//M, l%M
            l_i, m_i = get_l(l1), get_m(l1)
            if i != j and l_i <= L_parallel:
                l2, m2 = l_i+l_j, m_i-m_j
                ind = int(l2*(l2+1)/2+np.abs(m2))
                element1 = Hs[k1,l1]/coords[j,i,3]**(l2+1)
                element2 = Hs[k1,l1]/coords[j,i,4]**(l2+1)
                if m2 >= 0:
                    B[k] += A[l]*(element1*spherical_harmonics[j,i,ind] +
                    ↪  element2*(-1)**(l_i+m_i)*beta*image_spherical_harmonics[j,i,ind])
                else:
                    B[k] +=
                    ↪  A[l]*(element1*(-1)**m2*np.conj(spherical_harmonics[j,i,ind])
                    ↪  + element2*(-1)**(l_i+m_i
                    ↪  +m2)*beta*np.conj(image_spherical_harmonics[j,i,ind]))
            elif i == j and l_i <= L_orthogonal and m_j == m_i:
                l2= l_i+l_j
                ind = int(l2*(l2+1)/2)
                B[k] += A[l]*Hs[k1,l1]/coords[j,j,4]**(l2+1) *
                ↪  beta*(-1)**(l_i+m_i)*image_spherical_harmonics[j, j, ind]
    return B

# ------------------------------------------------------------------------

#Computes the B coefficient for m = l = 0
@jit(nopython=True)
def get_B_0(B_0, H_0, M, N, A, beta, r, coords, spherical_harmonics,
↪  image_spherical_harmonics, L_parallel, L_orthogonal, origin, b):
    for j in range(N):
        for l in range(M*N):
            i, l1 = l//M, l%M
            l_i, m_i = get_l(l1), get_m(l1)
            if i != j and l_i <= L_parallel:
                l2, m2 = l_i, m_i
                ind = int(l2*(l2+1)/2+np.abs(m2))
                element1 = H_0[l1]/coords[j,i,3]**(l2+1)
                element2 = H_0[l1]/coords[j,i,4]**(l2+1)
                if m2 >= 0:
                    B_0[j] += A[l]*(element1*spherical_harmonics[j,i,ind] +
                    ↪  element2*(-1)**(l_i+m_i)*beta*image_spherical_harmonics[j,i,ind])
                else:
                    B_0[j] +=
                    ↪  A[l]*(element1*(-1)**m2*np.conj(spherical_harmonics[j,i,ind])
                    ↪  + element2*(-1)**(l_i+m_i
                    ↪  +m2)*beta*np.conj(image_spherical_harmonics[j,i,ind]))
```

```
238             elif i == j and l_i <= L_orthogonal and m_i == 0:
239                 l2= l_i
240                 ind = int(l2*(l2+1)/2)
241                 B_0[j] += A[l]*H_0[l1]/coords[j,j,4]**(l2+1) *
                    ↪  beta*(-1)**(l_i+m_i)*image_spherical_harmonics[j, j, ind]
242         if j != origin:
243             B_0[j] += -np.sqrt(4*np.pi)*coords[origin, j,
                ↪  3]*(-b[0]*np.conj(spherical_harmonics[origin, j, 2]) +
                ↪  b[1]*spherical_harmonics[origin, j, 1] +
                ↪  b[2]*spherical_harmonics[origin, j, 2])
244     return B_0
245
246 # ------------------------------------------------------------------------
247
248 #Compute dimensionless dipole moment from A coefficients
249 def get_p(A, i, M, r):
250     k = i*M
251     p_x = np.sqrt(3/(8*np.pi))*(A[k]-A[k+2])/r[i]**2
252     p_y = -1j*np.sqrt(3/(8*np.pi))*(A[k]+A[k+2])/r[i]**2
253     p_z = np.sqrt(3/(4*np.pi))*A[k+1]/r[i]**2
254     return np.real(np.sqrt(p_x*np.conj(p_x) + p_y*np.conj(p_y) + p_z*np.conj(p_z)))
255
256 # ------------------------------------------------------------------------
257
258 #Compute positions for split ring of N spheres,
259 #separated by a distance d and radius r a height
260 #h above the substrate
261 def split_ring(N, d, r, h):
262     dtheta = 2*np.pi/(N+1)
263     R = (r+d/2)/np.sin(dtheta/2)
264     ring_pos = np.zeros([N, 3])
265     theta = 0
266     for i in range(N):
267         theta += dtheta
268         ring_pos[i] = (R*np.cos(theta), R*np.sin(theta), r+h)
269     return ring_pos
270
271 # ------------------------------------------------------------------------
272
273 #Compute positions for full ring
274 def full_ring(N, d, r, h):
275     dtheta = 2*np.pi/N
276     R = (r+d/2)/np.sin(dtheta/2)
277     ring_pos = np.zeros([N, 3])
278     theta = 0
279     for i in range(N):
280         ring_pos[i] = (R*np.cos(theta), R*np.sin(theta), r+h)
281         theta += dtheta
```

```
282     return ring_pos
283 
284 # ------------------------------------------------------------------------
285 
286 #Finds all i1 lattice coordinates to be included
287 #at lattice coordinate j1
288 def get_i1(R, b_x, b_y, alpha, j1):
289     R1 = np.abs(j1*np.sin(alpha)*b_y)
290     if R1 < R:
291         i1_2 = int(np.floor(1/b_x*(np.sqrt(R**2 - (j1*np.sin(alpha)*b_y)**2) -
        ↪  j1*np.cos(alpha)*b_y)))
292         i1_1 = int(np.ceil(1/b_x*(-np.sqrt(R**2 - (j1*np.sin(alpha)*b_y)**2) -
        ↪  j1*np.cos(alpha)*b_y)))
293         return np.arange(i1_1, i1_2+1)
294     elif R1 == R:
295         i1_1 = int(1/b_x*(-j1*np.cos(alpha)*b_y))
296         return np.arange(i1_1, i1_1+1)
297     else:
298         return np.array([])
299 
300 # ------------------------------------------------------------------------
301 
302 #Finds all lattice coordinates j1 to be included
303 #based on the interaction distance R and
304 #lattice distance b_y
305 def get_j1(R, b_y, alpha):
306     j1_1 = R//(b_y*np.sin(alpha))
307     return np.arange(-j1_1, j1_1+1)
308 
309 # ------------------------------------------------------------------------
310 
311 #Returns number of lattice sites to be included
312 def get_length(R, b_x, b_y, alpha):
313     length = 0
314     j1s = get_j1(R, b_y, alpha)
315     for j1 in j1s:
316         length += len(get_i1(R, b_x, b_y, alpha, j1))
317     return length
318 
319 # ------------------------------------------------------------------------
320 
321 #Returns the electric field angles for p- and s-polarized
322 #light from incident wave angles
323 def E_angles(theta_0, phi_0):
324     return np.pi/2 - theta_0, phi_0, np.pi/2, np.pi/2 + phi_0
325 
326 # ------------------------------------------------------------------------
327 
```

```
328 #Computes the susceptibilities gamma and beta from A coefficients
329 def susceptibilities(A, M, theta_0, phi_0, epsilon_plus, d, rho):
330     if np.abs(np.cos(theta_0)) < 1e-6:
331         alpha_ort_0 = 0
332         alpha_ort_10 = 0
333         alpha_par_0 =
        ↪  -4*np.pi*epsilon_plus*A[2]/(np.sqrt(2*np.pi/3)*np.sin(theta_0)*np.exp(-1j*phi_0))
334         alpha_par_10 =
        ↪  -4*np.pi*epsilon_plus*A[6]/(np.sqrt(6*np.pi/5)*np.sin(theta_0)*np.exp(-1j*phi_0))
335     elif np.abs(np.sin(theta_0)) < 1e-6:
336         alpha_par_0 = 0
337         alpha_par_10 = 0
338         alpha_ort_0 = -2*np.pi*epsilon_plus*A[1]/(np.sqrt(np.pi/3)*np.cos(theta_0))
339         alpha_ort_10 = np.pi*epsilon_plus*A[5]/(np.sqrt(np.pi/5)*np.cos(theta_0))
340     else:
341         alpha_par_0 =
        ↪  -4*np.pi*epsilon_plus*A[2]/(np.sqrt(2*np.pi/3)*np.sin(theta_0)*np.exp(-1j*phi_0))
342         alpha_ort_0 = 2*np.pi*epsilon_plus*A[1]/(np.sqrt(np.pi/3)*np.cos(theta_0))
343         alpha_par_10 =
        ↪  -4*np.pi*epsilon_plus*A[6]/(np.sqrt(6*np.pi/5)*np.sin(theta_0)*np.exp(-1j*phi_0))
344         alpha_ort_10 = np.pi*epsilon_plus*A[5]/(np.sqrt(np.pi/5)*np.cos(theta_0))
345
346     gamma_e = rho*alpha_par_0
347     beta_e = rho*alpha_ort_0/epsilon_plus**2
348
349     return gamma_e, beta_e
350
351 # ------------------------------------------------------------------------
352
353 #Returns the reflection and transmission coefficients
354 #for s-polarized light
355 def s_polarization(theta, omega, gamma_e, epsilon_plus, epsilon_minus, scale):
356     n_plus, n_minus = np.sqrt(epsilon_plus), np.sqrt(epsilon_minus)
357     theta_t = np.arcsin(n_plus/n_minus*np.sin(theta))
358     HC = 1.97e-7/scale
359     cos_theta, cos_theta_t = np.cos(theta), np.cos(theta_t)
360     r_s = n_plus*cos_theta - n_minus*cos_theta_t + 1j*(omega/HC)*gamma_e
361     t_s = 4*(n_minus*cos_theta)**2
362     D_s = n_plus*cos_theta + n_minus*cos_theta_t - 1j*(omega/HC)*gamma_e
363     return r_s/D_s, t_s/D_s
364
365 # ------------------------------------------------------------------------
366
367 #Returns the reflection and transmission coefficients
368 #for p-polarized light
369 def p_polarization(theta, omega, gamma_e, epsilon_plus, epsilon_minus, beta_e,
    ↪  scale):
370     n_plus, n_minus = np.sqrt(epsilon_plus), np.sqrt(epsilon_minus)
```

```
371     theta_t = np.arcsin(n_plus/n_minus*np.sin(theta))
372     HC = 1.97e-7/scale
373     cos_theta, cos_theta_t = np.cos(theta), np.cos(theta_t)
374     r_p = (n_minus*cos_theta -
        ↪  n_plus*cos_theta_t)*(1-(omega/(2*HC))**2*epsilon_plus*gamma_e*beta_e*np.sin(theta)**2)
        ↪  - 1j*omega/HC*gamma_e*cos_theta*cos_theta_t +
        ↪  1j*omega/HC*n_plus*n_minus*epsilon_plus*beta_e*np.sin(theta)**2
375     t_p = 2*n_minus*cos_theta
376     D_p = (n_minus*cos_theta +
        ↪  n_plus*cos_theta_t)*(1-(omega/(2*HC))**2*epsilon_plus*gamma_e*beta_e*np.sin(theta)**2)
        ↪  - 1j*omega/HC*gamma_e*cos_theta*cos_theta_t -
        ↪  1j*omega/HC*n_plus*n_minus*epsilon_plus*beta_e*np.sin(theta)**2
377     return r_p/D_p, t_p/D_p
378
379 # ------------------------------------------------------------------------
380
381 #Determines whether a point is contained by a sphere or not.
382 #If contained then the sphere containing it is returned too.
383 @jit(nopython=True)
384 def is_outside(point, N, r, p):
385     status = True
386     sphere = -1
387     for i in range(N):
388         if np.sqrt((point[0]-p[i,0])**2 + (point[1]-p[i,1])**2 +
            ↪  (point[2]-p[i,2])**2) < r[i]:
389             status = False
390             sphere = i
391             break
392     return status, sphere
393
394 # ------------------------------------------------------------------------
395
396 #Returns the x and z coordinates for the nearfield grid.
397 #Also computes the five coordinates from get_spherical_coords
398 #for all the spheres relative to all the grid points,
399 #and relative to all the image positions of the grid.
400 # @jit(nopython=True) #, parallel = True)
401 @jit(nopython=True) #, parallel = True)
402 def coord_table(table, p, N, x0, x1, z0, z1, Nx, Nz, y):
403     x = np.linspace(x0, x1, Nx)
404     z = np.linspace(z0, z1, Nz)
405
406     image_positions = p.copy().T
407     image_positions[-1] *= -1
408     image_positions = image_positions.T
409     image_table = table.copy()
410
411     for k in range(N):
```

