Tutorial of Generalized Mie Theory

Theoretical Background and Implementation

Ming-Wei Lee* and Liang-Yan Hsu*

Institute of Atomic and Molecular Sciences, Academia Sinica, Taipei 106, Taiwan

E-mail: mingwei.wayne.lee@gmail.com; hsu.liangyan@gmail.com

Contents

Part I. Theoretical Background

Electromagnetic Scattering Problems and Maxwell's Equations

Mie theory provides a systematic framework to solve the linear electromagnetic scattering problem of spherically symmetric scatterers. In Mie theory, we usually assume that the incident light is a plane wave; however, the incident light can no longer be a plane wave (e.g., electric dipole) when refering to a generalized Mie theory. This tutorial presents the details of the generalized Mie theory for an electric dipole as a source, including solution and implementation. The governing equation in Mie theory originates from macroscopic Maxwell's equations (SI unit):

$$\nabla \cdot \mathbf{D}(\mathbf{r}, t) = 0, \tag{1a}$$

$$\nabla \cdot \mathbf{B}(\mathbf{r}, t) = 0, \tag{1b}$$

$$\nabla \times \mathbf{E}(\mathbf{r}, t) = -\frac{\partial}{\partial t}\mathbf{B}(\mathbf{r}, t), \tag{1c}$$

$$\nabla \times \mathbf{H}(\mathbf{r}, t) = \frac{\partial}{\partial t}\mathbf{D}(\mathbf{r}, t), \tag{1d}$$

where $\mathbf{D}(\mathbf{r}, t)$, $\mathbf{E}(\mathbf{r}, t)$, $\mathbf{B}(\mathbf{r}, t)$, and $\mathbf{H}(\mathbf{r}, t)$ denote displacement fields, electric fields, magnetic fields, and magnetizing fields. Note that Eqs. (1a) and (1d) depict a system without free charge and current density. To describe the electromagnetic response from materials, we consider the following constitutive relations,

$$\begin{aligned}\mathbf{D}(\mathbf{r}, t) &\equiv \epsilon_0 \mathbf{E}(\mathbf{r}, t) + \mathbf{P}_{\mathrm{mat}}(\mathbf{r}, t) + \mathbf{P}_{\mathrm{D}}(\mathbf{r}, t), \\ &\simeq \epsilon_0 \int_{-\infty}^{t} \epsilon_{\mathrm{r}}(\mathbf{r}, t-\tau)\mathbf{E}(\mathbf{r}, \tau)\, \mathrm{d}\tau + \mathbf{P}_{\mathrm{D}}(\mathbf{r}, t), \end{aligned} \tag{2a}$$

$$\begin{aligned}\mathbf{H}(\mathbf{r}, t) &\equiv \mu_0^{-1}\mathbf{B}(\mathbf{r}, t) - \mathbf{M}_{\mathrm{mat}}(\mathbf{r}, t) \\ &= \mu_0^{-1}\mathbf{B}(\mathbf{r}, t), \end{aligned} \tag{2b}$$

where $\mathbf{P}_{\mathrm{mat}}(\mathbf{r},t)$ and $\mathbf{M}_{\mathrm{mat}}(\mathbf{r},t)$ are the polarization field and the magnetization field from materials, respectively. Here, we suppose the polarization field induced by materials is linearly proportional to electric fields and is suitably described by dielectric functions. Moreover, we suppose that the time non-locality dielectric functions are invariant under time translations. Thus, the displacement field can be described by the convolution of dielectric functions and the electric field, as shown in Eq. (2a). It is worth to remind that the upper limit of the integral in Eq. (2a) is t because the material response in future cannot affect the electric field at time t. To add electric dipole as an external source, we include the polarization field $\mathbf{P}_{\mathrm{D}}(\mathbf{r},t)$ in the displacement field. Finally, we do not focus on magnetic materials; thus, the magnetization of materials is excluded in the constitutive relations. For the sake of convenience, we make a Fourier transform to the constitutive relations,

$$\begin{aligned}\mathbf{D}(\mathbf{r},\omega) &\equiv \int_{-\infty}^{\infty} \mathbf{D}(\mathbf{r},t)\cdot e^{i\omega t}\,\mathrm{d}t \\ &= \epsilon_0\epsilon_{\mathrm{r}}(\mathbf{r},\omega)\mathbf{E}(\mathbf{r},\omega)+\mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega), \qquad (3a)\\ \mathbf{H}(\mathbf{r},\omega) &\equiv \int_{-\infty}^{\infty} \mathbf{H}(\mathbf{r},t)\cdot e^{i\omega t}\,\mathrm{d}t \\ &= \mu_0^{-1}\mathbf{B}(\mathbf{r},\omega), \qquad (3b)\end{aligned}$$

because the convolution is simply reduced to the product in the frequency domain. In the frequency domain, the Maxwell's equations become:

$$\nabla\cdot\mathbf{D}(\mathbf{r},\omega)=0, \qquad (4a)$$

$$\nabla\cdot\mathbf{B}(\mathbf{r},\omega)=0, \qquad (4b)$$

$$\nabla\times\mathbf{E}(\mathbf{r},\omega)=i\omega\mathbf{B}(\mathbf{r},\omega), \qquad (4c)$$

$$\nabla\times\mathbf{H}(\mathbf{r},\omega)=-i\omega\mathbf{D}(\mathbf{r},\omega). \qquad (4d)$$

Making a few steps of algebraic operations and requiring $\epsilon_{\mathrm{r}}(\mathbf{r},\omega)\to\epsilon_{\mathrm{r},i}(\omega)$ to be piecewise-homogeneous function, we get two second-order inhomogeneous differential equations (detail

derivation and description can be found in Appendix A). One is related to the electric field in the i-th region $[\mathbf{E}^{(i)}(\mathbf{r},\omega)]$,

$$\left[\frac{\omega^2\epsilon_{\mathrm{r},i}(\omega)}{c^2}-\nabla\times\nabla\times\right]\mathbf{E}^{(i)}(\mathbf{r},\omega)=-\frac{\omega^2}{\epsilon_0 c^2}\sum_j\mathbf{P}_{\mathrm{D}}^{(j)}(\mathbf{r},\omega), \tag{5}$$

and the other is related to the magnetizing field in the i-th region $[\mathbf{H}^{(i)}(\mathbf{r},\omega)]$,

$$\left[\frac{\omega^2\epsilon_{\mathrm{r},i}(\omega)}{c^2}-\nabla\times\nabla\times\right]\mathbf{H}^{(i)}(\mathbf{r},\omega)=i\omega\nabla\times\sum_j\mathbf{P}_{\mathrm{D}}^{(j)}(\mathbf{r},\omega) \tag{6}$$

Here, $\mathbf{P}_{\mathrm{D}}^{(j)}(\mathbf{r},\omega)$ is the polarization field created by a electric dipole in the j-th region. Note that the superscript indices with parentheses denote the region according to the piecewise function $\epsilon_{\mathrm{r},i}(\omega)$; they should not be interpreted as the vector components.

Green's Function Method for Maxwell's Equations

The inhomogeneous differential equation for the electric field in Eq. (5) can be solved via Green's function method. The dyadic Green's function of Maxwell's equations is defined as follows,

$$\mathscr{L}_i\,\overline{\overline{\mathbf{G}}}^{(ij)}(\mathbf{r},\mathbf{r}',\omega)\equiv\left[\frac{\omega^2\epsilon_{\mathrm{r},i}(\omega)}{c^2}-\nabla\times\nabla\times\right]\overline{\overline{\mathbf{G}}}^{(ij)}(\mathbf{r},\mathbf{r}',\omega)=-\bar{\bar{\mathbf{I}}}\delta(\mathbf{r}-\mathbf{r}'), \tag{7}$$

where $\bar{\bar{\mathbf{I}}}$ and $\delta(\mathbf{r}-\mathbf{r}')$ denote the three dimensional identity matrix and delta function, respectively. Utilizing the dyadic Green's function, we can calculate the electric field by

$$\mathbf{E}^{(i)}(\mathbf{r},\omega)=\mathbf{E}_{\mathrm{homo}}^{(i)}(\mathbf{r},\omega)+\frac{\omega^2}{c^2\epsilon_0}\sum_j\int\overline{\overline{\mathbf{G}}}^{(ij)}(\mathbf{r},\mathbf{r}',\omega)\cdot\mathbf{P}_{\mathrm{D}}^{(j)}(\mathbf{r}',\omega)\,\mathrm{d}^3\mathbf{r}', \tag{8}$$

where $\mathbf{E}_{\mathrm{homo}}^{(i)}(\mathbf{r},\omega)$ is the homogeneous solution of Maxwell's equations. Here, we adopt the spectral method to solve the dyadic Green's function; therefore, we need to introduce the eigenfunctions (i.e., the homogeneous solutions) of the linear operator $\mathscr{L}_i$ first.

Strum-Liouville Problem and Eigenfunctions

According to the Helmholtz decomposition, a vector field that decays faster than $1/r$ can be decomposed into a curl-free vector field (longitudinal mode) and two divergence-free vector fields (transverse modes). Therefore, the homogeneous solution of the linear operator,

$$\mathscr{L}_i \, \mathbf{X}^{(i)}(\mathbf{r},\omega) = \left[k_i^2(\omega) - \nabla \times \nabla \times\right] \mathbf{X}^{(i)}(\mathbf{r},\omega) = 0, \tag{9}$$

becomes the summation of $\mathbf{L}^{(i)}(\mathbf{r},\omega)$ (curl-free vector field), $\mathbf{M}^{(i)}(\mathbf{r},\omega)$ and $\mathbf{N}^{(i)}(\mathbf{r},\omega)$ (two divergence-free vector fields),

$$\mathbf{X}^{(i)}(\mathbf{r},\omega) = \mathbf{L}^{(i)}(\mathbf{r},\omega) + \mathbf{M}^{(i)}(\mathbf{r},\omega) + \mathbf{N}^{(i)}(\mathbf{r},\omega). \tag{10}$$

Note that $\mathbf{X}^{(i)}(\mathbf{r},\omega) = \mathbf{E}^{(i)}(\mathbf{r},\omega)$ or $\mathbf{H}^{(i)}(\mathbf{r},\omega)$ and $k_i(\omega) = \underline{n}_i(\omega)k_0(\omega)$, where $k_0(\omega) = \omega/c$ and $\underline{n}_i = \sqrt{\epsilon_{\mathrm{r},i}(\omega)}$ are the magnitude of the wavevector in vacuum and the complex refractive index in the i-th medium respectively. The curl-free vector field $\mathbf{L}^{(i)}(\mathbf{r},\omega)$ can be intuitively generated by a scalar function $\phi^{(i)}(\mathbf{r},\omega)$,

$$\mathbf{L}^{(i)}(\mathbf{r},\omega) \equiv \mathscr{T}_{\mathbf{L}} \left\{\phi^{(i)}(\mathbf{r},\omega)\right\} = k_i^{-1}(\omega) \; \nabla\phi^{(i)}(\mathbf{r},\omega). \tag{11}$$

It is worthwhile to mention that the contribution of the curl-free field $\mathbf{L}^{(i)}(\mathbf{r},\omega)$ is suppressed in Eq. (9) because Eq. (9) is source-less (i.e., a system is electrically neutral). Although $\mathbf{L}^{(i)}(\mathbf{r},\omega)$ is unimportant in the divergenceless system, it cannot be ignored in the representation of Green's functions due to the completeness of basis functions. Moreover, the divergence-free vector fields can also be generated by the scalar function via

$$\mathbf{M}^{(i)}(\mathbf{r},\omega) \equiv \mathscr{T}_{\mathbf{M}} \left\{\phi^{(i)}(\mathbf{r},\omega)\right\} = \nabla \times [\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)], \tag{12}$$

$$\mathbf{N}^{(i)}(\mathbf{r},\omega) \equiv \mathscr{T}_{\mathbf{N}} \left\{\phi^{(i)}(\mathbf{r},\omega)\right\} = k_i^{-1}(\omega) \; \nabla \times \nabla \times [\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)]. \tag{13}$$