```
412         for i in range(Nz):
413             for j in range(Nx):
414                 point = np.array([x[j], y, z[i]])
415                 table[k,i,j,0], table[k,i,j,1], table[k,i,j,2], table[k,i,j,3],
                    ↪ table[k,i,j,4] = get_spherical_coords(p[k], point)
416                 image_table[k,i,j,0], image_table[k,i,j,1], image_table[k,i,j,2],
                    ↪ image_table[k,i,j,3], image_table[k,i,j,4] =
                    ↪ get_spherical_coords(image_positions[k], point)
417     return x, z, table, image_table
418
419 # ------------------------------------------------------------------------
420
421 #Computes the spherical harmonics for the angles given by the
422 #positions and the corresponding image positions on the grid
423 #from coord_table
424 def spherical_harm2(coords, image_coords, L, N, Nx, Nz):
425     spherical_harmonics = np.zeros([N, Nz, Nx, (L+2)*(L+1)//2], dtype =
        ↪ np.complex128)
426     image_spherical_harmonics = np.zeros([N, Nz, Nx, (L+2)*(L+1)//2], dtype =
        ↪ np.complex128)
427     for k in range(N):
428         for i in range(Nz):
429             for j in range(Nx):
430                 spherical_harmonics[k,i,j] = expand.spharm(L,
                    ↪ np.real(coords[k,i,j,0]), np.real(coords[k,i,j,1]),
                    ↪ normalization = 'ortho', kind = 'complex', csphase = -1, packed
                    ↪ = True, degrees = False)[0]
431                 image_spherical_harmonics[k,i,j] = expand.spharm(L,
                    ↪ np.real(image_coords[k,i,j,0]), np.real(image_coords[k,i,j,1]),
                    ↪ normalization = 'ortho', kind = 'complex', csphase = -1, packed
                    ↪ = True, degrees = False)[0]
432
433     return spherical_harmonics, image_spherical_harmonics
434
435
436 # ------------------------------------------------------------------------
437
438 #Computes the spherical harmonics for the angles given by the
439 #positions and the corresponding image positions on the grid
440 #from coord_table (uses scipy instead of shtools, is 15
441 #times faster for the near field calculations, but is
442 #limited to L<43.)
443 def spherical_harm(sph, coords, image_coords, L, N, Nx, Nz):
444     imsph = sph.copy()
445     k = (L+2)*(L+1)//2
446     l = np.floor(np.sqrt(2*np.arange(k)+0.25)-0.5)
447     m = np.arange(k) - l*(l+1)/2
448
```

```
449     ls = np.tile(l.astype(int), N*Nz*Nx)
450     ms = np.tile(m.astype(int), N*Nz*Nx)
451 
452     sph = np.reshape(sph_harm(ms, ls, np.repeat(np.reshape(coords[:,:,:,1],
        ↪  N*Nz*Nx), k), np.repeat(np.reshape(coords[:,:,:,0], N*Nz*Nx), k)), [N, Nz,
        ↪  Nx, k])
453     imsph = np.reshape(sph_harm(ms, ls, np.repeat(np.reshape(image_coords[:,:,:,1],
        ↪  N*Nz*Nx), k), np.repeat(np.reshape(image_coords[:,:,:,0], N*Nz*Nx), k)),
        ↪  [N, Nz, Nx, k])
454 
455     return sph, imsph
456 
457 # ------------------------------------------------------------------------
458 
459 #Evaluates the complex scalar potential divided by the
460 #potential from the electric field from the incident
461 #wave at each point on the grid from coord_table
462 @jit(nopython=True, parallel = True)
463 def potential_grid(pot, origin, x, y, z, Nx, Nz, coord_table, image_coord_table,
    ↪  spherical_harmonics, image_spherical_harmonics, A, r, N, M, beta, b, positions,
    ↪  B, beta2, epsilon_plus, epsilon_minus, B0, theta):
464     for i in prange(Nz):
465         for j in prange(Nx):
466             point = np.array([x[j], y, z[i]])
467             outside, sphere = is_outside(point, N, r, positions)
468             if outside:
469                 if point[2] >= 0:
470                     pot[i, j] += coord_table[origin,i,j,3] *
                        ↪  b[0]*np.conj(spherical_harmonics[origin,i,j,2])
471                     pot[i, j] += -coord_table[origin,i,j,3] *
                        ↪  b[1]*spherical_harmonics[origin,i,j,1]
472                     pot[i, j] += -coord_table[origin,i,j,3] *
                        ↪  b[2]*spherical_harmonics[origin,i,j,2]
473                     for k in range(N*M):
474                         j1, k1 = k//M, k%M
475                         l = get_l(k1)
476                         m = get_m(k1)
477                         ind = int(l*(l+1)/2+np.abs(m))
478                         if m >= 0:
479                             pot[i, j] += A[k] * (coord_table[j1,i,j,3]**(-l-1) *
                                ↪  spherical_harmonics[j1,i,j,ind] + (-1)**(l+m)*beta
                                ↪  * image_coord_table[j1,i,j,3]**(-l-1) *
                                ↪  image_spherical_harmonics[j1,i,j,ind])
480                         else:
```

```
481                             pot[i, j] += (-1)**m*A[k] *
                                ↪  (coord_table[j1,i,j,3]**(-l-1) *
                                ↪  np.conj(spherical_harmonics[j1,i,j,ind]) +
                                ↪  (-1)**(l+m)*beta *
                                ↪  image_coord_table[j1,i,j,3]**(-l-1) *
                                ↪  np.conj(image_spherical_harmonics[j1,i,j,ind]))
482                     else:
483                         pot[i, j] += coord_table[origin,i,j,3] *
                            ↪  b[0]*np.conj(spherical_harmonics[origin,i,j,2])
484                         pot[i, j] += -coord_table[origin,i,j,3] *
                            ↪  b[1]*spherical_harmonics[origin,i,j,1] *
                            ↪  epsilon_plus/epsilon_minus
485                         pot[i, j] += -coord_table[origin,i,j,3] *
                            ↪  b[2]*spherical_harmonics[origin,i,j,2]
486                         pot[i, j] += -positions[origin,2] * (epsilon_plus/epsilon_minus
                            ↪  -1) * np.cos(theta)
487                         for k in range(N*M):
488                             j1, k1 = k//M, k%M
489                             l = get_l(k1)
490                             m = get_m(k1)
491                             ind = int(l*(l+1)/2+np.abs(m))
492                             if m >= 0:
493                                 pot[i, j] += beta2*A[k] * coord_table[j1,i,j,3]**(-l-1)
                                    ↪  * spherical_harmonics[j1,i,j,ind]
494                             else:
495                                 pot[i, j] += beta2*(-1)**m*A[k] *
                                    ↪  coord_table[j1,i,j,3]**(-l-1) *
                                    ↪  np.conj(spherical_harmonics[j1,i,j,ind])
496                 else:
497                     pot[i, j] += np.sqrt(1/(4*np.pi))*B0[sphere]
498                     for k in range(M):
499                         l = get_l(k)
500                         m = get_m(k)
501                         ind = int(l*(l+1)/2+np.abs(m))
502                         if m >= 0:
503                             pot[i, j] += B[sphere*M+k] * coord_table[sphere,i,j,3]**l *
                                ↪  spherical_harmonics[sphere,i,j,ind]
504                         else:
505                             pot[i, j] += (-1)**m * B[sphere*M+k] *
                                ↪  coord_table[sphere,i,j,3]**l *
                                ↪  np.conj(spherical_harmonics[sphere,i,j,ind])
506     return pot
507
508 # ---------------------------------------------------------------------------
509
510 #Evaluates the length of the electric field divided by
511 #the electric field from the incident wave for
512 #each point on the grid from coord_table
```

```
513 @jit(nopython=True)
514 def field_grid(E, origin, x, y, z, Nx, Nz, coord_table, image_coord_table,
    ↪ spherical_harmonics, image_spherical_harmonics, A, r, N, M, beta, b, positions,
    ↪ B, beta2, epsilon_plus, epsilon_minus):
515     for i in range(Nz):
516         for j in range(Nx):
517             x1, y1, z1 = 0, 0, 0
518             point = np.array([x[j], y, z[i]])
519             outside, sphere = is_outside(point, N, r, positions)
520             if outside:
521                 dSH0_theta = 1/np.tan(coord_table[origin,i,j,0]) *
                    ↪ np.conj(spherical_harmonics[origin,i,j,2]) +
                    ↪ np.sqrt(2)*np.exp(-1j*coord_table[origin,i,j,1]) *
                    ↪ spherical_harmonics[origin,i,j,1]
522                 dSH1_theta = np.sqrt(2) * np.exp(-1j*coord_table[origin,i,j,1]) *
                    ↪ spherical_harmonics[origin,i,j,2]
523                 dSH2_theta = spherical_harmonics[origin,i,j,2] /
                    ↪ np.tan(coord_table[origin,i,j,0])
524                 dSH0_phi = 1j*np.conj(spherical_harmonics[origin,i,j,2])
525                 dSH2_phi = 1j*spherical_harmonics[origin,i,j,2]
526                 dSH0_r = -np.conj(spherical_harmonics[origin,i,j,2])
527                 dSH1_r = spherical_harmonics[origin,i,j,1]
528                 dSH2_r = spherical_harmonics[origin,i,j,2]
529                 if point[2] >= 0:
530                     x1 += np.sin(coord_table[origin,i,j,0]) *
                        ↪ np.cos(coord_table[origin,i,j,1]) * (b[0]*dSH0_r +
                        ↪ b[1]*dSH1_r + b[2]*dSH2_r)
531                     x1 += np.cos(coord_table[origin,i,j,0]) *
                        ↪ np.cos(coord_table[origin,i,j,1]) * (b[0]*dSH0_theta +
                        ↪ b[1]*dSH1_theta + b[2]*dSH2_theta)
532                     x1 += -np.sin(coord_table[origin,i,j,1]) /
                        ↪ np.sin(coord_table[origin,i,j,0]) * (b[0]*dSH0_phi +
                        ↪ b[2]*dSH2_phi)
533
534                     y1 += np.sin(coord_table[origin,i,j,0]) *
                        ↪ np.sin(coord_table[origin,i,j,1]) * (b[0]*dSH0_r +
                        ↪ b[1]*dSH1_r + b[2]*dSH2_r)
535                     y1 += np.cos(coord_table[origin,i,j,0]) *
                        ↪ np.sin(coord_table[origin,i,j,1]) * (b[0]*dSH0_theta +
                        ↪ b[1]*dSH1_theta + b[2]*dSH2_theta)
536                     y1 += np.cos(coord_table[origin,i,j,1]) /
                        ↪ np.sin(coord_table[origin,i,j,0]) * (b[0]*dSH0_phi +
                        ↪ b[2]*dSH2_phi)
537
538                     z1 += np.cos(coord_table[origin,i,j,0]) * (b[0]*dSH0_r +
                        ↪ b[1]*dSH1_r + b[2]*dSH2_r)
539                     z1 += -np.sin(coord_table[origin,i,j,0]) * (b[0]*dSH0_theta +
                        ↪ b[1]*dSH1_theta + b[2]*dSH2_theta)
```