Here, $\mathbf{r}$ accompanied with the scalar function $\phi^{(i)}(\mathbf{r},\omega)$ is the pilot vector in the spherically symmetric system.[1] It can be shown that the scalar function $\phi^{(i)}(\mathbf{r},\omega)$ obeys the linear differential equation (details can be found in Appendix B),

$$\left[\nabla^2 + \frac{\omega^2 \epsilon_{\mathrm{r},i}(\omega)}{c^2}\right] \phi^{(i)}(\mathbf{r},\omega) = \left[\nabla^2 + k_i^2(\omega)\right] \phi^{(i)}(\mathbf{r},\omega) = 0. \tag{14}$$

One type of the eigenfunctions of the scalar differential equation is the combination of spherical Bessel functions $[j_n(k_i r)]$, associated Legendre polynomials $[P_n^m(\cos\theta)]$, and exponential functions $(e^{im\phi})$ (see Appendix C for detail derivations)[1],

$$\phi_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) = j_n(k_i r) P_n^m(\cos\theta) e^{im\phi}, \tag{15}$$

where $n \in \mathbb{N}_0$ (including zero) and $m \in \{x | -n \leq x \leq n, n \in \mathbb{N}_0\}$. In the following discussion, we set the definitions of n and m to default if we do not specifically emphasize the range of n and m. Note that $\phi_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$ is not normalized and the superscript of the Roman numeral 'I' indicates the spherical Bessel function of the first kind is applied. Thus, $\phi^{(i)}(\mathbf{r},\omega)$ is the superposition of the eigenfunction $\phi_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$,

$$\phi^{(i)}(\mathbf{r},\omega) = \sum_{n=0}^{\infty} \sum_{m=-n}^{n} c_{nm}(k_i) \phi_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi), \tag{16}$$

where $c_{nm}(k_i)$ is the expansion coefficient (conceptually). In addition, it is worth mentioning that the dimensionless variable $k_i r$ indicates the relative scale is the only important stuff. According to Eq. (15), the vector spherical functions [eigenfunctions of Eq. (9)] defined by

[1] From now on, we simply denote $\underline{n}_i(\omega)$ and $k_0(\omega)$ by $\underline{n}_i$ and k_0. Please keep in mind that both of them are ω-dependent, and the discussion is in frequency space.

the linear transformations in Eqs. (11) to (13) become

$$\begin{aligned}
\mathbf{L}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) &= k_i^{-1}\nabla\phi_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) \\
&= \frac{\mathrm{d}j_n(k_i r)}{\mathrm{d}(k_i r)}P_n^m(\cos\theta)e^{im\phi}\ \hat{r} + \frac{j_n(k_i r)}{k_i r}\cdot e^{im\phi}\left[\tau_{nm}(\theta)\ \hat{\theta} + i\pi_{nm}(\theta)\ \hat{\phi}\right], \qquad (17)\\
\mathbf{M}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) &= \nabla\times\left[\mathbf{r}\phi_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)\right] \\
&= j_n(k_i r)e^{im\phi}\left[i\pi_{nm}(\theta)\hat{\theta} - \tau_{nm}(\theta)\hat{\phi}\right], \qquad (18)\\
\mathbf{N}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) &= k_i^{-1}\nabla\times\mathbf{M}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) \\
&= \frac{j_n(k_i r)}{k_i r}\cdot n(n+1)P_n^m(\cos\theta)e^{im\phi}\hat{r} \\
&\quad + \frac{1}{k_i r}\frac{\mathrm{d}\psi_n(k_i r)}{\mathrm{d}(k_i r)}\cdot e^{im\phi}\left[\tau_{nm}(\theta)\hat{\theta} + i\pi_{nm}(\theta)\hat{\phi}\right]. \qquad (19)
\end{aligned}$$

In Eq. (18) and (19), we define the three new functions, π function, τ function,

$$\tau_{nm}(\theta) = \frac{d}{d\theta}\left[P_n^m(\cos\theta)\right], \qquad (20)$$

$$\pi_{nm}(\theta) = \frac{m}{\sin\theta}P_n^m(\cos\theta), \qquad (21)$$

and Riccati-Bessel function,

$$\psi_n(k_i r) = k_i r\cdot j_n(k_i r). \qquad (22)$$

Here, we would like to emphasize that $\mathbf{L}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$, $\mathbf{M}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$, and $\mathbf{N}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$ have not been normalized, and the superscripts (Roman numerals) imply the spherical Bessel functions of the first kind is applied.

Orthogonality of the Eigenfunctions [Vector (Scalar) Spherical Functions]

The orthogonality of the basis functions provide us a complete set to expand the dyadic Green's functions. Because the vector spherical functions [$\mathbf{L}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$, $\mathbf{M}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$, and $\mathbf{N}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$] are elements in $\mathscr{H}\otimes\mathbb{R}^3$, it is necessary to prove that the orthogonality in the two subspace individually. Note that $\mathscr{H}$ and $\mathbb{R}^3$ denote a Hilbert space and a three-dimensional Euclidean space, respectively. First, we investigate the orthogonality of scalar spherical functions [$\phi_{nm}^{(\mathrm{I})}(kr,\theta,\phi)\in\mathscr{H}$]. For $k,k'\in\mathbb{R}$, the orthogonality of scalar spherical functions is expressed by

$$\begin{aligned}
&\langle k',n',m'|k,n,m\rangle\\
&=\langle k',n',m'|\left[\int \mathrm{d}^3\mathbf{r}\ \ |r,\theta,\phi\rangle\langle r,\theta,\phi|\right]|k,n,m\rangle\\
&=\int \mathrm{d}^3\mathbf{r}\ \phi_{n'm'}^{(\mathrm{I})*}(k'r,\theta,\phi)\cdot\phi_{nm}^{(\mathrm{I})}(kr,\theta,\phi)\\
&=\int_0^\infty j_n(k'r)j_n(kr)\ r^2\mathrm{d}r\int_0^\pi P_{n'}^{m'}(\cos\theta)P_n^m(\cos\theta)\ \sin\theta\mathrm{d}\theta\int_0^{2\pi}e^{-im'\phi}e^{im\phi}\ \mathrm{d}\phi. \qquad (23)
\end{aligned}$$

It is found that only the azimuthal part of $\phi_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$ takes a complex conjugate in the evaluation. To avoid the ambiguity when describing dissipative environments (the input argument of spherical Bessel functions is in the complex domain), we use the notation

$$(-1)^m\phi_{n(-m)}^{(\mathrm{I})}(kr,\theta,\phi)=j_n(kr)P_n^m(\cos\theta)e^{-im\phi} \qquad (24)$$

to express the complex conjugate instead of $\phi_{n'm'}^{(\mathrm{I})*}(kr,\theta,\phi)$. Note that we use the identity[2], $P_n^{(-m)}(\cos\theta)=(-1)^mP_n^m(\cos\theta)$. According to Eq. (24), the orthogonality of the scalar spherical function $\phi_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$ is

$$(-1)^{m'}\int\phi_{n'(-m')}^{(\mathrm{I})}(k'r,\theta,\phi)\phi_{nm}^{(\mathrm{I})}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r}=\frac{\pi\delta(k-k')}{2k^2}f_{nm}\delta_{nn'}\delta_{mm'}, \qquad (25)$$

[2]The definition of associated Legendre polynomials when $m<0$ is slightly different from the common definition. Please see also Appendix C.

where we use the two identities,[1]

$$\int j_n(k'r)j_n(kr)\ r^2\mathrm{d}r = \frac{\pi\delta(k-k')}{2k^2}, \qquad k,k' \in \mathbb{R}, \tag{26}$$

and

$$\int P_{n'}^{m'}(\cos\theta)P_n^m(\cos\theta)e^{i(m-m')\phi}\ \mathrm{d}\Omega = f_{nm}\delta_{nn'}\delta_{mm'}, \qquad f_{nm} = \frac{4\pi}{2n+1}\frac{(n+|m|)!}{(n-|m|)!}. \tag{27}$$

Hence, we can define the (partially) normalized scalar spherical functions,

$$\underline{\phi}_{nm}^{(\mathrm{I})}(kr,\theta,\phi) \equiv \frac{1}{\sqrt{f_{nm}}}\cdot\phi_{nm}^{(\mathrm{I})}(kr,\theta,\phi), \tag{28}$$

so that

$$(-1)^{m'}\int \underline{\phi}_{n'(-m')}^{(\mathrm{I})}(k'r,\theta,\phi)\underline{\phi}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r} = \frac{\pi\delta(k-k')}{2k^2}\delta_{nn'}\delta_{mm'}. \tag{29}$$

Second, we investigate the orthogonality of vector spherical functions. To check the orthogonality of vector spherical functions, we have to identify $C_2^3+3=6$ combinations between each pair of $\mathbf{L}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$, $\mathbf{M}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$, and $\mathbf{N}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$. According to Eqs. (17) - (19), we get that

$$\begin{aligned}&(-1)^{m'}\int \mathbf{L}_{n'(-m')}^{(\mathrm{I})}(k'r,\theta,\phi)\cdot\mathbf{M}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r}\\ &= \frac{1}{k'}\int_0^\infty j_n(k'r)j_n(kr)r\ \mathrm{d}r\int i\left[\tau_{n'm'}(\theta)\pi_{nm}(\theta)+\pi_{n'm'}(\theta)\tau_{nm}(\theta)\right]e^{i(m-m')\phi}\ \mathrm{d}\Omega\end{aligned} \tag{30}$$

It can be shown that the two vector spherical functions are orthogonal (i.e., the integral becomes zero) because the integral of the angular part is zero,

$$\int i\left[\tau_{n'm'}(\theta)\pi_{nm}(\theta)+\pi_{n'm'}(\theta)\tau_{nm}(\theta)\right]e^{i(m-m')\phi}\ \mathrm{d}\Omega = 0 \tag{31}$$

The proof of this integral can be found in Appendix D. Thus, $\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ and $\mathbf{M}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ are orthogonal. For the same reason, we can also show that

$$\begin{aligned}
&(-1)^{m'}\int \mathbf{N}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\mathbf{M}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r}\\
&=\frac{1}{k'}\int_0^\infty \frac{\mathrm{d}\psi_n(k'r)}{\mathrm{d}(k'r)}j_n(kr)r\ \mathrm{d}r\int i\left[\tau_{n'm'}(\theta)\pi_{nm}(\theta)+\pi_{n'm'}(\theta)\tau_{nm}(\theta)\right]e^{i(m-m')\phi}\ \mathrm{d}\Omega=0
\end{aligned}\tag{32}$$

In the case of $\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ and $\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$, the integral is slightly more complicated,

$$\begin{aligned}
&(-1)^{m'}\int \mathbf{L}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r}\\
&=\int_0^\infty \frac{\mathrm{d}j_{n'}(k'r)}{\mathrm{d}(k'r)}\frac{j_n(kr)}{kr}\ r^2\mathrm{d}r\int n(n+1)P^{m'}_{n'}(\cos\theta)P^m_n(\cos\theta)e^{i(m-m')\phi}\ \mathrm{d}\Omega\\
&+\int_0^\infty \frac{j_{n'}(k'r)}{k'r}\frac{1}{kr}\frac{\mathrm{d}\psi_n(kr)}{\mathrm{d}(kr)}\ r^2\mathrm{d}r\int\left[\tau_{n'm'}(\theta)\tau_{nm}(\theta)+\pi_{n'm'}(\theta)\pi_{nm}(\theta)\right]e^{i(m-m')\phi}\ \mathrm{d}\Omega
\end{aligned}\tag{33}$$