```
                    for k in range(N*M):
                        j1, k1 = k//M, k%M
                        l = get_l(k1)
                        m = get_m(k1)
                        ind = int(l*(l+1)/2+np.abs(m))
                        dSH_theta, dISH_theta = 0, 0
                        dSH_phi, dISH_phi = 0, 0
                        dSH_r, dISH_r = 0, 0
                        if m >= 0:
                            dSH_theta += m*spherical_harmonics[j1,i,j,ind] /
                            ↪  np.tan(coord_table[j1,i,j,0])
                            dISH_theta += m*image_spherical_harmonics[j1,i,j,ind] /
                            ↪  np.tan(image_coord_table[j1,i,j,0])
                            dSH_phi += 1j*m*spherical_harmonics[j1,i,j,ind]
                            dISH_phi += 1j*m*image_spherical_harmonics[j1,i,j,ind]
                            dSH_r += spherical_harmonics[j1,i,j,ind]
                            dISH_r += image_spherical_harmonics[j1,i,j,ind]
                            if m != l:
                                dSH_theta += np.sqrt((l-m)*(l+m+1)) *
                                ↪  np.exp(-1j*coord_table[j1,i,j,1]) *
                                ↪  spherical_harmonics[j1,i,j,ind+1]
                                dISH_theta += np.sqrt((l-m)*(l+m+1)) *
                                ↪  np.exp(-1j*image_coord_table[j1,i,j,1]) *
                                ↪  image_spherical_harmonics[j1,i,j,ind+1]
                        else:
                            dSH_theta += (-1)**m*m *
                            ↪  np.conj(spherical_harmonics[j1,i,j,ind]) /
                            ↪  np.tan(coord_table[j1,i,j,0]) +
                            ↪  (-1)**m*np.sqrt((l-m)*(l+m+1)) *
                            ↪  np.exp(-1j*coord_table[j1,i,j,1]) *
                            ↪  np.conj(spherical_harmonics[j1,i,j,ind-1])
                            dISH_theta += (-1)**m*m *
                            ↪  np.conj(image_spherical_harmonics[j1,i,j,ind]) /
                            ↪  np.tan(image_coord_table[j1,i,j,0]) +
                            ↪  (-1)**m*np.sqrt((l-m)*(l+m+1)) *
                            ↪  np.exp(-1j*image_coord_table[j1,i,j,1]) *
                            ↪  np.conj(image_spherical_harmonics[j1,i,j,ind-1])
                            dSH_phi += (-1)**m*1j*m *
                            ↪  np.conj(spherical_harmonics[j1,i,j,ind])
                            dISH_phi += (-1)**m*1j*m *
                            ↪  np.conj(image_spherical_harmonics[j1,i,j,ind])
                            dSH_r += (-1)**m *
                            ↪  np.conj(spherical_harmonics[j1,i,j,ind])
                            dISH_r += (-1)**m *
                            ↪  np.conj(image_spherical_harmonics[j1,i,j,ind])
```

```
                    x1 += (l+1)*A[k]*np.sin(coord_table[j1,i,j,0]) *
                    ↪  np.cos(coord_table[j1,i,j,1]) *
                    ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_r
                    x1 += (l+1)*A[k]*np.sin(image_coord_table[j1,i,j,0]) *
                    ↪  np.cos(image_coord_table[j1,i,j,1]) *
                    ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                    ↪  dISH_r

                    x1 += -A[k]*np.cos(coord_table[j1,i,j,0]) *
                    ↪  np.cos(coord_table[j1,i,j,1]) *
                    ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_theta
                    x1 += -A[k]*np.cos(image_coord_table[j1,i,j,0]) *
                    ↪  np.cos(image_coord_table[j1,i,j,1]) *
                    ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                    ↪  dISH_theta

                    x1 += A[k]*np.sin(coord_table[j1,i,j,1]) /
                    ↪  np.sin(coord_table[j1,i,j,0]) *
                    ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_phi
                    x1 += A[k]*np.sin(image_coord_table[j1,i,j,1]) /
                    ↪  np.sin(image_coord_table[j1,i,j,0]) *
                    ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                    ↪  dISH_phi


                    y1 += (l+1)*A[k]*np.sin(coord_table[j1,i,j,0]) *
                    ↪  np.sin(coord_table[j1,i,j,1]) *
                    ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_r
                    y1 += (l+1)*A[k]*np.sin(image_coord_table[j1,i,j,0]) *
                    ↪  np.sin(image_coord_table[j1,i,j,1]) *
                    ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                    ↪  dISH_r

                    y1 += -A[k]*np.cos(coord_table[j1,i,j,0]) *
                    ↪  np.sin(coord_table[j1,i,j,1]) *
                    ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_theta
                    y1 += -A[k]*np.cos(image_coord_table[j1,i,j,0]) *
                    ↪  np.sin(image_coord_table[j1,i,j,1]) *
                    ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                    ↪  dISH_theta

                    y1 += -A[k]*np.cos(coord_table[j1,i,j,1]) /
                    ↪  np.sin(coord_table[j1,i,j,0]) *
                    ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_phi
                    y1 += -A[k]*np.cos(image_coord_table[j1,i,j,1]) /
                    ↪  np.sin(image_coord_table[j1,i,j,0]) *
                    ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                    ↪  dISH_phi
```

```
585
586
587                         z1 += (l+1)*A[k]*np.cos(coord_table[j1,i,j,0]) *
                            ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_r
588                         z1 += (l+1)*A[k]*np.cos(image_coord_table[j1,i,j,0]) *
                            ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                            ↪  dISH_r
589
590                         z1 += A[k]*np.sin(coord_table[j1,i,j,0]) *
                            ↪  coord_table[j1,i,j,3]**(-l-2)*dSH_theta
591                         z1 += A[k]*np.sin(image_coord_table[j1,i,j,0]) *
                            ↪  (-1)**(l+m)*beta*image_coord_table[j1,i,j,3]**(-l-2) *
                            ↪  dISH_theta
592
593                 else:
594                     x1 += np.sin(coord_table[origin,i,j,0]) *
                        ↪  np.cos(coord_table[origin,i,j,1]) * (b[0]*dSH0_r +
                        ↪  b[1]*dSH1_r*epsilon_plus/epsilon_minus + b[2]*dSH2_r)
595                     x1 += np.cos(coord_table[origin,i,j,0]) *
                        ↪  np.cos(coord_table[origin,i,j,1]) * (b[0]*dSH0_theta +
                        ↪  b[1]*dSH1_theta*epsilon_plus/epsilon_minus +
                        ↪  b[2]*dSH2_theta)
596                     x1 += -1*np.sin(coord_table[origin,i,j,1]) /
                        ↪  np.sin(coord_table[origin,i,j,0]) * (b[0]*dSH0_phi +
                        ↪  b[2]*dSH2_phi)
597
598                     y1 += np.sin(coord_table[origin,i,j,0]) *
                        ↪  np.sin(coord_table[origin,i,j,1]) * (b[0]*dSH0_r +
                        ↪  b[1]*dSH1_r*epsilon_plus/epsilon_minus + b[2]*dSH2_r)
599                     y1 += np.cos(coord_table[origin,i,j,0]) *
                        ↪  np.sin(coord_table[origin,i,j,1]) * (b[0]*dSH0_theta +
                        ↪  b[1]*dSH1_theta*epsilon_plus/epsilon_minus +
                        ↪  b[2]*dSH2_theta)
600                     y1 += np.cos(coord_table[origin,i,j,1]) /
                        ↪  np.sin(coord_table[origin,i,j,0]) * (b[0]*dSH0_phi +
                        ↪  b[2]*dSH2_phi)
601
602                     z1 += np.cos(coord_table[origin,i,j,0]) * (b[0]*dSH0_r +
                        ↪  b[1]*dSH1_r*epsilon_plus/epsilon_minus + b[2]*dSH2_r)
603                     z1 += -1*np.sin(coord_table[origin,i,j,0]) * (b[0]*dSH0_theta +
                        ↪  b[1]*dSH1_theta*epsilon_plus/epsilon_minus +
                        ↪  b[2]*dSH2_theta)
604                     for k in range(N*M):
605                         j1, k1 = k//M, k%M
606                         l = get_l(k1)
607                         m = get_m(k1)
608                         ind = int(l*(l+1)/2+np.abs(m))
609                         dSH_theta = 0
```

```
                    dSH_phi = 0
                    dSH_r = 0
                    if m >= 0:
                        dSH_theta += m*spherical_harmonics[j1,i,j,ind] /
                        ↪  np.tan(coord_table[j1,i,j,0])
                        dSH_phi += 1j*m*spherical_harmonics[j1,i,j,ind]
                        dSH_r += spherical_harmonics[j1,i,j,ind]
                        if m != l:
                            dSH_theta += np.sqrt((l-m)*(l+m+1)) *
                            ↪  np.exp(-1j*coord_table[j1,i,j,1]) *
                            ↪  spherical_harmonics[j1,i,j,ind+1]
                    else:
                        dSH_theta += (-1)**m*m *
                        ↪  np.conj(spherical_harmonics[j1,i,j,ind]) /
                        ↪  np.tan(coord_table[j1,i,j,0]) +
                        ↪  (-1)**m*np.sqrt((l-m)*(l+m+1)) *
                        ↪  np.exp(-1j*coord_table[j1,i,j,1]) *
                        ↪  np.conj(spherical_harmonics[j1,i,j,ind-1])
                        dSH_phi += (-1)**m*1j*m *
                        ↪  np.conj(spherical_harmonics[j1,i,j,ind])
                        dSH_r += (-1)**m *
                        ↪  np.conj(spherical_harmonics[j1,i,j,ind])

                    x1 += (l+1)*beta2*A[k] * np.sin(coord_table[j1,i,j,0]) *
                    ↪  np.cos(coord_table[j1,i,j,1]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_r)
                    x1 += -beta2*A[k] * np.cos(coord_table[j1,i,j,0]) *
                    ↪  np.cos(coord_table[j1,i,j,1]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_theta)
                    x1 += beta2*A[k] * np.sin(coord_table[j1,i,j,1]) /
                    ↪  np.sin(coord_table[j1,i,j,0]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_phi)

                    y1 += (l+1)*beta2*A[k] * np.sin(coord_table[j1,i,j,0]) *
                    ↪  np.sin(coord_table[j1,i,j,1]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_r)
                    y1 += -beta2*A[k] * np.cos(coord_table[j1,i,j,0]) *
                    ↪  np.sin(coord_table[j1,i,j,1]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_theta)
                    y1 += -beta2*A[k] * np.cos(coord_table[j1,i,j,1]) /
                    ↪  np.sin(coord_table[j1,i,j,0]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_phi)

                    z1 += (l+1)*beta2*A[k] * np.cos(coord_table[j1,i,j,0]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_r)
                    z1 += beta2*A[k] * np.sin(coord_table[j1,i,j,0]) *
                    ↪  (coord_table[j1,i,j,3]**(-l-2)*dSH_theta)
```

```
634             else:
635                 for k in range(M):
636                     l = get_l(k)
637                     m = get_m(k)
638                     ind = int(l*(l+1)/2+np.abs(m))
639                     dSH_theta = 0
640                     dSH_phi = 0
641                     dSH_r = 0
642                     if m >= 0:
643                         dSH_theta += m*spherical_harmonics[sphere,i,j,ind] /
                            ↪  np.tan(coord_table[sphere,i,j,0])
644                         dSH_phi += 1j*m*spherical_harmonics[sphere,i,j,ind]
645                         dSH_r += spherical_harmonics[sphere,i,j,ind]
646                         if m != l:
647                             dSH_theta += np.sqrt((l-m)*(l+m+1)) *
                                ↪  np.exp(-1j*coord_table[sphere,i,j,1]) *
                                ↪  spherical_harmonics[sphere,i,j,ind+1]
648                     else:
649                         dSH_theta += (-1)**m*m *
                            ↪  np.conj(spherical_harmonics[sphere,i,j,ind]) /
                            ↪  np.tan(coord_table[sphere,i,j,0]) +
                            ↪  (-1)**m*np.sqrt((l-m)*(l+m+1)) *
                            ↪  np.exp(-1j*coord_table[sphere,i,j,1]) *
                            ↪  np.conj(spherical_harmonics[sphere,i,j,ind-1])
650                         dSH_phi += (-1)**m*1j*m *
                            ↪  np.conj(spherical_harmonics[sphere,i,j,ind])
651                         dSH_r += (-1)**m *
                            ↪  np.conj(spherical_harmonics[sphere,i,j,ind])
652
653                     x1 += -l*B[sphere*M+k] * coord_table[sphere,i,j,3]**(l-1)*dSH_r
654                     y1 += -B[sphere*M+k] *
                        ↪  coord_table[sphere,i,j,3]**(l-1)*dSH_theta
655                     z1 += -B[sphere*M+k] * coord_table[sphere,i,j,3]**(l-1)*dSH_phi
                        ↪  / np.sin(coord_table[sphere,i,j,0])
656
657             E[i, j] = np.sqrt(x1*np.conj(x1) + y1*np.conj(y1) + z1*np.conj(z1))
658     return E
659
660 # ------------------------------------------------------------------------
661 # --- end implementing functions
662 # ------------------------------------------------------------------------
663
664 # ========================================================================
665 #  Main
666 # ========================================================================
667
668 #Returns a dictionary with the elements specified by the argument output.
669 #Parameters:
```