In Appendix D, we show that the integral with respect to the solid angle gives the result of

$$\begin{aligned}
&\int\left[\tau_{n'm'}(\theta)\tau_{nm}(\theta)+\pi_{n'm'}(\theta)\pi_{nm}(\theta)\right]e^{i(m-m')\phi}\ \mathrm{d}\Omega\\
&=n(n+1)\int_0^\pi P^m_{n'}(\cos\theta)P^m_n(\cos\theta)\sin\theta\mathrm{d}\theta=n(n+1)f_{nm}\delta_{nn'}\delta_{mm'}.
\end{aligned}\tag{34}$$

Therefore, Eq. (33) can be simplified to

$$\begin{aligned}
&(-1)^{m'}\int \mathbf{L}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r}\\
&=n(n+1)\int_0^\infty\left[\frac{\mathrm{d}j_{n'}(k'r)}{\mathrm{d}(k'r)}\frac{j_n(kr)}{kr}+\frac{j_{n'}(k'r)}{k'r}\frac{1}{kr}\frac{\mathrm{d}\psi_n(kr)}{\mathrm{d}(kr)}\right]r^2\mathrm{d}r\ f_{nm}\delta_{nn'}\delta_{mm'}
\end{aligned}\tag{35}$$

This consequence indicates that the angular part of $\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ and $\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ does not determine that the orthogonality when $n'=n$ and $m'=m$. We need to further verify whether the radial integral guarantees the total orthogonality. To simplify the radial integral,

we use the following recurrence relations,

$$\frac{j_n(kr)}{kr} = \frac{1}{2n+1}\left[j_{n-1}(kr) + j_{n+1}(kr)\right], \tag{36}$$

$$\frac{\mathrm{d}j_n(kr)}{d(kr)} = \frac{1}{2n+1}\left[nj_{n-1}(kr) - (n+1)j_{n+1}(kr)\right], \tag{37}$$

and the radial integral becomes

$$\begin{aligned}&\int_0^\infty \left[\frac{\mathrm{d}j_n(k'r)}{\mathrm{d}(k'r)}\frac{j_n(kr)}{kr} + \frac{j_n(k'r)}{k'r}\frac{1}{kr}\frac{\mathrm{d}\psi_n(kr)}{\mathrm{d}(kr)}\right] r^2\mathrm{d}r \\ &= \frac{1}{2n+1}\int_0^\infty \left[j_{n-1}(k'r)j_{n-1}(k'r) - j_{n+1}(k'r)j_{n+1}(k'r)\right]\ r^2\mathrm{d}r = 0\end{aligned} \tag{38}$$

Recall that the two integrals become two delta functions [Eq. (26)] that mutually cancel out. Moreover, the orthogonality of each vector spherical functions is also necessary for normalization. By using Eqs. (26) and (34), it is easy to obtain that

$$(-1)^{m'}\int \mathbf{M}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\mathbf{M}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r} = \frac{\pi\delta(k'-k)}{2k^2}n(n+1)f_{nm}\delta_{nn'}\delta_{mm'} \tag{39}$$

For the cases of $\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ and $\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$, a further derivation is needed due to the radial integrals. In the radial integral of $\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$, we encounter to the integral,

$$\begin{aligned}&\int_0^\infty \left[n(n+1)\frac{j_n(k'r)}{k'r}\frac{j_n(kr)}{kr} + \frac{1}{k'r}\frac{\mathrm{d}\psi_n(k'r)}{\mathrm{d}(k'r)}\frac{1}{kr}\frac{\mathrm{d}\psi_n(kr)}{\mathrm{d}(kr)}\right]\ r^2\mathrm{d}r \\ &= \frac{1}{2n+1}\int_0^\infty \left[(n+1)j_{n-1}(k'r)j_{n-1}(kr) + nj_{n+1}(k'r)j_{n+1}(kr)\right]\ r^2\mathrm{d}r = \frac{\pi}{2k^2}\delta(k-k'),\end{aligned} \tag{40}$$

where we use the recurrence relations of spherical Bessel functions to get the result. Combining the consequence of Eqs. (34) and (40) , we obtain that

$$(-1)^{m'}\int \mathbf{N}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r} = \frac{\pi\delta(k'-k)}{2k^2}n(n+1)f_{nm}\delta_{nn'}\delta_{mm'}. \tag{41}$$

Finally, the radial integral of $\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ gives that

$$\begin{aligned}&\int_0^\infty \left[\frac{\mathrm{d}j_n(k'r)}{\mathrm{d}(k'r)}\frac{\mathrm{d}j_n(kr)}{\mathrm{d}(kr)} + n(n+1)\frac{j_n(k'r)}{k'r}\frac{j_n(kr)}{kr}\right]\ r^2\mathrm{d}r \\ &= \frac{1}{2n+1}\int_0^\infty \left[nj_{n-1}(k'r)j_{n-1}(kr) + (n+1)j_{n+1}(k'r)j_{n+1}(kr)\right]\ r^2\mathrm{d}r = \frac{\pi}{2k^2}\delta(k-k'), \quad (42)\end{aligned}$$

and we obtain that

$$(-1)^{m'}\int \mathbf{L}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r} = \frac{\pi\delta(k'-k)}{2k^2}f_{nm}\delta_{nn'}\delta_{mm'}. \quad (43)$$

It is worthwhile to emphasize that the property of k, n, and m are quite different. For n and m, which is associated with the irreducible representation of $\mathfrak{so}(3)$ Lie algebra, the values are discretized. In other words, we can properly normalized the angular part of vector spherical functions. In contrast, k is continuous, which indicate the normalization of the radial part is not well-defined. In this tutorial, we define the (partially) normalized vector spherical functions as

$$\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \equiv \frac{1}{\sqrt{f_{nm}}}\cdot\mathbf{L}^{(\mathrm{I})}_{nm}(kr,\theta,\phi), \quad (44a)$$

$$\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \equiv \frac{1}{\sqrt{n(n+1)f_{nm}}}\cdot\mathbf{M}^{(\mathrm{I})}_{nm}(kr,\theta,\phi), \quad (44b)$$

$$\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \equiv \frac{1}{\sqrt{n(n+1)f_{nm}}}\cdot\mathbf{N}^{(\mathrm{I})}_{nm}(kr,\theta,\phi), \quad (44c)$$

so that

$$(-1)^{m'}\int \underline{\mathbf{F}}^{(\mathrm{I})}_{n'(-m')}(k'r,\theta,\phi)\cdot\underline{\mathbf{F}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\ \mathrm{d}^3\mathbf{r} = \frac{\pi\delta(k'-k)}{2k^2}\delta_{nn'}\delta_{mm'}, \quad (45)$$

where $\underline{\mathbf{F}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) = \left\{\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi),\ \underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi),\ \underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\right\}$.

Eigenfunction Expansion of the Free-Space Dyadic Green's Function

The next goal is expanding dyadic Green's functions by the complete basis. Before we discuss the Green's function of a spherical scatterer, we would like to discuss the simplest case first. In vacuum [3], the free-space dyadic Green's function $\overline{\overline{\mathbf{G}}}_{\text{vac}}(\mathbf{r},\mathbf{r}',\omega)$ denote is defined by

$$\mathscr{L}_0\,\overline{\overline{\mathbf{G}}}_{\text{vac}}(\mathbf{r},\mathbf{r}',\omega)=\left[k_0^2-\nabla\times\nabla\times\right]\overline{\overline{\mathbf{G}}}_{\text{vac}}(\mathbf{r},\mathbf{r}',\omega)=-\bar{\bar{\mathbf{I}}}\delta(\mathbf{r}-\mathbf{r}'). \tag{46}$$

First, we utilize the orthogonality of vector spherical functions ($\mathbf{L}$, $\mathbf{M}$, and $\mathbf{N}$) to expand the dyadic delta function (details can be found in Appendix E)

$$\begin{aligned}\bar{\bar{\mathbf{I}}}\delta(\mathbf{r}-\mathbf{r}')=\frac{2}{\pi}\int_0^\infty k^2\mathrm{d}k\sum_{nm}(-1)^m\Big[\;&\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{L}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\\&+\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\\&+\underline{\mathbf{N}}^{(\mathrm{I})}_{n}(kr,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\;\Big].\end{aligned} \tag{47}$$

Note that $\sum_{nm}$ denotes n from 0 to ∞ and m from $-n$ to n. In the same way, the dyadic Green's function in the vacuum can also be expanded by the vector spherical functions. Applying Eq. (47) to Eq. (46) and using the results: $\nabla\times\nabla\times\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)=k^2\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ and $\nabla\times\nabla\times\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)=k^2\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$, we obtain that

$$\begin{aligned}\overline{\overline{\mathbf{G}}}_{\text{vac}}(\mathbf{r},\mathbf{r}',\omega)=\frac{2}{\pi}\int_0^\infty k^2\mathrm{d}k\sum_{nm}(-1)^m\Big[\;&-k_0^{-2}\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{L}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\\&+\frac{1}{k^2-k_0^2}\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\\&+\frac{1}{k^2-k_0^2}\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\;\Big].\end{aligned} \tag{48}$$

To further simplify the free-space dyadic Green's function, we need to evaluate the k-dependent integrals; however, it is inconvenient to manipulate tensor-type integrand. To

[3] The derivation in this section cannot permit the correctness of dyadic Green's function in a complex dielectric environment, which is originated from the restriction of orthogonality (completeness) in Eq. (26).

reduce the complexity, we can change the order of differentiation and integration. First, we consider the integral with respect to $\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$,

$$\begin{aligned}
&\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{-k^2}{k_0^2}\ \sum_{nm}(-1)^m \underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes \underline{\mathbf{L}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \\
&= \frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{-k^2}{k_0^2}\ \sum_{nm}(-1)^m \mathscr{T}_{\mathbf{L}}\left\{\underline{\phi}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\right\}\otimes \mathscr{T}'_{\mathbf{L}}\left\{\underline{\phi}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\right\} \\
&= \nabla\otimes\nabla'\left\{\sum_{nm}\frac{-(-1)^m}{k_0^2 f_{nm}}P_n^{-m}(\cos\theta')P_n^m(\cos\theta)e^{im(\phi-\phi')}\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ j_n(kr')j_n(kr)\right\}. \qquad (49)
\end{aligned}$$

Recall that $\mathscr{T}_{\mathbf{L}}\{g\}\equiv k^{-1}\nabla g$, which is defined in Eq. (11). Also remind that we use the fact that $f_{n(-m)}=f_{nm}$ in Eq. (49). In Appendix E we proved that the integral becomes

$$\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ j_n(kr')j_n(kr)=\frac{1}{(2n+1)}\frac{r_<^n}{r_>^{n+1}}, \qquad (50)$$

where $r_>\equiv\max(\mathbf{r},\mathbf{r}')$ and $r_<\equiv\min(\mathbf{r},\mathbf{r}')$. Plugging Eq. (50) to Eq. (49) and using the identity (Laplace expansion),

$$\begin{aligned}
\frac{1}{4\pi|\mathbf{r}-\mathbf{r}'|} &= \sum_{n=0}^\infty \frac{1}{4\pi}\frac{r_<^n}{r_>^{n+1}}P_n(\cos\Theta), \qquad \cos\Theta = cos\theta'\cos\theta+\sin\theta'\sin\theta\cos(\phi-\phi') \qquad (51)\\
&= \sum_{nm}\frac{(-1)^m}{(2n+1)f_{nm}}\frac{r_<^n}{r_>^{n+1}}P_n^{-m}(\cos\theta')P_n^m(\cos\theta)e^{im(\phi-\phi')}, \qquad (52)
\end{aligned}$$

the part of $\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ in the free-space dyadic Green's function becomes

$$-\frac{2}{\pi}\int_0^\infty \frac{k^2}{k_0^2}\mathrm{d}k\ \sum_{nm}(-1)^m \underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes \underline{\mathbf{L}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') = \nabla\otimes\nabla'\left\{\frac{-1}{4\pi k_0^2|\mathbf{r}-\mathbf{r}'|}\right\}. \qquad (53)$$

It is evident that $\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ are the static-like fields, which play a similar role to the Coulomb interaction. In fact, we anticipate that the static-like fields should vanish because the linear operator $\mathscr{L}_0$ is purely transverse outside the source region. That is to say, for dyadic Green's functions (point-like sources), the region outside the source indicates that $\mathbf{r}\neq\mathbf{r}'$.

The disappearance of static-like fields will be verified after we evaluating the $\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ part. Next, using the same strategy, the integral with respect to $\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ becomes

$$\begin{aligned}
&\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{k^2}{k^2-k_0^2}\ \sum_{nm}(-1)^m \underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes \underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \\
&= \mathscr{T}_{\mathbf{M}}\otimes\mathscr{T}'_{\mathbf{M}}\left\{\sum_{nm}\frac{(-1)^m}{n(n+1)}\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{k^2}{k^2-k_0^2}\ \underline{\phi}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\underline{\phi}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\right\}. \qquad (54)
\end{aligned}$$

Also recall that $\mathscr{T}_{\mathbf{M}}\{g\}\equiv\nabla\times(\mathbf{r}g)$. According to the identity derived in Appendix F,

$$\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{k^2}{k^2-k_0^2}j_n(kr')j_n(kr) = ik_0h_n^{(1)}(k_0r_>)j_n(k_0r_<), \qquad (55)$$

the final result of $\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ part in the free-space dyadic Green's function becomes

$$\begin{aligned}
&\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{k^2}{k^2-k_0^2}\ \sum_{nm}(-1)^m \underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes \underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \\
&= \begin{cases} ik_0\displaystyle\sum_{nm}(-1)^m \underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(k_0r,\theta,\phi)\otimes \underline{\mathbf{M}}^{(\mathrm{III})}_{n(-m)}(k_0r',\theta',\phi'), & r<r' \\ ik_0\displaystyle\sum_{nm}(-1)^m \underline{\mathbf{M}}^{(\mathrm{III})}_{nm}(k_0r,\theta,\phi)\otimes \underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(k_0r',\theta',\phi'), & r>r' \end{cases}. \qquad (56)
\end{aligned}$$

Here, $h_n^{(1)}(x)$ denotes the Hankel function of the first kind, which fulfills the Sommerfeld radiation condition. In other words, $h_n^{(1)}(x)$ can be interpreted as an outgoing spherical wave. The superscript [(III)] denotes the Hankel function of the first kind is adopted. Finally, for the part of $\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$,

$$\begin{aligned}
&\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{k^2}{k^2-k_0^2}\ \sum_{nm}(-1)^m \underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes \underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \\
&= \mathscr{T}_{\mathbf{N}}\otimes\mathscr{T}'_{\mathbf{N}}\left\{\sum_{nm}\frac{(-1)^m}{n(n+1)}\frac{2}{\pi}\int_0^\infty \mathrm{d}k\ \frac{k^2}{k^2-k_0^2}\ \underline{\phi}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\underline{\phi}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\right\} \\
&= \nabla\times\mathscr{T}_{\mathbf{M}}\otimes\nabla'\times\mathscr{T}'_{\mathbf{M}}\left\{\sum_{nm}\frac{(-1)^m}{n(n+1)}\frac{2}{\pi}\int_0^\infty \frac{\mathrm{d}k}{k^2-k_0^2}\ \underline{\phi}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi')\underline{\phi}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\right\}, \qquad (57)
\end{aligned}$$

we utilize the identity (see also in Appendix F),

$$\frac{2}{\pi}\int_0^\infty \frac{j_n(kr')j_n(kr)}{k^2-k_0^2}\,\mathrm{d}k = \frac{i}{k_0}h_n^{(1)}(k_0r_>)j_n(k_0r_<) - \frac{1}{(2n+1)k_0^2}\frac{r_<^n}{r_>^{n+1}} \tag{58}$$

to simplify the result. Recall that $\mathscr{T}_{\mathbf{N}}\{g\} \equiv k^{-1}\nabla\times\nabla\times(\mathbf{r}g) = k^{-1}\nabla\times\mathscr{T}_{\mathbf{M}}\{g\}$. Note that Eq. (58) comprises a dynamic-like and a static-like term. The dynamic-like term returns

$$\begin{aligned}
&\begin{cases}
\nabla\times\mathscr{T}_{\mathbf{M}}\otimes\nabla'\times\mathscr{T}'_{\mathbf{M}}\left\{\dfrac{i}{k_0}\displaystyle\sum_{nm}\frac{(-1)^m}{n(n+1)}\underline{\phi}^{(\mathrm{III})}_{n(-m)}(k_0r',\theta',\phi')\underline{\phi}^{(\mathrm{I})}_{nm}(k_0r,\theta,\phi)\right\}, & r<r'\\
\nabla\times\mathscr{T}_{\mathbf{M}}\otimes\nabla'\times\mathscr{T}'_{\mathbf{M}}\left\{\dfrac{i}{k_0}\displaystyle\sum_{nm}\frac{(-1)^m}{n(n+1)}\underline{\phi}^{(\mathrm{I})}_{n(-m)}(k_0r',\theta',\phi')\underline{\phi}^{(\mathrm{III})}_{nm}(k_0r,\theta,\phi)\right\}, & r>r'
\end{cases}\\
=&\begin{cases}
ik_0\displaystyle\sum_{nm}(-1)^m\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(k_0r,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\mathrm{III})}_{n(-m)}(k_0r',\theta',\phi'), & r<r'\\
ik_0\displaystyle\sum_{nm}(-1)^m\underline{\mathbf{N}}^{(\mathrm{III})}_{nm}(k_0r,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(k_0r',\theta',\phi'), & r>r'
\end{cases},
\end{aligned} \tag{59}$$

which is analog to the result of the $\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(k_0r,\theta,\phi)$ part. On the other hand, the static-like term becomes

$$\begin{cases}
\nabla\times\mathscr{T}_{\mathbf{M}}\otimes\nabla'\times\mathscr{T}'_{\mathbf{M}}\left\{-\dfrac{1}{4\pi k_0^2}\displaystyle\sum_n\frac{1}{n(n+1)}\frac{r^n}{(r')^{n+1}}P_n(\cos\Theta)\right\}, & r<r'\\
\nabla\times\mathscr{T}_{\mathbf{M}}\otimes\nabla'\times\mathscr{T}'_{\mathbf{M}}\left\{-\dfrac{1}{4\pi k_0^2}\displaystyle\sum_n\frac{1}{n(n+1)}\frac{(r')^n}{r^{n+1}}P_n(\cos\Theta)\right\}, & r>r'
\end{cases} \tag{60}$$

$$\begin{cases}
\nabla\otimes\nabla'\left\{\dfrac{1}{4\pi k_0^2}\displaystyle\sum_n\frac{r^n}{(r')^{n+1}}P_n(\cos\Theta)\right\}, & r<r'\\
\nabla\otimes\nabla'\left\{\dfrac{1}{4\pi k_0^2}\displaystyle\sum_n\frac{(r')^n}{r^{n+1}}P_n(\cos\Theta)\right\}, & r>r'
\end{cases} \tag{61}$$

$$=\nabla\otimes\nabla'\left\{\frac{1}{4\pi k_0^2|\mathbf{r}-\mathbf{r}'|}\right\}. \tag{62}$$

In Eq. (60), we use the identity to simplify the summation of m,

$$\frac{1}{4\pi}P_n(\cos\Theta) = \sum_{m=-n}^{n}\frac{(-1)^m}{(2n+1)f_{nm}}P_n^{-m}(\cos\theta')P_n^m(\cos\theta)e^{im(\phi-\phi')}. \tag{63}$$

In addition, we use the vector identity ,$\nabla \times \mathscr{T}_{\mathbf{M}}\{g\} = \nabla[\partial_r(\mathbf{r}g)] - \mathbf{r}\nabla^2 g$, in Eq. (61). Because the Laplacian of the static-like term becomes zero, only the gradient of the static-like term survives. It is obtained that the static-like term in the $\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ part [Eq. (62)] and that in the $\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ part [Eq. (53)] cancel each other. Therefore, for $r \neq r'$, we get the result that

$$\overline{\overline{\mathbf{G}}}_{\mathrm{vac}}(\mathbf{r},\mathbf{r}',\omega) = \begin{cases} ik_0 \sum_{nm} (-1)^m \Big[\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\mathrm{III})}_{n(-m)}(k_0 r',\theta',\phi') \\ \qquad\qquad + \underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\mathrm{III})}_{n(-m)}(k_0 r',\theta',\phi') \Big], \quad r < r' \\ ik_0 \sum_{nm} (-1)^m \Big[\underline{\mathbf{M}}^{(\mathrm{III})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(k_0 r',\theta',\phi') \\ \qquad\qquad + \underline{\mathbf{N}}^{(\mathrm{III})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(k_0 r',\theta',\phi') \Big], \quad r > r' \end{cases} \tag{64}$$

When $r = r'$, the free-space dyadic Green's function behaves like the a delta function,

$$\overline{\overline{\mathbf{G}}}_{\mathrm{vac}}(\mathbf{r},\mathbf{r}',\omega) = \frac{1}{k_0^2}\delta(\mathbf{r}-\mathbf{r}')\hat{r}\hat{r} \tag{65}$$

according to the definition of dyadic Green's functions [i.e., Eq. (46)]. Here, we will not spend efforts to derive it because the free-space dyadic Green's function is useless in practice.

Electric Fields in the Language of Free-Space Green's Functions

Before we explore the electric field in a spherical-scatterer system, we discuss that in vacuum first. In this subsection, we discuss two typical sources in optics, plane wave and electric point dipole.