```
670 #
671 #output - array or list of strings to be used as dictionary keys
672 #for the output dictionary. Possible options are:
673 #"p_x" is the dimensinoless dipole moment with incident electric field
674 #parallel to the x-axis. Similarly for "p_y" and "p_z". "p_c" has an
675 #incident field spesified by the arguments theta_0 and phi_0. The
676 #dipole moments have dimensions [N_cell, N_ohm].
677 #Likewise for the potentials "pot_x, "pot_y", "pot_z" and "pot_c",
678 #and the electric fields "E_x", "E_y", "E_z" and "E_c". The
679 #potentials and fields have dimensions [N_ohm, N_x, N_z].
680 #Lastly, the reflectance "R_s" for s-polarized light and "R_p" for
681 #p_polarised light can be computed.
682 #
683 #omega - array of energies in eV for the incident wave
684 #N_cell - number of spheres in the unit cell
685 #p - list of positions of sphere centres (x,y,z)
686 #r - corresponding relative radiuses for spheres in p
687 #L_parallel - multipole order of sphere interaction
688 #L_orthogonal - multipole order of substrate interaction
689 #epsilon_minus - dielectric function of the substrate.
690 #Can be a string with a material in the SOPRA database,
691 #a single value or an array of values corresponding to the
692 #incident wave energies in omega
693 #
694 #epsilon_plus - dielectric function of the ambient medium.
695 #Must be a single value
696 #
697 #epsilons - dielectric function of the spheres given as
698 #an array of dielectric values corresponding to the
699 #incident wave energies in omega. The dielectric values
700 #can be N_cell long arrays corresponding to the spheres
701 #in p, or a single value which will be applied to
702 #all the spheres.
703 #
704 #material - dielectric function of the spheres given as
705 #a string of the material from the SOPRA database. If
706 #only a single string is present it will be applied for
707 #all the spheres, otherwise an array of N_cell strings
708 #must be used corresponding to each sphere in p
709 #
710 #omega_p and gamma - the two parameters for the Drude
711 #model. If single values, they're applied for all spheres.
712 #Otherwise, an array of N_cell values must be submitted.
713 #
714 #grid_specs - an array of 7 parameters used for the
715 #grid where the potential and electric field is
716 #evaluated. First two are start and end x coordinates.
717 #Next two are start and end z coordinates. Next two
```

```
718 #are number of x and z points. Last is y coordinate.
719 #
720 #origin - which sphere to be used as an origin in
721 #the potential plots
722 #
723 #periodic - whether or not the unit cell shall be
724 #repeated periodically. If True, nearfield calculations
725 #are not calculated. If False, reflectances are not
726 #calculated.
727 #
728 #theta and phi - polar and azimuthal angles of
729 #the incident plane wave.
730 #
731 #theta_0 and phi_0 - polar and azimuthal angles of
732 #the electric field for custom calculations of
733 #dipole moment, potential and electric field
734 #(p_c, pot_c and E_c).
735 #
736 #R_interaction - interaction distance. Only unit cells
737 #within this distance will be included in calculations
738 #
739 #length_scale - conversion for the sphere radiuses in r
740 #to metres
741 #b1_x and b1_y - the two lattice spacings between the
742 #unit cells
743 #
744 #alpha - the angle between b1_x and b1_y
745 #verbose - If computation times shall be printed
746 #freq - Number of energies between each printing
747 #tolerance - tolaerance for the iterative solver
748 def main(output, omega, N_cell, p, r, L_parallel, L_orthogonal, epsilon_minus,
    ↪  epsilon_plus = 1, epsilons = None, material = None, omega_p = None, gamma =
    ↪  None, grid_specs = None, origin = 0, periodic = False, theta = None, phi =
    ↪  None, theta_0 = None, phi_0 = None, R_interaction = None, length_scale = None,
    ↪  b1_x = None, b1_y = None, alpha = np.pi/2, verbose = True, freq = 10, tolerance
    ↪  = 1e-6):
749     start1 = time()
750     start0 = time()
751     L = max(L_parallel, L_orthogonal) #Largest truncation
752     M = (L+1)**2 - 1  #Number of unknowns per sphere
753     N_ohm = len(omega)
754
755     #Control input parameters related to periodic boundary conditions
756     if periodic:
757         if np.any([theta == None, phi == None, R_interaction == None, length_scale
            ↪  == None, b1_x == None, b1_y == None, alpha == None]):
758             print("Error: Some of the parameters theta, phi, R_interaction,
                ↪  length_scale, b1_x, b1_y and alpha are unspecified.")
```

```
759             return {}
760         else:
761             rho = N_cell/(b1_x*b1_y*np.sin(alpha))
762             theta_p, phi_p, theta_s, phi_s = E_angles(theta, phi)
763     else:
764         b1_x = b1_y = 1000
765         alpha = np.pi/2
766         R_interaction = 1
767         length_scale = 1
768         theta = phi = 0
769 
770     #Prepare the dielectric functions
771     if epsilons != None:
772         if np.shape(epsilons) != (N_ohm, N_cell) and np.shape(epsilons) !=
            ↪  (N_ohm,):
773             print("Error: epsilons wrongly defined. Must either have dimensions
                ↪  (N_ohm, N_cell) or (N_ohm,).")
774             return {}
775         elif np.shape(epsilons) != (N_ohm,):
776             epsilons = np.reshape(np.tile(epsilons, N_cell), [N_cell, N_ohm]).T
777     elif material != None:
778         if np.shape(material) == ():
779             SOPRA_obj = EpsilonSOPRA(material)
780             epsilons = SOPRA_obj.getepsilon(omega)
781             epsilons = np.reshape(np.tile(epsilons, N_cell), [N_cell, N_ohm]).T
782         elif np.shape(material) == (N_cell,):
783             epsilons = np.zeros([N_cell, N_ohm], dtype = np.complex128)
784             for i in range(N_cell):
785                 SOPRA_obj = EpsilonSOPRA(material[i])
786                 epsilons[i] = SOPRA_obj.getepsilon(omega)
787             epsilons = epsilons.T
788         else:
789             print("Error: material wrongly defined. Must either be string or
                ↪  list/array of strings.")
790             return {}
791     elif omega_p != None and gamma != None:
792         epsilons = 1 - omega_p**2/(omega*(omega + 1j*gamma))
793         epsilons = np.reshape(np.tile(epsilons, N_cell), [N_cell, N_ohm]).T
794     else:
795         print("Error: Dielectric function must be defined either by epsilons,
            ↪  material or omega_p and gamma.")
796         return {}
797 
798     if type(epsilon_minus) == str:
799         SOPRA_obj = EpsilonSOPRA(epsilon_minus)
800         epsilon_minus = SOPRA_obj.getepsilon(omega)
801     else:
802         if type(epsilon_minus) != np.ndarray:
```

```
803             epsilon_minus = np.array([epsilon_minus]).flatten()
804         if len(epsilon_minus) != N_ohm and len(epsilon_minus) != 1:
805             print("Error: epsilon_minus must either be  a string of a material, a
                ↪  number or a list/array of numbers with length N_ohm.")
806             return {}
807         elif len(epsilon_minus) == 1:
808             epsilon_minus = epsilon_minus*np.ones(N_ohm, dtype = np.complex128)
809 
810     beta = (epsilon_plus - epsilon_minus)/(epsilon_plus + epsilon_minus)
811     beta2 = 2*epsilon_plus/(epsilon_plus + epsilon_minus)
812 
813     #Allocate memory
814     contours = np.array(["pot_x", "pot_y", "pot_z", "pot_c", "E_x", "E_y", "E_z",
        ↪  "E_c"])
815     if np.any(np.isin(output, contours)):
816         if not periodic:
817             if grid_specs != None:
818                 if L > 42 and grid_specs[4]*grid_specs[5] > 1e4:
819                     print("Runtime warning: Nearfield calculations with high
                        ↪  resolution has long initialisation time for the spherical
                        ↪  harmonics.")
820 
821                 table = np.zeros([N_cell, grid_specs[5], grid_specs[4], 5], dtype =
                    ↪  np.complex128)
822                 x, z, grid_coords, grid_image_coords = coord_table(table, p,
                    ↪  N_cell, grid_specs[0], grid_specs[1], grid_specs[2],
                    ↪  grid_specs[3], grid_specs[4], grid_specs[5], grid_specs[6])
823                 if L < 43:
824                     grid_spherical_harmonics, grid_image_spherical_harmonics =
                        ↪  spherical_harm(np.zeros([N_cell, grid_specs[5],
                        ↪  grid_specs[4], (L+2)*(L+1)//2], dtype = np.complex128),
                        ↪  np.real(grid_coords).astype(np.float64),
                        ↪  np.real(grid_image_coords).astype(np.float64), L, N_cell,
                        ↪  grid_specs[4], grid_specs[5])
825                 else:
826                     grid_spherical_harmonics, grid_image_spherical_harmonics =
                        ↪  spherical_harm2(grid_coords, grid_image_coords, L, N_cell,
                        ↪  grid_specs[4], grid_specs[5])
827                 H0s = H1(np.arange(M))
828 
829                 if np.isin("pot_x", output):
830                     pot_x = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
831                 if np.isin("pot_y", output):
832                     pot_y = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
833                 if np.isin("pot_z", output):
```

```
834                     pot_z = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
835                 if np.isin("pot_c", output):
836                     pot_c = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
837
838                 if np.isin("E_x", output):
839                     E_x = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
840                 if np.isin("E_y", output):
841                     E_y = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
842                 if np.isin("E_z", output):
843                     E_z = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
844                 if np.isin("E_c", output):
845                     E_c = np.zeros([N_ohm, grid_specs[5], grid_specs[4]], dtype =
                        ↪  np.complex128)
846             else:
847                 print("Error: grid_specs undefined")
848                 return {}
849         else:
850             print("Error: Nearfield calculations only works for finite systems in
                ↪  this code. Set finite to True.")
851             output = np.setdiff1d(output, contours)
852
853     dipole_moments = np.array(["p_x", "p_y", "p_z", "p_c"])
854     if np.any(np.isin(output, dipole_moments)):
855         if np.isin("p_x", output):
856             p_x = np.zeros([N_cell, N_ohm])
857         if np.isin("p_y", output):
858             p_y = np.zeros([N_cell, N_ohm])
859         if np.isin("p_z", output):
860             p_z = np.zeros([N_cell, N_ohm])
861         if np.isin("p_c", output):
862             p_c = np.zeros([N_cell, N_ohm])
863
864     reflectances = np.array(["R_s", "R_p"])
865     if np.any(np.isin(output, reflectances)):
866         if periodic:
867             if np.isin("R_p", output):
868                 R_p = np.zeros(N_ohm, dtype = np.complex128)
869             if np.isin("R_s", output):
870                 R_s = np.zeros(N_ohm, dtype = np.complex128)
871         else:
872             print("Error: The calculations must be done for a periodic infinite
                ↪  system in order to determine reflectances. Set 'finite' to False.")
873             output = np.setdiff1d(output, reflectances)
```