Plane Wave

A plane-wave source in Mie thoery is used to evaluate the optical properties (e.g. scattering and absorption cross section) of a particle. As mentioned in Eq. (8), the electric field comprises a homogeneous solution and an inhomogeneous solution. It is obvious that a plane wave is a homogeneous solution. In the spherical coordinate, a plane wave becomes

$$\mathbf{E}^{(0)}(\mathbf{r},\omega) = \sum_{nm}\left[q_{nm,\mathrm{pw}}\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(k_0 r,\theta,\phi) + p_{nm,\mathrm{pw}}\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(k_0 r,\theta,\phi)\right], \tag{66}$$

where $q_{nm,\mathrm{pw}}$ and $p_{nm,\mathrm{pw}}$ are the coefficients which are expressed by

$$q_{nm,\mathrm{pw}} = 4\pi(-1)^m i^n \mathbf{E}_0 \cdot \mathbf{C}(\theta) e^{im\phi'} \tag{67}$$

$$p_{nm,\mathrm{pw}} = 4\pi(-1)^m i^n \mathbf{E}_0 \cdot \mathbf{C}(\theta) e^{im\phi'} \tag{68}$$

For a x-polarized plane wave,

$$\mathbf{E}_{\mathrm{pw}} = E_0 e^{ikr\cos\theta}\hat{\mathbf{e}}_x = E_0 e^{ikr\cos\theta}\left(\sin\theta\cos\phi\hat{\mathbf{e}}_r + \cos\theta\cos\phi\hat{\mathbf{e}}_\theta + \sin\phi\hat{\mathbf{e}}_\phi\right) \tag{69}$$

For alternative method, see the reference[2]

Electric Point Dipole

For an electric point dipole located at $\mathbf{r}_{\mathrm{D}}$ in the zeroth region (vacuum) without any additional structure, the polarization field in Eq. (5) becomes

$$\sum_j \mathbf{P}_{\mathrm{D}}^{(j)}(\mathbf{r}',\omega) \to \mathbf{P}_{\mathrm{D}}^{(0)}(\mathbf{r}',\omega) = \mathbf{p}_{\mathrm{D}}(\omega)\delta(\mathbf{r}'-\mathbf{r}_{\mathrm{D}}) \tag{70}$$

In the spherical coordinate, the electric field reads

$$\begin{aligned}
\mathbf{E}^{(0)}(\mathbf{r},\omega) &= \frac{\omega^2}{c^2\epsilon_0}\int \overline{\overline{\mathbf{G}}}_{\mathrm{vac}}(\mathbf{r},\mathbf{r}',\omega)\cdot\mathbf{p}_{\mathrm{D}}^{(0)}(\omega)\delta(\mathbf{r}'-\mathbf{r}_{\mathrm{D}})\ \mathrm{d}^3\mathbf{r}' \\
&= \frac{\omega^2}{c^2\epsilon_0}\overline{\overline{\mathbf{G}}}_{\mathrm{vac}}(\mathbf{r},\mathbf{r}_{\mathrm{D}},\omega)\cdot\mathbf{p}_{\mathrm{D}}^{(0)}(\omega) \\
&= \begin{cases} \displaystyle\sum_{nm}\left[q_{nm}\underline{\mathbf{M}}_{nm}^{(\mathrm{I})}(k_0r,\theta,\phi)+p_{nm}\underline{\mathbf{N}}_{nm}^{(\mathrm{I})}(k_0r,\theta,\phi)\right], & r<r_{\mathrm{D}} \\ \displaystyle\sum_{nm}\left[s_{nm}\underline{\mathbf{M}}_{nm}^{(\mathrm{III})}(k_0r,\theta,\phi)+r_{nm}\underline{\mathbf{N}}_{nm}^{(\mathrm{III})}(k_0r,\theta,\phi)\right], & r>r_{\mathrm{D}} \end{cases}
\end{aligned} \tag{71}$$

where

$$p_{nm} = ik_0^3\epsilon_0^{-1}(-1)^m\underline{\mathbf{N}}_{n(-m)}^{(\mathrm{III})}(k_0r_{\mathrm{D}},\theta_{\mathrm{D}},\phi_{\mathrm{D}})\cdot\mathbf{p}_{\mathrm{D}}^{(0)}(\omega), \tag{72a}$$

$$q_{nm} = ik_0^3\epsilon_0^{-1}(-1)^m\underline{\mathbf{M}}_{n(-m)}^{(\mathrm{III})}(k_0r_{\mathrm{D}},\theta_{\mathrm{D}},\phi_{\mathrm{D}})\cdot\mathbf{p}_{\mathrm{D}}^{(0)}(\omega), \tag{72b}$$

$$r_{nm} = ik_0^3\epsilon_0^{-1}(-1)^m\underline{\mathbf{N}}_{n(-m)}^{(\mathrm{I})}(k_0r_{\mathrm{D}},\theta_{\mathrm{D}},\phi_{\mathrm{D}})\cdot\mathbf{p}_{\mathrm{D}}^{(0)}(\omega), \tag{72c}$$

$$s_{nm} = ik_0^3\epsilon_0^{-1}(-1)^m\underline{\mathbf{M}}_{n(-m)}^{(\mathrm{I})}(k_0r_{\mathrm{D}},\theta_{\mathrm{D}},\phi_{\mathrm{D}})\cdot\mathbf{p}_{\mathrm{D}}^{(0)}(\omega), \tag{72d}$$

can be viewed as the expansion coefficients of the electric point dipole.

Spheres and Electromagnetic Boundary conditions

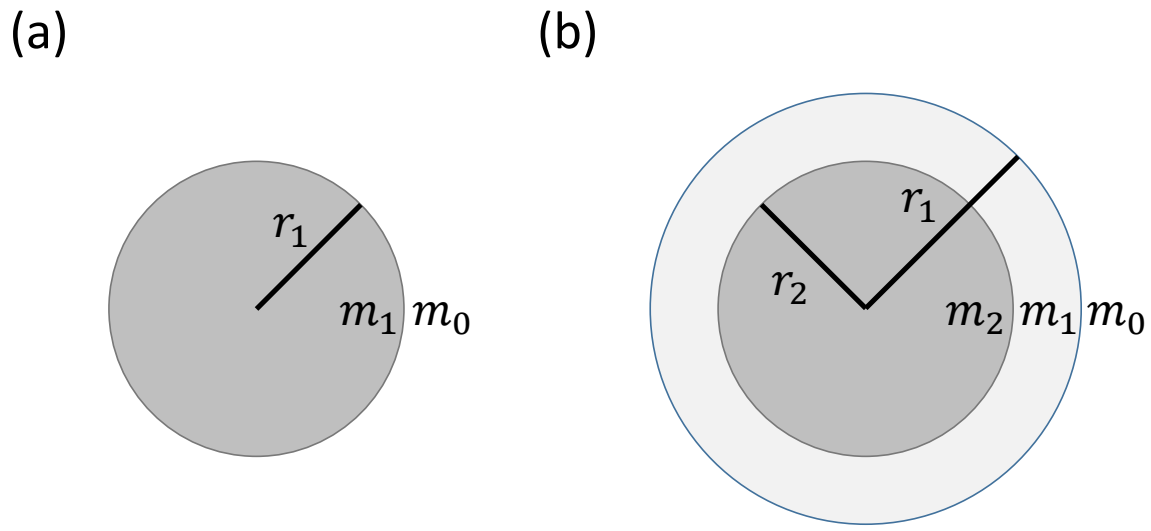

Figure 1: The structures of (a) single sphere and (b) core/shell sphere. r_1 represents the radius of the outer boundary, and r_2 represents the radius of the inner boundary (for the core/shell sphere). The dielectric function in each region is idealized to be homogeneous, i.e., independent of position.

In this subsection, we focus on the electric fields in the two systems, (a) single sphere and (b) core/shell sphere, as shown in Fig. 1, in order to solve the dyadic Green's functions, $\overline{\overline{\mathbf{G}}}^{(00)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega)$, $\overline{\overline{\mathbf{G}}}^{(00)}_{\text{scat}}(\mathbf{r},\mathbf{r}',\omega)$, and $\overline{\overline{\mathbf{G}}}^{(10)}(\mathbf{r},\mathbf{r}',\omega)$. As mentioned in Eq. (5), we aim to solve the inhomogeneous vector differential equation,

$$\left[\frac{\omega^2\epsilon_{\text{r},i}(\omega)}{c^2} - \nabla\times\nabla\times\right]\mathbf{E}^{(i)}(\mathbf{r},\omega) = -\frac{\omega^2}{\epsilon_0 c^2}\sum_j \mathbf{P}^{(j)}_{\text{D}}(\mathbf{r},\omega).$$

According to the systems illustrated in Fig. 1, the dielectric function of the two systems are shown in the following table. In addition, the polarization $\mathbf{P}^{(j)}_{\text{D}}(\mathbf{r},\omega)$ is expressed as

$$\sum_j \mathbf{P}^{(j)}_{\text{D}}(\mathbf{r},\omega) \to \mathbf{P}^{(0)}_{\text{D}}(\mathbf{r},\omega) = \mathbf{p}_{\text{D}}(\omega)\delta(\mathbf{r}-\mathbf{r}_{\text{D}}).$$

Table 1: Dielectric function of the two systems. Note that $\epsilon_{\mathrm{r}}(\mathbf{r},\omega)$ is piecewise-homogeneous.

	Single Sphere		Core/Shell Sphere	
	$\epsilon_{\mathrm{r},0}(\omega)=1$	$r>r_1$	$\epsilon_{\mathrm{r},0}(\omega)=1$	$r>r_1$
$\epsilon_{\mathrm{r}}(\mathbf{r},\omega)$	$\epsilon_{\mathrm{r},1}(\omega)$	$r_1>r>0$	$\epsilon_{\mathrm{r},1}(\omega)$	$r_1>r>r_2$
	N/A	N/A	$\epsilon_{\mathrm{r},2}(\omega)$	$r_2>r>0$

In Eqs. (71) and (72), the electric point dipole are expanded to vector spherical functions. The next procedure is solving the scattering process via considering the electromagnetic boundary condition. In electrodynamics, the electric and magnetic field obey the four conditions for each boundary:

$$\mathbf{n}_i \times \left[\mathbf{E}^{(i)}(\mathbf{r}_i,\omega)-\mathbf{E}^{(i-1)}(\mathbf{r}_i,\omega)\right]=0, \tag{73a}$$

$$\mathbf{n}_i \cdot \left[\mathbf{D}^{(i)}(\mathbf{r}_i,\omega)-\mathbf{D}^{(i-1)}(\mathbf{r}_i,\omega)\right]=\sigma_s, \tag{73b}$$

$$\mathbf{n}_i \cdot \left[\mathbf{B}^{(i)}(\mathbf{r}_i,\omega)-\mathbf{B}^{(i-1)}(\mathbf{r}_i,\omega)\right]=0, \tag{73c}$$

$$\mathbf{n}_i \times \left[\mathbf{H}^{(i)}(\mathbf{r}_i,\omega)-\mathbf{H}^{(i-1)}(\mathbf{r}_i,\omega)\right]=\mathbf{j}_s. \tag{73d}$$

where $i=1$ for the single sphere system and $i=1,\ 2$ for the core/shell system. $\mathbf{n}_l$ is the normal vector of the boundary surface between l^{th} region and $(l-1)^{\mathrm{th}}$ region. σ_{s} and $\mathbf{j}_{\mathrm{s}}$ denote the surface charge and the surface current density, respectively. $\mathbf{r}_l$ describes the position vector of boundary, and ω represents the angular frequency. Note that the bounded surface charge and bounded surface current density are equal to zero, in our cases. In the following two subsections, we will individually discuss the two cases. Specifically, the electric field is projected on the vector spherical functions and solve the corresponding coefficients by Eqs. (73a) to (73d). Additional details can be found in the textbook.[2]

Single Sphere

For a single sphere illustrated in Figure 1a, the electric field for each region is written as

$$\mathbf{E}^{(0)}(\mathbf{r},\omega) = \mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega) + \mathbf{E}^{(0)}_{\text{scat}}(\mathbf{r},\omega), \tag{74a}$$

$$\mathbf{E}^{(1)}(\mathbf{r},\omega) = \mathbf{E}^{(1)}_{\text{core}}(\mathbf{r},\omega). \tag{74b}$$

In the zeroth region, the total electric field is separated into two parts, the source part $\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega)$ and the scattering part $\mathbf{E}^{(0)}_{\text{scat}}(\mathbf{r},\omega)$, in order to distinguish incident processes and scattering processes. In the first region, $\mathbf{E}^{(1)}_{\text{core}}(\mathbf{r},\omega)$ describes the transmitted electric field in the core. For the electric fields, they can be projected on the two spherical vector functions, $\underline{\mathbf{N}}^{(j)}_{nm}(k_i r,\theta,\phi)$ and $\underline{\mathbf{M}}^{(j)}_{nm}(k_i r,\theta,\phi)$,

$$\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega) = \sum_{nm}\left[r_{nm}\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) + s_{nm}\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r > r_{\text{D}} \tag{75a}$$

$$\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_0 r,\theta,\phi) + q_{nm}\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r_1 < r < r_{\text{D}} \tag{75b}$$

$$\mathbf{E}^{(0)}_{\text{scat}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\alpha^{(0)}_n\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) + q_{nm}\beta^{(0)}_n\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r > r_1 \tag{75c}$$

$$\mathbf{E}^{(1)}_{\text{core}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\delta^{(1)}_n\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) + q_{nm}\gamma^{(1)}_n\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi)\right], \qquad 0 < r < r_1. \tag{75d}$$

where the summation runs from $n = 1$ to ∞ and from $m = -n$ to n. To write down Eq. (75), the following four things should be explained. First, we do not consider $\underline{\mathbf{L}}^{(j)}_{nm}(k_i r,\theta,\phi)$ since the coefficients of the source (electric point dipole) to $\underline{\mathbf{L}}^{(j)}_{nm}(k_i r,\theta,\phi)$ are zero. Also note that the superscript of spherical vector functions [$\underline{\mathbf{N}}^{(j)}_{nm}(k_i r,\theta,\phi)$ and $\underline{\mathbf{M}}^{(j)}_{nm}(k_i r,\theta,\phi)$] indicates the type of spherical Bessel function used in the radial part rather than the region of piecewise dielectric environment. Specifically, $j = \text{I}$ indicates the use of the spherical Bessel functions and $j = \text{III}$ indicates the use of the spherical Hankel functions of the first kind. Second, for a second-order differential equation, there are two independent solutions for the radial function. We use spherical Bessel functions and spherical Hankel functions of the first kind as the two

linear independent solutions to describe $\mathbf{E}^{(0)}(\mathbf{r},\omega)$. However, we only use spherical Bessel functions to describe $\mathbf{E}^{(1)}(\mathbf{r},\omega)$ because spherical Hankel functions of the first kind diverged at zero and give nonphysical results. Third, $\alpha_n^{(0)}$, $\beta_n^{(0)}$, $\gamma_n^{(1)}$, and $\delta_n^{(1)}$ are the Mie coefficients of an single sphere. Here, $\alpha_n^{(0)}$ and $\beta_n^{(0)}$ can be interpreted as the reflective coefficients of TM fields and TE fields respectively; $\gamma_n^{(1)}$ and $\delta_n^{(1)}$ can be interpreted as the transmission coefficients TE fields and TM fields respectively. Fourth, the Mie coefficients are only depend on n due to the electromagnetic boundary condition in Eq. (73). Comparing the vector spherical functions of the two regions, the only different variable is $k_i r$, which only influences on radial functions. Therefore, Mie coefficients only associated with n. Additionally, the magnetic fields are deduced by the relation: $i\omega\mu_0\mathbf{H}^{(i)}(\mathbf{r}_i,\omega) = \nabla\times\mathbf{E}^{(i)}(\mathbf{r}_i,\omega)$. By considering Eqs. (73a) - (73d) at $r_i = r_1$, we get the four equations,

$$\begin{aligned}
\underline{n}_0\psi_n'(\underline{n}_1\rho_1)\delta_n^{(1)} &= \underline{n}_1\xi_n'(\underline{n}_0\rho_1)\alpha_n^{(0)} + \underline{n}_1\psi_n'(\underline{n}_0\rho_1)\\
\psi_n(\underline{n}_1\rho_1)\delta_n^{(1)} &= \xi_n(\underline{n}_0\rho_1)\alpha_n^{(0)} + \psi_n(\underline{n}_0\rho_1)\\
\underline{n}_0\psi_n(\underline{n}_1\rho_1)\gamma_n^{(1)} &= \underline{n}_1\xi_n(\underline{n}_0\rho_1)\beta_n^{(0)} + \underline{n}_1\psi_n(\underline{n}_0\rho_1)\\
\psi_n'(\underline{n}_1\rho_1)\gamma_n^{(1)} &= \xi_n'(\underline{n}_0\rho_1)\beta_n^{(0)} + \psi_n'(\underline{n}_0\rho_1)
\end{aligned}, \tag{76}$$

where $\xi_n(\underline{n}_0\rho_1) = \underline{n}_0\rho_1\cdot h_n^{(1)}(\underline{n}_0\rho_1)$ is the n-th order of Riccati-Hankel functions of the first kind and $\rho_1 \equiv k_0 r_1$. Recall that $k_i = \underline{n}_i k_0$ and $\underline{n}_i$ is the complex refractive index of the i-th region. Although $\underline{n}_0 = 1$ in the zeroth region, we still use $\underline{n}_0$ in order to keep the symmetry of equations. Solving the linear equations, we get the analytical expression of Mie coefficients,

$$\alpha_n^{(0)} = -\frac{\underline{n}_1\psi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\psi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}{\underline{n}_1\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}, \tag{77a}$$

$$\beta_n^{(0)} = -\frac{\underline{n}_0\psi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_1\psi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}{\underline{n}_0\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_1\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}, \tag{77b}$$

$$\gamma_n^{(1)} = \frac{\underline{n}_1\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_0\rho_1) - \underline{n}_1\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_0\rho_1)}{\underline{n}_0\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_1\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}, \tag{77c}$$

$$\delta_n^{(1)} = \frac{\underline{n}_1\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_0\rho_1) - \underline{n}_1\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_0\rho_1)}{\underline{n}_1\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}. \tag{77d}$$

Applying Eq. (77) to Eq. (75), we are able to calculate the electric fields for each region now. Furthermore, because the electric fields are equivalent to the Green's functions, we can also obtain the explicit expression of the dyadic Green's functions by the Mie coefficients,

$$\overline{\overline{\mathbf{G}}}^{(00)}(\mathbf{r},\mathbf{r}',\omega) = \overline{\overline{\mathbf{G}}}^{(00)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega) + \overline{\overline{\mathbf{G}}}^{(00)}_{\text{scat}}(\mathbf{r},\mathbf{r}',\omega) \tag{78}$$

$$\begin{aligned}\overline{\overline{\mathbf{G}}}^{(10)}(\mathbf{r},\mathbf{r}',\omega) = ik_0 \sum_{nm}(-1)^m \Big[&\gamma_n^{(1)} \underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \\ &+\delta_n^{(1)} \underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi')\Big],\end{aligned} \tag{79}$$

where $\overline{\overline{\mathbf{G}}}^{(00)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega)$ and $\overline{\overline{\mathbf{G}}}^{(00)}_{\text{scat}}(\mathbf{r},\mathbf{r}',\omega)$ are

$$\overline{\overline{\mathbf{G}}}^{(00)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega) = \begin{cases} ik_0 \sum_{nm}(-1)^m \Big[\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \\ \qquad +\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi')\Big], & r < r' \\ ik_0 \sum_{nm}(-1)^m \Big[\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{I})}_{n(-m)}(k_0 r',\theta',\phi') \\ \qquad +\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{I})}_{n(-m)}(k_0 r',\theta',\phi')\Big], & r > r' \end{cases}, \tag{80}$$

and

$$\begin{aligned}\overline{\overline{\mathbf{G}}}^{(00)}_{\text{scat}}(\mathbf{r},\mathbf{r}',\omega) = ik_0 \sum_{nm}(-1)^m \Big[&\beta_n^{(0)} \underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \\ &+\alpha_n^{(0)} \underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi')\Big],\end{aligned} \tag{81}$$

respectively.

Core/Shell Sphere

For a core/shell sphere system illustrated in Figure 1b, we need to deal with two boundary constraints. Hence, there are three regions for the electric field:

$$\mathbf{E}^{(0)}(\mathbf{r},\omega) = \mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega) + \mathbf{E}^{(0)}_{\text{scat}}(\mathbf{r},\omega), \tag{82a}$$

$$\mathbf{E}^{(1)}(\mathbf{r},\omega) = \mathbf{E}^{(1)}_{\text{shell}}(\mathbf{r},\omega), \tag{82b}$$

$$\mathbf{E}^{(2)}(\mathbf{r},\omega) = \mathbf{E}^{(2)}_{\text{core}}(\mathbf{r},\omega). \tag{82c}$$

Using the same strategy as explained in the previous subsection, we can express the electric fields for each region by

$$\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega) = \sum_{nm}\left[r_{nm}\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) + s_{nm}\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r > r_{\text{D}} \tag{83a}$$

$$\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_0 r,\theta,\phi) + q_{nm}\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r_1 < r < r_{\text{D}} \tag{83b}$$

$$\mathbf{E}^{(0)}_{\text{scat}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\alpha^{(0)}_n\mathbf{N}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) + q_{nm}\beta^{(0)}_n\mathbf{M}^{(\text{III})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r > r_1 \tag{83c}$$

$$\begin{aligned}\mathbf{E}^{(1)}_{\text{shell}}(\mathbf{r},\omega) = \sum_{nm}\big[&p_{nm}\alpha^{(1)}_n\mathbf{N}^{(\text{III})}_{nm}(k_1 r,\theta,\phi) + q_{nm}\beta^{(1)}_n\mathbf{M}^{(\text{III})}_{nm}(k_1 r,\theta,\phi)\\ &+ p_{nm}\delta^{(1)}_n\mathbf{N}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) + q_{nm}\gamma^{(1)}_n\mathbf{M}^{(\text{I})}_{nm}(k_1 r,\theta,\phi)\big], \qquad r_2 < r < r_1\end{aligned} \tag{83d}$$

$$\mathbf{E}^{(2)}_{\text{core}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\delta^{(2)}_n\mathbf{N}^{(\text{I})}_{nm}(k_2 r,\theta,\phi) + q_{nm}\gamma^{(2)}_n\mathbf{M}^{(\text{I})}_{nm}(k_2 r,\theta,\phi)\right], \qquad 0 < r < r_2, \tag{83e}$$

Especially, in the location of the shell, we use both Bessel-type functions and Hankel-type functions to describe the electric field because both of them are not suppressed by boundary conditions. In the same way, the magnetic fields are defined according to the relation, $i\omega\mu_0\mathbf{H}^{(i)}(\mathbf{r}_i,\omega) = \nabla\times\mathbf{E}^{(i)}(\mathbf{r}_i,\omega)$. By considering the boundary conditions in Eqs. (73a) -