```
874
875     computations = np.array(["p_x", "p_y", "p_z", "p_c", "R_s", "R_p", "pot_x",
        ↪  "pot_y", "pot_z", "pot_c", "E_x", "E_y", "E_z", "E_c"])
876     if not np.any(np.isin(output, computations)):
877         print("No optional computations specified")
878         return {}
879
880
881     #Compute values independent of the dielectric functions of the spheres:
882     #These are faster to just look up than to compute each time, and do
883     #only have to be computed once. Numba is also not as cooperative with
884     #pyshtools and binomial coefficients.
885
886     #coords[i, j, p, k] is the kth coordinate from get_polar_coords
887     #of sphere j  at lattice point p relative to sphere i in centre
888     #lattice point. The lattice point's corrdinates are retrieved by
889     #lattice_vectors[p].
890
891     #spherical_harmonics[i, j, p] is an array of all the spherical harmonics
892     #for sphere j at lattice point p relative to sphere i in the centre
893     #lattice point. This is on compact form so the
894     #elements are indexed by ind = int(l*(l+1)/2+np.abs(m)),
895     #such that only for m>=0 are contained.
896
897     lattice_points = get_length(R_interaction, b1_x, b1_y, alpha)
898     coords = np.zeros([N_cell, N_cell, lattice_points, 5])
899     spherical_harmonics = np.zeros([N_cell, N_cell, lattice_points,
        ↪  (2*L+2)*(2*L+1)//2], dtype = np.complex128)
900     image_spherical_harmonics = np.zeros([N_cell, N_cell, lattice_points,
        ↪  (2*L+2)*(2*L+1)//2], dtype = np.complex128)
901     lattice_vectors = np.zeros([lattice_points, 2])
902
903     lattice_point = 0
904     centre_point = 0
905     j1s = get_j1(R_interaction, b1_y, alpha)
906     for j1 in j1s:
907         i1s = get_i1(R_interaction, b1_x, b1_y, alpha, j1)
908         for i1 in i1s:
909             lattice_vectors[lattice_point] = np.array([i1*b1_x +
                ↪  j1*b1_y*np.cos(alpha), j1*b1_y*np.sin(alpha)])*r[0]
910             if j1 == 0 and i1 == 0:
911                 centre_point = lattice_point
912             for j in range(N_cell):
913                 for i in range(N_cell):
914                     coords[j,i,lattice_point] = get_spherical_coords(p[j],
                        ↪  p[i]+np.concatenate((lattice_vectors[lattice_point],
                        ↪  np.array([0]))))
```

```
                    spherical_harmonics[j,i,lattice_point] = expand.spharm(2*L,
                    ↪  coords[j,i,lattice_point,0], coords[j,i,lattice_point,1],
                    ↪  normalization = 'ortho', kind = 'complex', csphase = -1,
                    ↪  packed = True, degrees = False)[0]
                    image_spherical_harmonics[j,i,lattice_point] =
                    ↪  expand.spharm(2*L, coords[j,i,lattice_point,2],
                    ↪  coords[j,i,lattice_point,1], normalization = 'ortho', kind
                    ↪  = 'complex', csphase = -1, packed = True, degrees =
                    ↪  False)[0]
            lattice_point += 1

    #Compute H(l_j, m_j | l_i, m_i) for all the M*M outcomes
    #and store them in a table indexed by k and l
    Hs = np.reshape(H(np.arange(M**2), M), [M, M])


    end0 = time()
    if verbose:
        print("Initialization complete. Excecution time taken: ", end0-start0)
        print("Frequency number:  Excecution time:  Expected excecution time [hrs]:
        ↪  ")

    #First energy, LU-factorisation is used to solve for A
    k = get_k(omega[0], theta, phi, length_scale)
    C = np.zeros([M*N_cell, M*N_cell], dtype = np.complex128)
    C = get_C(C, L_parallel, L_orthogonal, M, N_cell, lattice_points, centre_point,
    ↪  lattice_vectors, epsilons[0], r, beta[0], epsilon_plus, coords,
    ↪  spherical_harmonics, image_spherical_harmonics, Hs, k, b1_x, b1_y, alpha)


    x_computations = np.array(["p_x", "pot_x", "E_x"])
    x_nearfields = np.array(["pot_x", "E_x"])
    if np.any(np.isin(output, x_computations)):
        b_x = get_b(np.pi/2, 0, M, N_cell)
        A_x = np.linalg.solve(C, b_x)
        A_x_prev = A_x.copy()

        if np.isin("p_x", output):
            for n in range(N_cell):
                p_x[n,0] = get_p(A_x, n, M, r)

        if np.any(np.isin(x_nearfields, output)):
            B_x = get_B(M, N_cell, A_x, b_x, beta[0], r, Hs,
            ↪  coords[:,:,centre_point,:], spherical_harmonics[:,:,centre_point],
            ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
            ↪  L_orthogonal)
            if np.isin("pot_x", output):
```

```
949                 B0_x = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                    ↪  N_cell, A_x, beta[0], r, coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_x)
950                 pot_x[0] = potential_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords, grid_image_coords,
                    ↪  grid_spherical_harmonics, grid_image_spherical_harmonics, A_x,
                    ↪  r, N_cell, M, beta[0], b_x, p, B_x, beta2[0], epsilon_plus,
                    ↪  epsilon_minus[0], B0_x, np.pi/2)
951             if np.isin("E_x", output):
952                 E_x[0] = field_grid(np.zeros([grid_specs[5], grid_specs[4]], dtype
                    ↪  = np.complex128), origin, x, grid_specs[6], z, grid_specs[5],
                    ↪  grid_specs[4], grid_coords, grid_image_coords,
                    ↪  grid_spherical_harmonics, grid_image_spherical_harmonics, A_x,
                    ↪  r, N_cell, M, beta[0], b_x, p, B_x, beta2[0], epsilon_plus,
                    ↪  epsilon_minus[0])
953
954
955     y_computations = np.array(["p_y", "pot_y", "E_y"])
956     y_nearfields = np.array(["pot_y", "E_y"])
957     if np.any(np.isin(output, y_computations)):
958         b_y = get_b(np.pi/2, np.pi/2, M, N_cell)
959         A_y = np.linalg.solve(C, b_y)
960         A_y_prev = A_y.copy()
961
962         if np.isin("p_y", output):
963             for n in range(N_cell):
964                 p_y[n,0] = get_p(A_y, n, M, r)
965
966         if np.any(np.isin(y_nearfields, output)):
967             B_y = get_B(M, N_cell, A_y, b_y, beta[0], r, Hs,
                    ↪  coords[:,:,centre_point,:], spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal)
968             if np.isin("pot_y", output):
969                 B0_y = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                    ↪  N_cell, A_y, beta[0], r, coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_y)
970                 pot_y[0] = potential_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords, grid_image_coords,
                    ↪  grid_spherical_harmonics, grid_image_spherical_harmonics, A_y,
                    ↪  r, N_cell, M, beta[0], b_y, p, B_y, beta2[0], epsilon_plus,
                    ↪  epsilon_minus[0], B0_y, np.pi/2)
```

```
971             if np.isin("E_x", output):
972                 E_y[0] = field_grid(np.zeros([grid_specs[5], grid_specs[4]], dtype
                    ↪  = np.complex128), origin, x, grid_specs[6], z, grid_specs[5],
                    ↪  grid_specs[4], grid_coords, grid_image_coords,
                    ↪  grid_spherical_harmonics, grid_image_spherical_harmonics, A_y,
                    ↪  r, N_cell, M, beta[0], b_y, p, B_y, beta2[0], epsilon_plus,
                    ↪  epsilon_minus[0])
973
974
975     z_computations = np.array(["p_z", "pot_z", "E_z"])
976     z_nearfields = np.array(["pot_z", "E_z"])
977     if np.any(np.isin(output, z_computations)):
978         b_z = get_b(np.pi, 0, M, N_cell)
979         A_z = np.linalg.solve(C, b_z)
980         A_z_prev = A_z.copy()
981
982         if np.isin("p_z", output):
983             for n in range(N_cell):
984                 p_z[n,0] = get_p(A_z, n, M, r)
985
986         if np.any(np.isin(z_nearfields, output)):
987             B_z = get_B(M, N_cell, A_z, b_z, beta[0], r, Hs,
                    ↪  coords[:,:,centre_point,:], spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal)
988             if np.isin("pot_z", output):
989                 B0_z = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                        ↪  N_cell, A_z, beta[0], r, coords[:,:,centre_point,:],
                        ↪  spherical_harmonics[:,:,centre_point],
                        ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                        ↪  L_orthogonal, origin, b_z)
990                 pot_z[0] = potential_grid(np.zeros([grid_specs[5], grid_specs[4]],
                        ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                        ↪  grid_specs[5], grid_specs[4], grid_coords, grid_image_coords,
                        ↪  grid_spherical_harmonics, grid_image_spherical_harmonics, A_z,
                        ↪  r, N_cell, M, beta[0], b_z, p, B_z, beta2[0], epsilon_plus,
                        ↪  epsilon_minus[0], B0_z, np.pi)
991             if np.isin("E_z", output):
992                 E_z[0] = field_grid(np.zeros([grid_specs[5], grid_specs[4]], dtype
                    ↪  = np.complex128), origin, x, grid_specs[6], z, grid_specs[5],
                    ↪  grid_specs[4], grid_coords, grid_image_coords,
                    ↪  grid_spherical_harmonics, grid_image_spherical_harmonics, A_z,
                    ↪  r, N_cell, M, beta[0], b_z, p, B_z, beta2[0], epsilon_plus,
                    ↪  epsilon_minus[0])
993
994
995     c_computations = np.array(["p_c", "pot_c", "E_c"])
996     c_nearfields = np.array(["pot_c", "E_c"])
```

```
    if np.any(np.isin(output, c_computations)):
        if theta_0 != None and phi_0 != None:
            b_c = get_b(theta_0, phi_0, M, N_cell) #Build B-vector
            A_c = np.linalg.solve(C, b_c) #Solve for A
            A_c_prev = A_c.copy() #Copy solution to start iteration

            if np.isin("p_c", output):
                for n in range(N_cell): #Find dipole moment of sphere n
                    p_c[n,0] = get_p(A_c, n, M, r)

            if np.any(np.isin(c_nearfields, output)):
                B_c = get_B(M, N_cell, A_c, b_c, beta[0], r, Hs,
                ↪  coords[:,:,centre_point,:],
                ↪  spherical_harmonics[:,:,centre_point],
                ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                ↪  L_orthogonal)
                if np.isin("pot_c", output):
                    B0_c = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                    ↪  N_cell, A_c, beta[0], r, coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_c)
                    pot_c[0] = potential_grid(np.zeros([grid_specs[5],
                    ↪  grid_specs[4]], dtype = np.complex128), origin, x,
                    ↪  grid_specs[6], z, grid_specs[5], grid_specs[4],
                    ↪  grid_coords, grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_c, r, N_cell, M, beta[0],
                    ↪  b_c, p, B_c, beta2[0], epsilon_plus, epsilon_minus[0],
                    ↪  B0_c, theta_0)
                if np.isin("E_c", output):
                    E_c[0] = field_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords,
                    ↪  grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_c, r, N_cell, M, beta[0],
                    ↪  b_c, p, B_c, beta2[0], epsilon_plus, epsilon_minus[0])

        else:
            print("Error: theta_0 and phi_0 must be specified for custom angle
            ↪  input computations.")
            output = np.setdiff1d(output, c_computations)


    if np.any(np.isin(output, reflectances)):
        if np.isin("R_s", output):
            b_s = get_b(theta_s, phi_s, M, N_cell)
            A_s = np.linalg.solve(C, b_s)
            A_s_prev = A_s.copy()
```