(73d) at $r_i = r_1$ and r_2 , we obtain the eight equations,

$$\begin{array}{rcrcrcl}
\underline{n}_1\psi_n'(\underline{n}_2\rho_2)\delta_n^{(2)} & = & \underline{n}_2\xi_n'(\underline{n}_1\rho_2)\alpha_n^{(1)} & + & \underline{n}_2\psi_n'(\underline{n}_1\rho_2)\delta_n^{(1)} \\
\psi_n(\underline{n}_2\rho_2)\delta_n^{(2)} & = & \xi_n(\underline{n}_1\rho_2)\alpha_n^{(1)} & + & \psi_n(\underline{n}_1\rho_2)\delta_n^{(1)} \\
\underline{n}_1\psi_n(\underline{n}_2\rho_2)\gamma_n^{(2)} & = & \underline{n}_2\xi_n(\underline{n}_1\rho_2)\beta_n^{(1)} & + & \underline{n}_2\psi_n(\underline{n}_1\rho_2)\gamma_n^{(1)} \\
\psi_n'(\underline{n}_2\rho_2)\gamma_n^{(2)} & = & \xi_n'(\underline{n}_1\rho_2)\beta_n^{(1)} & + & \psi_n'(\underline{n}_1\rho_2)\gamma_n^{(1)} \\
\underline{n}_0\xi_n'(\underline{n}_1\rho_1)\alpha_n^{(1)} + \underline{n}_0\psi_n'(\underline{n}_1\rho_1)\delta_n^{(1)} & = & \underline{n}_1\xi_n'(\underline{n}_0\rho_1)\alpha_n^{(0)} & + & \underline{n}_1\psi_n'(\underline{n}_0\rho_1) \\
\xi_n(\underline{n}_1\rho_1)\alpha_n^{(1)} + \psi_n(\underline{n}_1\rho_1)\delta_n^{(1)} & = & \xi_n(\underline{n}_0\rho_1)\alpha_n^{(0)} & + & \psi_n(\underline{n}_0\rho_1) \\
\underline{n}_0\xi_n(\underline{n}_1\rho_1)\beta_n^{(1)} + \underline{n}_0\psi_n(\underline{n}_1\rho_1)\gamma_n^{(1)} & = & \underline{n}_1\xi_n(\underline{n}_0\rho_1)\beta_n^{(0)} & + & \underline{n}_1\psi_n(\underline{n}_0\rho_1) \\
\xi_n'(\underline{n}_1\rho_1)\beta_n^{(1)} + \psi_n'(\underline{n}_1\rho_1)\gamma_n^{(1)} & = & \xi_n'(\underline{n}_0\rho_1)\beta_n^{(0)} & + & \psi_n'(\underline{n}_0\rho_1)
\end{array} , \tag{84}$$

where $\rho_2 = k_0 r_2$ and $\rho_1 = k_0 r_1$. Note that the eight equation can be separated to two set of simultaneous equations which is guaranteed by the orthogonality of $\underline{\mathbf{M}}_{nm}^{(j)}(k_i r, \theta, \phi)$ and $\underline{\mathbf{N}}_{nm}^{(j)}(k_i r, \theta, \phi)$. Furthermore, for the concern of numerical stability, we substitute the derivatives of Riccati-Bessel (-Hankel) functions for logarithmic derivatives,

$$\mathscr{D}\psi_n(z) = \frac{\mathrm{d}}{\mathrm{d}z}\ln\psi_n(z) = \frac{\psi_n'(z)}{\psi_n(z)}, \tag{85a}$$

$$\mathscr{D}\xi_n(z) = \frac{\mathrm{d}}{\mathrm{d}z}\ln\xi_n(z) = \frac{\xi_n'(z)}{\xi_n(z)}. \tag{85b}$$

It has been mentioned that the logarithmic derivative of Riccati-Bessel (-Hankel) function is numerically more stable than the derivative of Riccati-Bessel (-Hankel) function.[3] Solving the above equations, the Mie scattering coefficients turn out to:

$$\alpha_n^{(0)} = \frac{\mathscr{A}_1 - \mathscr{A}}{\mathscr{A}_2 - \mathscr{A}} \cdot \frac{\underline{n}_0\mathscr{D}\psi_n(\underline{n}_1\rho_1) - \underline{n}_1\mathscr{D}\psi_n(\underline{n}_0\rho_1)}{\underline{n}_1\mathscr{D}\xi_n(\underline{n}_0\rho_1) - \underline{n}_0\mathscr{D}\psi_n(\underline{n}_1\rho_1)} \cdot \frac{\psi_n(\underline{n}_0\rho_1)}{\xi_n(\underline{n}_0\rho_1)}, \tag{86a}$$

$$\beta_n^{(0)} = \frac{\mathscr{B}_1 - \mathscr{B}}{\mathscr{B}_2 - \mathscr{B}} \cdot \frac{\underline{n}_0\mathscr{D}\psi_n(\underline{n}_0\rho_1) - \underline{n}_1\mathscr{D}\psi_n(\underline{n}_1\rho_1)}{\underline{n}_1\mathscr{D}\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\mathscr{D}\xi_n(\underline{n}_0\rho_1)} \cdot \frac{\psi_n(\underline{n}_0\rho_1)}{\xi_n(\underline{n}_0\rho_1)}, \tag{86b}$$

where

$$\mathscr{A} = \frac{\underline{n}_2 \mathscr{D}\xi_n(\underline{n}_1\rho_2) - \underline{n}_1 \mathscr{D}\psi_n(\underline{n}_2\rho_2)}{\underline{n}_1 \mathscr{D}\psi_n(\underline{n}_2\rho_2) - \underline{n}_2 \mathscr{D}\psi_n(\underline{n}_1\rho_2)} \cdot \frac{\xi_n(\underline{n}_1\rho_2)}{\psi_n(\underline{n}_1\rho_2)}, \tag{87a}$$

$$\mathscr{B} = \frac{\underline{n}_2 \mathscr{D}\psi_n(\underline{n}_2\rho_2) - \underline{n}_1 \mathscr{D}\xi_n(\underline{n}_1\rho_2)}{\underline{n}_1 \mathscr{D}\psi_n(\underline{n}_1\rho_2) - \underline{n}_2 \mathscr{D}\psi_n(\underline{n}_2\rho_2)} \cdot \frac{\xi_n(\underline{n}_1\rho_2)}{\psi_n(\underline{n}_1\rho_2)}, \tag{87b}$$

$$\mathscr{A}_1 = \frac{\underline{n}_1 \mathscr{D}\psi_n(\underline{n}_0\rho_1) - \underline{n}_0 \mathscr{D}\xi_n(\underline{n}_1\rho_1)}{\underline{n}_0 \mathscr{D}\psi_n(\underline{n}_1\rho_1) - \underline{n}_1 \mathscr{D}\psi_n(\underline{n}_0\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\psi_n(\underline{n}_1\rho_1)}, \tag{87c}$$

$$\mathscr{A}_2 = \frac{\underline{n}_0 \mathscr{D}\xi_n(\underline{n}_1\rho_1) - \underline{n}_1 \mathscr{D}\xi_n(\underline{n}_0\rho_1)}{\underline{n}_1 \mathscr{D}\xi_n(\underline{n}_0\rho_1) - \underline{n}_0 \mathscr{D}\psi_n(\underline{n}_1\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\psi_n(\underline{n}_1\rho_1)}, \tag{87d}$$

$$\mathscr{B}_1 = \frac{\underline{n}_1 \mathscr{D}\xi_n(\underline{n}_1\rho_1) - \underline{n}_0 \mathscr{D}\psi_n(\underline{n}_0\rho_1)}{\underline{n}_0 \mathscr{D}\psi_n(\underline{n}_0\rho_1) - \underline{n}_1 \mathscr{D}\psi_n(\underline{n}_1\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\psi_n(\underline{n}_1\rho_1)}, \tag{87e}$$

$$\mathscr{B}_2 = \frac{\underline{n}_0 \mathscr{D}\xi_n(\underline{n}_0\rho_1) - \underline{n}_1 \mathscr{D}\xi_n(\underline{n}_1\rho_1)}{\underline{n}_1 \mathscr{D}\psi_n(\underline{n}_1\rho_1) - \underline{n}_0 \mathscr{D}\xi_n(\underline{n}_0\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\psi_n(\underline{n}_1\rho_1)}. \tag{87f}$$

The calculations of the logarithmic derivatives of Riccati-Bessel (Hankel) functions can be found in the previous study.[3] Using the same strategy described in the section on a single sphere, we would obtain the explicit expression of the dyadic Green's functions,

$$\overline{\overline{\mathbf{G}}}^{(00)}(\mathbf{r},\mathbf{r}',\omega) = \overline{\overline{\mathbf{G}}}^{(00)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega) + \overline{\overline{\mathbf{G}}}^{(00)}_{\text{scat}}(\mathbf{r},\mathbf{r}',\omega) \tag{88}$$

$$\begin{aligned}\overline{\overline{\mathbf{G}}}^{(10)}(\mathbf{r},\mathbf{r}',\omega) = ik_0 \sum_{nm}(-1)^m \Big[&\beta_n^{(1)} \underline{\mathbf{M}}^{(\text{III})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \\ &+\alpha_n^{(1)} \underline{\mathbf{N}}^{(\text{III})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \qquad (89)\\ &+\gamma_n^{(1)} \underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \qquad (90)\\ &+\delta_n^{(1)} \underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \Big], \qquad (91)\end{aligned}$$

$$\begin{aligned}\overline{\overline{\mathbf{G}}}^{(20)}(\mathbf{r},\mathbf{r}',\omega) = ik_0 \sum_{nm}(-1)^m \Big[&\gamma_n^{(2)} \underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \\ &+\delta_n^{(2)} \underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) \otimes \underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_0 r',\theta',\phi') \Big], \qquad (92)\end{aligned}$$

where $\overline{\overline{\mathbf{G}}}^{(00)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega)$ and have the same form as described in Eqs. (80) and (81); the only difference is the scattering coefficients, $\alpha_n^{(0)}$ and $\beta_n^{(0)}$.

Part II. Implementation

In Part II, we will describe the calculation of electric fields in a system that includes an electric point dipole and a spherical scatterer. As described in Eqs. (74a) and (82a), the total electric field in the zeroth region is the sum of $\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega)$ and $\mathbf{E}^{(0)}_{\text{scat}}(\mathbf{r},\omega)$. Hence, calculations of the total electric fields in the zeroth region can be divided into four critical processes, as illustrated conceptually in Fig. 2. Before we dive into the details of the whole computation flow, the two numerical issues we need to handle in priority: singularity of free-space dyadic Green's functions [Eq. (65)] and numerical precision of vector spherical functions. In the following two subsections, we will describe the numerical schemes and their corresponding implementation (in MATLAB R2020a) to these issues.

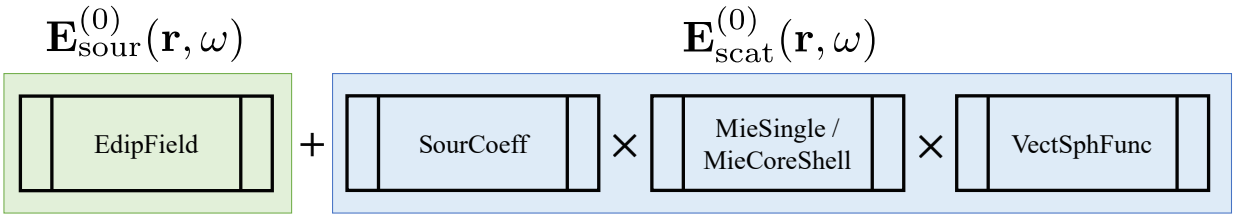

Figure 2: Conceptual illustration of calculating the total electric field in the zeroth region. `EdipField`, `SourCoeff`, `MieSingle`/`MieCoreShell`, and `VectSphFunc` are the MATLAB functions. These functions will be introduced afterward.