```
            gamma_s, beta_s = susceptibilities(A_s, M, theta_s, phi_s,
            ↪  epsilon_plus, p[0,2], rho)
            R_s[0] = s_polarization(theta, omega[0], gamma_s, epsilon_plus,
            ↪  epsilon_minus[0], length_scale)[0]

        if np.isin("R_p", output):
            b_p2 = get_b(theta_p, phi_p, M, N_cell)
            A_p2 = np.linalg.solve(C, b_p2)
            A_p2_prev = A_p2.copy()
            gamma_p, beta_p = susceptibilities(A_p2, M, theta_p, phi_p,
            ↪  epsilon_plus, p[0,2], rho)
            R_p[0] = p_polarization(theta, omega[0], gamma_p, epsilon_plus,
            ↪  epsilon_minus[0], beta_p, length_scale)[0]


    #For the rest of the energies an iterative solver is used
    for o in range(1, N_ohm):
        start2 = time()
        k = get_k(omega[o], theta, phi, length_scale)
        C = np.zeros([M*N_cell, M*N_cell], dtype = np.complex128)
        C = get_C(C, L_parallel, L_orthogonal, M, N_cell, lattice_points,
        ↪  centre_point, lattice_vectors, epsilons[o], r, beta[o], epsilon_plus,
        ↪  coords, spherical_harmonics, image_spherical_harmonics, Hs, k, b1_x,
        ↪  b1_y, alpha)


        if np.any(np.isin(output, x_computations)):
            A_x, info = gmres(C, b_x, A_x_prev, tol = tolerance)
            A_x_prev = A_x.copy()

            if np.isin("p_x", output):
                for n in range(N_cell):
                    p_x[n,o] = get_p(A_x, n, M, r)

            if np.any(np.isin(x_nearfields, output)):
                B_x = get_B(M, N_cell, A_x, b_x, beta[o], r, Hs,
                ↪  coords[:,:,centre_point,:],
                ↪  spherical_harmonics[:,:,centre_point],
                ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                ↪  L_orthogonal)
                if np.isin("pot_x", output):
                    B0_x = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                    ↪  N_cell, A_x, beta[o], r, coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_x)
```

```
                    pot_x[o] = potential_grid(np.zeros([grid_specs[5],
                    ↪  grid_specs[4]], dtype = np.complex128), origin, x,
                    ↪  grid_specs[6], z, grid_specs[5], grid_specs[4],
                    ↪  grid_coords, grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_x, r, N_cell, M, beta[o],
                    ↪  b_x, p, B_x, beta2[o], epsilon_plus, epsilon_minus[o],
                    ↪  B0_x, np.pi/2)
                if np.isin("E_x", output):
                    E_x[o] = field_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords,
                    ↪  grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_x, r, N_cell, M, beta[o],
                    ↪  b_x, p, B_x, beta2[o], epsilon_plus, epsilon_minus[o])


        if np.any(np.isin(output, y_computations)):
            A_y, info = gmres(C, b_y, A_y_prev, tol = tolerance)
            A_y_prev = A_y.copy()

            if np.isin("p_y", output):
                for n in range(N_cell):
                    p_y[n,o] = get_p(A_y, n, M, r)

            if np.any(np.isin(y_nearfields, output)):
                B_y = get_B(M, N_cell, A_y, b_y, beta[o], r, Hs,
                ↪  coords[:,:,centre_point,:],
                ↪  spherical_harmonics[:,:,centre_point],
                ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                ↪  L_orthogonal)
                if np.isin("pot_y", output):
                    B0_y = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                    ↪  N_cell, A_y, beta[o], r, coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_y)
                    pot_y[o] = potential_grid(np.zeros([grid_specs[5],
                    ↪  grid_specs[4]], dtype = np.complex128), origin, x,
                    ↪  grid_specs[6], z, grid_specs[5], grid_specs[4],
                    ↪  grid_coords, grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_y, r, N_cell, M, beta[o],
                    ↪  b_y, p, B_y, beta2[o], epsilon_plus, epsilon_minus[o],
                    ↪  B0_y, np.pi/2)
                if np.isin("E_x", output):
```

```
                    E_y[o] = field_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords,
                    ↪  grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_y, r, N_cell, M, beta[o],
                    ↪  b_y, p, B_y, beta2[o], epsilon_plus, epsilon_minus[o])


        if np.any(np.isin(output, z_computations)):
            A_z, info = gmres(C, b_z, A_z_prev, tol = tolerance)
            A_z_prev = A_z.copy()

            if np.isin("p_z", output):
                for n in range(N_cell):
                    p_z[n,o] = get_p(A_z, n, M, r)

            if np.any(np.isin(z_nearfields, output)):
                B_z = get_B(M, N_cell, A_z, b_z, beta[o], r, Hs,
                ↪  coords[:,:,centre_point,:],
                ↪  spherical_harmonics[:,:,centre_point],
                ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                ↪  L_orthogonal)
                if np.isin("pot_z", output):
                    B0_z = get_B_0(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                    ↪  N_cell, A_z, beta[o], r, coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_z)
                    pot_z[o] = potential_grid(np.zeros([grid_specs[5],
                    ↪  grid_specs[4]], dtype = np.complex128), origin, x,
                    ↪  grid_specs[6], z, grid_specs[5], grid_specs[4],
                    ↪  grid_coords, grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_z, r, N_cell, M, beta[o],
                    ↪  b_z, p, B_z, beta2[o], epsilon_plus, epsilon_minus[o],
                    ↪  B0_z, np.pi)
                if np.isin("E_z", output):
                    E_z[o] = field_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords,
                    ↪  grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_z, r, N_cell, M, beta[o],
                    ↪  b_z, p, B_z, beta2[o], epsilon_plus, epsilon_minus[o])


        if np.any(np.isin(output, c_computations)):
            A_c, info = gmres(C, b_c, A_c_prev, tol = tolerance)
            A_c_prev = A_c.copy()

```

```
            if np.isin("p_c", output):
                for n in range(N_cell):
                    p_c[n,o] = get_p(A_c, n, M, r)

            if np.any(np.isin(c_nearfields, output)):
                B_c = get_B(np.zeros(N_cell, dtype = np.complex128), H0s, M,
                ↪  N_cell, A_c, b_c, beta[o], r, Hs, coords[:,:,centre_point,:],
                ↪  spherical_harmonics[:,:,centre_point],
                ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                ↪  L_orthogonal)
                if np.isin("pot_c", output):
                    B0_c = get_B_0(M, N_cell, A_c, beta[o], r,
                    ↪  coords[:,:,centre_point,:],
                    ↪  spherical_harmonics[:,:,centre_point],
                    ↪  image_spherical_harmonics[:,:,centre_point], L_parallel,
                    ↪  L_orthogonal, origin, b_c)
                    pot_c[o] = potential_grid(np.zeros([grid_specs[5],
                    ↪  grid_specs[4]], dtype = np.complex128), origin, x,
                    ↪  grid_specs[6], z, grid_specs[5], grid_specs[4],
                    ↪  grid_coords, grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_c, r, N_cell, M, beta[o],
                    ↪  b_c, p, B_c, beta2[o], epsilon_plus, epsilon_minus[o],
                    ↪  B0_c, theta_0)
                if np.isin("E_c", output):
                    E_c[o] = field_grid(np.zeros([grid_specs[5], grid_specs[4]],
                    ↪  dtype = np.complex128), origin, x, grid_specs[6], z,
                    ↪  grid_specs[5], grid_specs[4], grid_coords,
                    ↪  grid_image_coords, grid_spherical_harmonics,
                    ↪  grid_image_spherical_harmonics, A_c, r, N_cell, M, beta[o],
                    ↪  b_c, p, B_c, beta2[o], epsilon_plus, epsilon_minus[o])


        if np.any(np.isin(output, reflectances)):
            if np.isin("R_s", output):
                A_s, info = gmres(C, b_s, A_s_prev, tol = tolerance)
                A_s_prev = A_s.copy()
                gamma_s, beta_s = susceptibilities(A_s, M, theta_s, phi_s,
                ↪  epsilon_plus, p[0,2], rho)
                R_s[o] = s_polarization(theta, omega[o], gamma_s, epsilon_plus,
                ↪  epsilon_minus[o], length_scale)[0]

            if np.isin("R_p", output):
                A_p2, info = gmres(C, b_p2, A_p2_prev, tol = tolerance)
                A_p2_prev = A_p2.copy()
                gamma_p, beta_p = susceptibilities(A_p2, M, theta_p, phi_p,
                ↪  epsilon_plus, p[0,2], rho)
                R_p[o] = p_polarization(theta, omega[o], gamma_p, epsilon_plus,
                ↪  epsilon_minus[o], beta_p, length_scale)[0]
```

```
1124
1125
1126            end2 = time()
1127            if verbose and o % freq == 0:
1128                print(o, end2-start2, (end2-start2)*N_ohm/3600)
1129
1130
1131        #Creating the output dictionary
1132        dict_out = {}
1133
1134        if np.any(np.isin(output, contours)):
1135            if np.isin("pot_x", output):
1136                dict_out["pot_x"] = pot_x
1137            if np.isin("pot_y", output):
1138                dict_out["pot_y"] = pot_y
1139            if np.isin("pot_z", output):
1140                dict_out["pot_z"] = pot_z
1141            if np.isin("pot_c", output):
1142                dict_out["pot_c"] = pot_c
1143
1144            if np.isin("E_x", output):
1145                dict_out["E_x"] = E_x
1146            if np.isin("E_y", output):
1147                dict_out["E_y"] = E_y
1148            if np.isin("E_z", output):
1149                dict_out["E_z"] = E_z
1150            if np.isin("E_c", output):
1151                dict_out["E_c"] = E_c
1152
1153            if np.isin("x", output):
1154                dict_out["x"] = x
1155            if np.isin("z", output):
1156                dict_out["z"] = z
1157
1158        if np.any(np.isin(output, dipole_moments)):
1159            if np.isin("p_x", output):
1160                dict_out["p_x"] = p_x
1161            if np.isin("p_y", output):
1162                dict_out["p_y"] = p_y
1163            if np.isin("p_z", output):
1164                dict_out["p_z"] = p_z
1165            if np.isin("p_c", output):
1166                dict_out["p_c"] = p_c
1167
1168        if np.any(np.isin(output, reflectances)):
1169            if np.isin("R_p", output):
1170                dict_out["R_p"] = np.absolute(R_p)**2
1171            if np.isin("R_s", output):
```

```
            dict_out["R_s"] = np.absolute(R_s)**2

    if np.isin("epsilon",output):
        dict_out["epsilon"] = epsilons

    end1 = time()
    if verbose:
        print("Task complete: ", end1-start1)
    return dict_out

# ------------------------------------------------------------------------
# --- end main
# ------------------------------------------------------------------------

# ========================================================================
#  Examples
# ========================================================================

h = 0.05
d = 0.1
N_cell = 2
r = np.ones(N_cell)
p2 = np.array([[0,0,1+h], [2+d,0,1+h]], dtype = np.float64)
omega = np.linspace(2.6, 3.8, 121)
omega2 = np.linspace(1.43, 1.43, 1)
gridspecs = (-1.4499, 3.5501, -1.4499, 3.5501, 301, 301, 0)

output1 = np.array(["R_s", "R_p"])
output2 = np.array(["pot_z"])
out1 = main(output1, omega, 1, p2, r, 10, 10, epsilon_minus = 2.76, material =
↪  "ag", periodic = True, theta = np.pi/4, phi = 0, R_interaction = 30,
↪  length_scale = 1e-8, b1_x = 2.2, b1_y = 2.2)
out2 = main(output2, omega2, N_cell, p2, r, 30, 30, epsilon_minus = 10, omega_p =
↪  3, gamma = 0.03, periodic = False, grid_specs = gridspecs)

# ------------------------------------------------------------------------
# --- end examples
# ------------------------------------------------------------------------
```

A.2 Dielectric file for Ag in SOPRA database

```
1          .6          6.6          120
1.064      14.4
.987875    13.155
.844       12.2
.672625    11.445
.514       10.8
.438563    10.11437
.396       9.48
.36        8.95375
.329       8.49
.287375    8.060625
.251       7.67
.235375    7.31375
.226       6.99
.212625    6.6975
.198       6.43
.179875    6.183125
.163       5.95
.151938    5.720624
.145       5.5
.142563    5.28875
.143       5.09
.145875    4.908125
.148       4.74
.144313    4.58625
.14        4.44
.140063    4.293125
.14        4.15
.136125    4.010625
.131       3.88
.
.
.
```

Appendix B

DDSCAT

B.1 Python script for processing results

```
import numpy as np
from matplotlib import pyplot as plt
import os
#Written by Bohren and Huffman, translated by Herbert Kaiser
from BHMIE import bhmie
#python wrapper for SOPRA values
from pysopra import EpsilonSOPRA, micron2eV, eV2micron

#Computes reflectance from a slab of thickness d and refractive index n for
#s- and p-polarised light with wave number k in an ambient medium of refractive
#index n_amb for an array of angles in radians
def Ref_slab(n, n_amb, theta, k, d):
    c_0 = np.cos(theta)
    c_t = np.cos(np.arcsin(n_amb/n*np.sin(theta)))
    pf = np.exp(2*1.j*k*n*d*c_t)

    r1_s = (n*c_t - n_amb*c_0)/(n*c_t + n_amb*c_0)
    r2_s = (n_amb*c_0 - n*c_t)/(n_amb*c_0 + n*c_t)

    r1_p = (n*c_0 - n_amb*c_t)/(n*c_0 + n_amb*c_t)
    r2_p = (n_amb*c_t - n*c_0)/(n_amb*c_t + n*c_0)

    R_s =  (r2_s + r1_s*pf)/(1 + r1_s*r2_s*pf)
    R_p =  (r2_p + r1_p*pf)/(1 + r1_p*r2_p*pf)

    return np.absolute(R_s)**2, np.absolute(R_p)**2

#Computes the absorption efficiency factor from Mie theory for a sphere of
# radius R, dielectric function epsilon and for incident wave energies omega
def sigma_abs_mie(omega, epsilon, R):
    sigmas = np.zeros(len(omega))
    HC = 2 * np.pi * 0.1973269631
```

```
    lamda = HC/omega
    x = 2*np.pi*R/lamda
    for i in range(len(omega)):
        vals = bhmie(x[i], np.sqrt(epsilon[i]), 20)
        sigmas[i] = vals[2]#-vals[3]
    return sigmas

#Creates file with dielectric values compatible for DDCSAT from a SOPRA file.
#The file to read from is filename_SOPRA and the new filename is filename_DDSCAT.
def make_dielectrics_for_DDSCAT(filename_SOPRA, filename_DDSCAT, description):
    header = np.loadtxt(filename_SOPRA, delimiter=None, skiprows=0, max_rows = 1,
    ↪  dtype = str)
    print(header)
    wave = np.linspace(float(header[1]), float(header[2]), int(header[3])+1)
    vals = np.loadtxt(filename_SOPRA, delimiter=None, skiprows=1, dtype =
    ↪  np.float64)
    n_real = vals.T[0]
    n_imag = vals.T[1]
    if int(header[0]) == 1:
        wave = eV2micron(wave[::-1])
        n_real = n_real[::-1]
        n_imag = n_imag[::-1]
    header1 = "1 2 3 0 0 = columns for wave, Re(n), Im(n), eps1, eps2"
    header2 = "wave(um) Re(n)  Im(n)    eps1   eps2"
    file_header = description + "\n" + header1 + "\n" + header2
    np.savetxt(filename_DDSCAT + ".txt", np.array([wave, n_real ,n_imag,
    ↪  np.real(n_real+n_imag*1.0j)**2, np.imag(n_real+n_imag*1.0j)**2]).T,
    ↪  fmt="%s", header = file_header, comments = '')

#Example:
description = "Dielectric values for aluminium oxide from SOPRA database"
make_dielectrics_for_DDSCAT("Database/al2o3.nk", "Al2O3", description)

#Creates an array of N wave energies from DDSCAT output files in directory
def waves(directory, N):
    HC = 2 * np.pi * 0.1973269631
    wave = np.zeros(N)
    for i in range(N):
        num = str(i)
        fname = directory + 'w' + num.zfill(3) + "r000.avg"
        text = float(np.loadtxt(fname, delimiter=None, skiprows=12, max_rows = 1,
        ↪  dtype = str)[1])
        wave[i] = HC/text
    return wave

#Creates an array of N dielectric values from DDSCAT output files in directory
def epsilons(directory, N):
    epsilon = np.zeros(N, dtype = np.complex128)
```

```
76      for i in range(N):
77          num = str(i)
78          fname = directory + 'w' + num.zfill(3) + "r000.avg"
79          text = np.loadtxt(fname, delimiter=None, skiprows=15, max_rows = 1, dtype =
            ↪  str)[[2,4]]
80          re = float(text[0])
81          im = float(text[1][:-2])
82          epsilon[i] = (re+im*1.0j)**2
83      return epsilon
84
85  #Creates a tensor of all the Mueller elements from the DDSCAT outut files.
86  #The function assumes only two Mueller elements are computed (S_11 and S_12).
87  #Tensor has dimensions [2 (transmission and reflection), N_waves wave energies,
88  # and 2 Mueller elements].
89  def Mueller_elements(directory, N_waves):
90      mat = np.zeros([2,N_waves,2])
91      for i in range(N_waves):
92          num = str(i)
93          fname = directory + 'w' + num.zfill(3) + "r000.avg"
94          text = np.loadtxt(fname, delimiter=None, skiprows=35)
95          mat[0,i] = text[0,3:]
96          mat[1,i] = text[1,3:]
97      return mat
98
99  #Finds number of DDSCAT output files in directory
100 def get_N(directory):
101     files = os.listdir(directory)
102     N = 0
103     for file in files:
104         if file[4:8] == "r000" and int(file[1:4]) > N:
105             N = int(file[1:4])
106     return N+1
107
108 #Creates an array of all the wave energies, transmission and reflection Mueller
109 #elements from the DDSCAT output files in directory
110 def get_values(directory):
111     N = get_N(directory)
112     wave = waves(directory, N)
113     M_t, M_r = Mueller_elements(directory, N)
114     return wave, M_t, M_r
115
116 #Creates an array of N_waves absorption efficiency factors from DDSCAT
117 # output files in directory
118 def Q_abs(directory, N_waves):
119     Q = np.zeros(N_waves)
120     for i in range(N_waves):
121         num = str(i)
122         fname = directory + 'w' + num.zfill(3) + "r000k000.sca"
```

```
123         text = np.loadtxt(fname, delimiter=None, skiprows=39, max_rows = 1, dtype =
            ↪  str)
124         Q[i] = float(text[2])
125     return Q
126 
127 #Creates an array of N_waves extinction efficiency factors from DDSCAT
128 #output files in directory
129 def Q_ext(directory, N_waves):
130     Q = np.zeros(N_waves)
131     for i in range(N_waves):
132         num = str(i)
133         fname = directory + 'w' + num.zfill(3) + "r000k000.sca"
134         text = np.loadtxt(fname, delimiter=None, skiprows=39, max_rows = 1, dtype =
            ↪  str)
135         Q[i] = float(text[1])
136     return Q
137 
138 #Creates an array of N_angles angles in radians, a matrix of dimensions
139 #[N_angles, 2 (Mueller elements S_11 and S_12)] and a an array of
140 #refractive inices from DDSCAT output files in directory
141 def substrate(directory, N_angles):
142     mat = np.zeros([N_angles,2])
143     angles = np.zeros(N_angles)
144     n = np.zeros(N_angles, dtype = np.complex128)
145     for i in range(N_angles):
146         num = str(i)
147         fname = directory + 'w000r000k' + num.zfill(3) + ".sca"
148         text = np.loadtxt(fname, delimiter=None, skiprows=40)
149         text2 = float(np.loadtxt(fname, delimiter=None, skiprows=32, max_rows = 1,
            ↪  dtype = str)[1])
150         text3 = np.loadtxt(fname, delimiter=None, skiprows=19, max_rows = 1, dtype
            ↪  = str)[[2,4]]
151         re = float(text3[0])
152         im = float(text3[1][:-2])
153         n[i] = re + 1.0j*im
154         mat[i] = text[1,3:]
155         angles[i] = text2
156     return angles, mat, n
157 
158 #Interchanges x and z coordinates in target file filename in directory and writes
159 #new file to newname in directory
160 def interchange(directory, filename, newname):
161     fname = directory + filename
162     description = ""
163     for i in range(7):
164         description += '\t'.join(np.loadtxt(fname, delimiter=None, skiprows = i,
            ↪  max_rows = 1, dtype = str))
165         if i != 6:
```

```
            description += "\n"
    table = np.loadtxt(fname, delimiter=None, skiprows = 7, dtype = str)
    table = table.T[[0,3,2,1,4,5,6]].T
    np.savetxt(directory + newname, table, fmt='%s', delimiter='\t', newline='\n',
    ↪ header=description, comments='')

#Cuts target file filename in directory in half along x-axis and saves new file
#to newname in directory
def cut_in_half(directory, filename, newname):
    fname = directory + filename
    table = np.loadtxt(fname, delimiter=None, skiprows = 7, dtype = str)
    indices = np.where(table.T[1].astype(np.int) >=
    ↪ int((np.amax(table.T[1].astype(np.int))-np.amin(table.T[1].astype(np.int)))/2))[0]
    table = table[indices]
    N = len(indices)
    table = table.T
    table[0] = np.arange(N).astype(str)
    table = table.T
    description = ""
    for i in range(7):
        string = np.loadtxt(fname, delimiter=None, skiprows = i, max_rows = 1,
        ↪ dtype = str)
        if i == 1:
            string[0] = str(N)
        description += '\t'.join(string)
        if i != 6:
            description += "\n"
    np.savetxt(directory + newname, table, fmt='%s', delimiter='\t', newline='\n',
    ↪ header=description, comments='')

#Creates a dimer of the two spheres filename1 and filename2  in
# directory and separates them by d times the radius of sphere
#filename1. The dimer is stored as newname in directory
def two_spheres(directory, filename1, filename2, newname, d):
    fname1 = directory + filename1
    fname2 = directory + filename2
    table1 = np.loadtxt(fname1, delimiter=None, skiprows = 7, dtype = str).T
    table2 = np.loadtxt(fname2, delimiter=None, skiprows = 7, dtype = str).T
    N = len(table1[0]) + len(table2[0])
    D = np.amax(table1[1].astype(np.int)) - np.amin(table2[1].astype(np.int)) +
    ↪ int(np.round(d*0.5*(np.amax(table1[1].astype(np.int)) -
    ↪ np.amin(table1[1].astype(np.int))))) - 2
    table2[1] = np.array(table2[1].astype(np.int) + D, dtype = str)
    table2[0] = np.arange(len(table1[0]), N).astype(str)
    table = np.concatenate((table1, table2), axis = 1)
    description = ""
    for i in range(7):
```

```
207         string = np.loadtxt(fname1, delimiter=None, skiprows = i, max_rows = 1,
            ↪ dtype = str)
208         if i == 1:
209             string[0] = str(N)
210         description += '\t'.join(string)
211         if i != 6:
212             description += "\n"
213     np.savetxt(directory + newname, table[[0,2,1,3,5,4,6]].T, fmt='%s',
        ↪ delimiter='\t', newline='\n', header=description, comments='')
214
215 #Modifies the parameter file ddpostprocess.par in directory
216 #to evaluate the electric near field on the N_y points
217 # spaced linearly between the two points (x, y_a, z) and
218 #(x, y_b, z). This is done for all the N_x values linearly
219 #spaced from x_a to x_b.
220 def DDPOSTPROCESS_file(directory, x_a, x_b, y_a, y_b, z, N_x, N_y):
221     N_y -= 1
222     fname = "ddpostprocess.par"
223     description = ""
224     for i in range(4):
225         description += ' '.join(np.loadtxt(directory + fname, delimiter=None,
            ↪ skiprows = i, max_rows = 1, dtype = str))
226         if i != 3:
227             description += "\n"
228     table = np.zeros([7, N_x], dtype=object)
229     table[2] = np.ones(N_x, dtype = np.float64)*z
230     table[5] = np.ones(N_x, dtype = np.float64)*z
231     table[1] = np.ones(N_x, dtype = np.float64)*y_a
232     table[4] = np.ones(N_x, dtype = np.float64)*y_b
233     table[6] = np.ones(N_x, dtype = int)*N_y
234     table[0] = np.linspace(x_a, x_b, N_x)
235     table[3] = np.linspace(x_a, x_b, N_x)
236     np.savetxt(directory + fname, table.T, fmt='%s', delimiter='\t', newline='\n',
        ↪ header=description, comments='')
237
238 #Returns a matrix of dimensions [N_x, N_y] of all the values
239 #for the electric field |E|/|E_0| stored in the file
240 #"ddpostprocess.out" in directory.
241 def Nearfield(directory, N_x, N_y):
242     indices = np.zeros(N_x, dtype = int)
243     index = 0
244     for i in range(N_x):
245         if i % 1000 == 0:
246             print(i)
247         found = False
248         while found == False:
249             text = np.loadtxt(directory + "ddpostprocess_P.out", delimiter=None,
                ↪ skiprows = index, max_rows = 1, dtype = str).T
```

```
250             if text[0] == "x_TF":
251                 indices[i] = index + 1
252                 found = True
253             index += 1
254 
255     Field = np.zeros([N_x, N_y], dtype = np.complex128)
256     for i in prange(N_x):
257         if i % 1000 == 0:
258             print(i)
259         table = np.loadtxt(directory + "ddpostprocess.out", delimiter=None,
            ↪  skiprows = indices[i] + N_y*i, max_rows = N_y, dtype = np.float64).T
260         E_x, E_y, E_z = table[3] + 1.j*table[4], table[5] + 1.j*table[6], table[7]
            ↪  + 1.j*table[8]
261         Field[i] = np.sqrt(E_x*np.conj(E_x) + E_y*np.conj(E_y) + E_z*np.conj(E_z))
262     return Field
```

B.2 Snippet of the Makefile

```
#--------- do NOT alter the following definitions: -------------------------
MPI_f = mpi_subs.f90 \
 mpi_bcast_char.f90 mpi_bcast_cplx.f90 mpi_bcast_int.f90\
 mpi_bcast_int2.f90 mpi_bcast_real.f90
MPI_o = mpi_subs.o \
 mpi_bcast_char.o mpi_bcast_cplx.o mpi_bcast_int.o\
 mpi_bcast_int2.o mpi_bcast_real.o
MKL_f = cxfft3_mkl.f90 mkl_dfti.f90
MKL_o = cxfft3_mkl.o mkl_dfti.o
MKL_m = mkl_dfti.mod
#-------------------------------------------------------------------------

# 7.  ifort compiler
#     sp + MKL + OpenMP + no MPI

# on some systems, before compiling, type
#   module purge
#   module load intel-mkl

# define the following:
PRECISION = sp
CXFFTMKL.f = $(MKL_f)
CXFFTMKL.o = $(MKL_o)
MKLM  = $(MKL_m)
DOMP  = -Dopenmp
OPENMP  = -qopenmp
MPI.f  = mpi_fake.f90
MPI.o  = mpi_fake.o
DMPI  =
FC  = ifort
FFLAGS  = -O2
LFLAGS  = -lmkl_intel_thread -lmkl_core -lpthread -lmkl_intel_lp64
```

B.3 The parameter file

```
'========== Parameter file for v7.3; created: 21_04_11 ==========='
'**** Preliminaries ****'
'NOTORQ' = CMTORQ*6 (NOTORQ, DOTORQ) -- either do or skip torque calculations
'PBCGS2' = CMDSOL*6 (PBCGS2, PBCGST, PETRKP) -- select solution method
'FFTMKL' = CMDFFT*6 (GPFAFT, FFTMKL)
'GKDLDR' = CALPHA*6 (GKDLDR, LATTDR)
'NOTBIN' = CBINFLAG (NOTBIN, ORIBIN, ALLBIN)
'**** Initial Memory Allocation ****'
200 200 200 = dimensioning allowance for target generation
'**** Target Geometry and Composition ****'
'FRMFILPBC' = CSHAPE*9 shape directive
110 110 1 'target_100r1_0r2_523305N.txt' (quotes must be used)
1 = NCOMP = number of dielectric materials
'../diel/Ag_evap.txt'
'**** Additional Near field calculation? ****'
0 = NRFLD (=0 to skip, 1 to calculate nearfield E)
0.5 0.5 0.5 0.5 0.5 0.5  (fract. extens. of calc. vol. in -x,+x,-y,+y,-z,+z)
'**** Error Tolerance ****'
1e-05 = TOL = MAX ALLOWED (NORM OF |G>=AC|E>-ACA|X>)/(NORM OF AC|E>)
'**** Maximum number of iterations allowed ****'
2000 = MXITER
'**** Interaction cutoff parameter for PBC calculations ****'
0.008 = GAMMA (1e-2 is normal, 3e-3 for greater accuracy)
'**** Angular resolution for calculation of <cos>, etc. ****'
2 = ETASCA (number of angles is proportional to [(3+x)/ETASCA]2 )
'**** Vacuum wavelengths (micron) ****'
0.335 0.459 11 'INV' = wavelengths (first,last,how many,how=LIN,INV,LOG)
'**** Refractive index of ambient medium ****'
1 = NAMBIENT
'**** Effective Radii (micron) ****'
0.00999813 0.00999813 1 'LIN' = eff. radii (first, last, how many, how=LIN,INV,LOG)
'**** Define Incident Polarizations ****'
(0,0) (1,0) (0,0) = Polarization state e01 (k along x axis)
2 = IORTH (=1 to do only pol. state e01; =2 to also do orth. pol. state)
'**** Specify which output files to write ****'
1 = IWRKSC (=0 to suppress, =1 to write ".sca" file for each target orient)
'**** Prescribe Target Rotations ****'
0 0 1 = BETAMI, BETAMX, NBETA (beta=rotation around a1)
45 45 1 = THETMI, THETMX, NTHETA (theta=angle between a1 and k)
0 0 1 = PHIMIN, PHIMAX, NPHI (phi=rotation angle of a1 around k)
'**** Specify first IWAV, IRAD, IORI (normally 0 0 0) ****'
0 0 0 = first IWAV, first IRAD, first IORI (0 0 0 to begin fresh)
'**** Select Elements of S_ij Matrix to Print ****'
2 = NSMELTS = number of elements of S_ij to print (not more than 9)
11 12 = indices ij of elements to print
'**** Specify Scattered Directions ****'
'TFRAME' = CMDFRM (LFRAME, TFRAME for Lab Frame or Target Frame)
1 = number of scattering orders
0 0 = M, N (diffraction orders)
```

B.4 The target file

---Sphere target---
523305 = Number of dipoles
1 0 0 = target vector X-Axis
0 1 0 = target vector Y_Axis
1 1 1 = dx/d dy/d dz/d (normally 1,1,1)
50 0 0 = location in lattice of target origin
ID x y z icx icy icz
1 50 0 -50 1 1 1
2 46 -9 -49 1 1 1
3 47 -9 -49 1 1 1
4 48 -9 -49 1 1 1
5 49 -9 -49 1 1 1
6 50 -9 -49 1 1 1
7 51 -9 -49 1 1 1
8 52 -9 -49 1 1 1
9 53 -9 -49 1 1 1
10 54 -9 -49 1 1 1
11 45 -8 -49 1 1 1
12 46 -8 -49 1 1 1
13 47 -8 -49 1 1 1
14 48 -8 -49 1 1 1
15 49 -8 -49 1 1 1
16 50 -8 -49 1 1 1
17 51 -8 -49 1 1 1
18 52 -8 -49 1 1 1
19 53 -8 -49 1 1 1
20 54 -8 -49 1 1 1
21 55 -8 -49 1 1 1
22 43 -7 -49 1 1 1
23 44 -7 -49 1 1 1
24 45 -7 -49 1 1 1
25 46 -7 -49 1 1 1
26 47 -7 -49 1 1 1
27 48 -7 -49 1 1 1
28 49 -7 -49 1 1 1
29 50 -7 -49 1 1 1
30 51 -7 -49 1 1 1
.
.
.

B.5 The dielectric file

Dielectric values for evaporated silver from SOPRA database
1 2 3 0 0 = columns for wave, Re(n), Im(n), eps1, eps2
wave(um) Re(n) Im(n) eps1 eps2
0.1878548295848922 0.995 1.13 0.990025 1.2768999999999997
0.1892888359176013 1.00425 1.149375 1.0085180625 1.3210628906250002
0.19074490388619822 1.012 1.16 1.024144 1.3456
0.1922235465519827 1.0195 1.168125 1.0393802500000002 1.3645160156250002
0.19372529300942007 1.028 1.18 1.056784 1.3923999999999999
0.19525068901736828 1.0375 1.194375 1.0764062500000002 1.426531640625
0.19680029766036325 1.048 1.21 1.0983040000000002 1.4641
0.19837470004164615 1.059625 1.225625 1.1228051406250001 1.502156640625
0.19997449600972395 1.072 1.24 1.1491840000000002 1.5376
0.2016003049203721 1.084812 1.25125 1.1768170753440002 1.5656265624999999
0.20325276643611287 1.098 1.26 1.2056040000000001 1.5876000000000001
0.20493254136533695 1.111625 1.265625 1.2357101406250002 1.601806640625
0.20664031254338142 1.125 1.27 1.265625 1.6129
0.20837678575803167 1.137188 1.275 1.2931965473440001 1.6256249999999999
0.21014269072208278 1.149 1.28 1.320201 1.6384
0.21193878209577582 1.161438 1.285312 1.3489382278439999 1.652026937344
0.2137658405621187 1.173 1.29 1.3759290000000002 1.6641000000000001
0.21562467395831103 1.181875 1.292813 1.396828515625 1.671365452969
.
.
.

B.6 The postrocessing parameter file

'w000r000k000.E1' = name of file with E stored
'VTRoutput' = prefix for name of VTR output files
1 = IVTR (set to 1 to create VTR output)
1 = ILINE (set to 1 to evaluate E along a line)
-0.02500 -0.0045 0.0 -0.02500 0.0455 0.0 5000
-0.02499 -0.0045 0.0 -0.02499 0.0455 0.0 5000
-0.02498 -0.0045 0.0 -0.02498 0.0455 0.0 5000
-0.02497 -0.0045 0.0 -0.02497 0.0455 0.0 5000
-0.02496 -0.0045 0.0 -0.02496 0.0455 0.0 5000
-0.02495 -0.0045 0.0 -0.02495 0.0455 0.0 5000
-0.02494 -0.0045 0.0 -0.02494 0.0455 0.0 5000
-0.02493 -0.0045 0.0 -0.02493 0.0455 0.0 5000
-0.02492 -0.0045 0.0 -0.02492 0.0455 0.0 5000
-0.02491 -0.0045 0.0 -0.02491 0.0455 0.0 5000
-0.02490 -0.0045 0.0 -0.02490 0.0455 0.0 5000
-0.02489 -0.0045 0.0 -0.02489 0.0455 0.0 5000
-0.02488 -0.0045 0.0 -0.02488 0.0455 0.0 5000
-0.02487 -0.0045 0.0 -0.02487 0.0455 0.0 5000
-0.02486 -0.0045 0.0 -0.02486 0.0455 0.0 5000
-0.02485 -0.0045 0.0 -0.02485 0.0455 0.0 5000
-0.02484 -0.0045 0.0 -0.02484 0.0455 0.0 5000
-0.02483 -0.0045 0.0 -0.02483 0.0455 0.0 5000
.
.
.

Fredrik Knapskog

Plasmonic response of supported and interacting spherical nanoparticles

NTNU
Norwegian University of
Science and Technology