Singularity of Free-Space Dyadic Green's Function

In Eq. (71), we have shown the electric field of an electric point dipole can be expressed by

$$\mathbf{E}^{(0)}_{\text{sour}}(\mathbf{r},\omega)=\begin{cases}\displaystyle\sum_{nm}\left[q_{nm}\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_0r,\theta,\phi)+p_{nm}\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_0r,\theta,\phi)\right], & r<r_{\text{D}}\\ \displaystyle\sum_{nm}\left[s_{nm}\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0r,\theta,\phi)+r_{nm}\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0r,\theta,\phi)\right], & r>r_{\text{D}}\end{cases}.$$

However, from the viewpoint of numerical calculations, the implementation based on Eq. (71) is pathological for the following two reasons. First, the expression of Eq. (71) cannot de-

scribe points when $r = r_{\mathrm{D}}$ because these points are singularity. Second, despite Eq. (71) is analytical, a summation of infinite series is impossible in numerical calculations. Moreover, a truncation inevitably leads to the Gibbs phenomenon. Thus, Eq. (71) cannot be directly implemented in programs. The expression of dipole field in Eq. (71) originates from expanding the field at an improper origin. To deal with this issue, we describe the dipole field by choosing a secondary coordinate (the blue frame) which makes $\mathrm{r}'_{\mathrm{D}} \to 0$, as shown in Fig. 3. Here, we add a prime to denote variables under the secondary coordinate, e.g., $\mathbf{r}'_{\mathrm{D}}$. Moreover, in the secondary coordinate, we additionally require r'_{D} approaches to zero along the z axis, i.e., $(r'_{\mathrm{D}}, \theta'_{\mathrm{D}}, \phi'_{\mathrm{D}}) \to (0, 0, 0)$, in order to define and apply the direction of spherical coordinate at the origin. The advantages of describing the dipole field in the secondary coordinate, the singularity is shrunk to $r'_{\mathrm{D}} = 0$ and the summation of infinite series becomes a summation of finite series under the secondary coordinate. To acquire the dipole field expressed in the

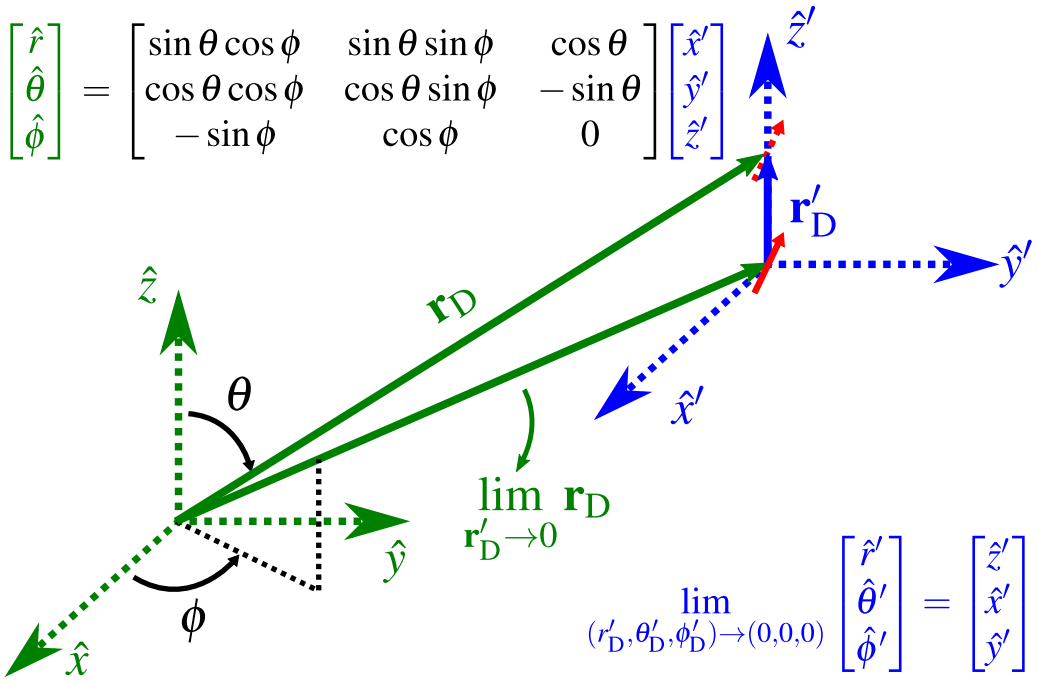

Figure 3: Primary coordinate (green) and secondary coordinate (blue). Variables with prime is in the representation of the secondary coordinate. The transformation of unit vectors between two coordinates is expressed in the upper left corner. Here, $\hat{r}'$, $\hat{\theta}'$, and $\hat{\phi}'$ correspond to $\hat{z}'$, $\hat{x}'$, and $\hat{y}'$ under the limit.

primary coordinate, all we need is to evaluate the coefficients

$$\lim_{\mathbf{r}'_{\rm D}\to 0} r_{nm} = ik_0^3\epsilon_0^{-1}(-1)^m \mathbf{p}_{\rm D}^{(0)}(\omega)\cdot \lim_{\mathbf{r}'_{\rm D}\to 0} \underline{\mathbf{N}}^{(\rm I)}_{n(-m)}(k_0 r'_{\rm D},\theta'_{\rm D},\phi'_{\rm D}), \tag{93}$$

$$\lim_{\mathbf{r}'_{\rm D}\to 0} s_{nm} = ik_0^3\epsilon_0^{-1}(-1)^m \mathbf{p}_{\rm D}^{(0)}(\omega)\cdot \lim_{\mathbf{r}'_{\rm D}\to 0} \underline{\mathbf{M}}^{(\rm I)}_{n(-m)}(k_0 r'_{\rm D},\theta'_{\rm D},\phi'_{\rm D}), \tag{94}$$

then do the coordinate transformation $\mathscr{R}(\theta,\phi,\theta',\phi')$,

$$\begin{bmatrix}\hat{r}\\ \hat{\theta}\\ \hat{\phi}\end{bmatrix} = \mathscr{R}(\theta,\phi,\theta',\phi')\begin{bmatrix}\hat{r}'\\ \hat{\theta}'\\ \hat{\phi}'\end{bmatrix} \xrightarrow{\phi=\phi'} \begin{bmatrix}\cos(\theta'-\theta) & -\sin(\theta'-\theta) & 0\\ \sin(\theta'-\theta) & \cos(\theta'-\theta) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{r}'\\ \hat{\theta}'\\ \hat{\phi}'\end{bmatrix} \tag{95}$$

First, we evaluate the values of r_{nm} and s_{nm} under the secondary coordinate. According to Eq. (190) in Appendix G, we obtain that $s_{nm}=0$ ($\mathbf{r}_{\rm D}\to 0,\ \forall n,m$). On the other hand, by using Eq. (194), the only nonzero term of r_{nm} for a linear-polarized electric point dipole aligned to z axis [$\mathbf{p}_{\rm D}^{(0)}(\omega)=p_{\rm D}^{(0)}(\omega)\ \hat{z}'=p_{\rm D}^{(0)}(\omega)\ \hat{r}'$] is

$$\lim_{\mathbf{r}'_{\rm D}\to 0} r_{10} = \frac{ik_0^3}{\epsilon_0}\frac{1}{\sqrt{2f_{10}}}\frac{2}{3}P_1^m(1)\cdot p_{\rm D}^{(0)}(\omega) = \frac{ik_0^3}{\epsilon_0}\left[\frac{1}{6\pi}\right]^{1/2} p_{\rm D}^{(0)}(\omega). \tag{96}$$

Thus, the electric field of a z-polarized point dipole in the new spherical coordinate becomes

$$\mathbf{E}^{'(0)}_{\rm z-dip}(\mathbf{r}',\omega) = r_{10}\underline{\mathbf{N}}^{(\rm III)}_{10}(k_0 r',\theta',\phi') = \frac{p_{\rm D}^{(0)}(\omega)}{4\pi\epsilon_0} ik_0^3 \mathbf{N}^{(\rm III)}_{10}(k_0 r',\theta',\phi') \tag{97}$$

For a x-polarized electric point dipole, $\mathbf{p}_{\rm D}^{(0)}(\omega)=p_{\rm D}^{(0)}(\omega)\ \hat{x}'=p_{\rm D}^{(0)}(\omega)\ \hat{\theta}'$, the nonzero terms of r_{nm} are

$$\lim_{\mathbf{r}'_{\rm D}\to 0} r_{11} = \frac{ik_0^3}{\epsilon_0}\frac{-1}{\sqrt{2f_{1(-1)}}}\frac{2}{3}\tau_{1(-1)}(0)\cdot p_{\rm D}^{(0)}(\omega) = -\frac{ik_0^3}{\epsilon_0}\left[\frac{1}{12\pi}\right]^{1/2} p_{\rm D}^{(0)}(\omega) \tag{98a}$$

$$\lim_{\mathbf{r}'_{\rm D}\to 0} r_{1(-1)} = \frac{ik_0^3}{\epsilon_0}\frac{-1}{\sqrt{2f_{11}}}\frac{2}{3}\tau_{11}(0)\cdot p_{\rm D}^{(0)}(\omega) = \frac{ik_0^3}{\epsilon_0}\left[\frac{1}{12\pi}\right]^{1/2} p_{\rm D}^{(0)}(\omega) \tag{98b}$$

The electric field of a x-polarized point dipole in the new spherical coordinate reads

$$\mathbf{E}^{'(0)}_{\mathrm{x-dip}}(\mathbf{r}',\omega)=\frac{p^{(0)}_{\mathrm{D}}(\omega)}{4\pi\epsilon_0}ik_0^3\cdot\frac{1}{2}\left[\mathbf{N}^{(\mathrm{III})}_{1(-1)}(k_0r',\theta',\phi')-\mathbf{N}^{(\mathrm{III})}_{11}(k_0r',\theta',\phi')\right] \tag{99}$$

In the same way, for a y-polarized electric point dipole $[\mathbf{p}^{(0)}_{\mathrm{D}}(\omega)=p^{(0)}_{\mathrm{D}}(\omega)\ \hat{y}'=p^{(0)}_{\mathrm{D}}(\omega)\ \hat{\phi}']$, the nonzero terms of r_{nm} are

$$\lim_{\mathbf{r}'_{\mathrm{D}}\to 0} r_{11}=\frac{ik_0^3}{\epsilon_0}\frac{-1}{\sqrt{2f_{1(-1)}}}\frac{2}{3}i\pi_{1(-1)}(0)\cdot p^{(0)}_{\mathrm{D}}(\omega)=\frac{ik_0^3}{\epsilon_0}\cdot i\left[\frac{1}{12\pi}\right]^{1/2}p^{(0)}_{\mathrm{D}}(\omega), \tag{100a}$$

$$\lim_{\mathbf{r}'_{\mathrm{D}}\to 0} r_{1(-1)}=\frac{ik_0^3}{\epsilon_0}\frac{-1}{\sqrt{2f_{11}}}\frac{2}{3}i\pi_{11}(0)\cdot p^{(0)}_{\mathrm{D}}(\omega)=\frac{ik_0^3}{\epsilon_0}\cdot i\left[\frac{1}{12\pi}\right]^{1/2}p^{(0)}_{\mathrm{D}}(\omega), \tag{100b}$$

and the electric field is

$$\mathbf{E}^{'(0)}_{\mathrm{y-dip}}(\mathbf{r}',\omega)=\frac{p^{(0)}_{\mathrm{D}}(\omega)}{4\pi\epsilon_0}ik_0^3\cdot\frac{i}{2}\left[\mathbf{N}^{(\mathrm{III})}_{1(-1)}(k_0r',\theta',\phi')+\mathbf{N}^{(\mathrm{III})}_{11}(k_0r',\theta',\phi')\right] \tag{101}$$

Equations (97), (99), and (101) are the final forms used in the numerical calculations. Generally, for an arbitrary linear-polarized electric dipole,

$$\mathbf{p}^{(0)}_{\mathrm{D}}(\omega)=p^{(0)}_{\mathrm{D}}(\omega)\ \hat{p}=p^{(0)}_{\mathrm{D,x'}}(\omega)\ \hat{x}'+p^{(0)}_{\mathrm{D,y'}}(\omega)\ \hat{y}'+p^{(0)}_{\mathrm{D,z'}}(\omega)\ \hat{z}' \tag{102}$$

$$=p^{(0)}_{\mathrm{D,x'}}(\omega)\ \hat{\theta}'+p^{(0)}_{\mathrm{D,y'}}(\omega)\ \hat{\phi}'+p^{(0)}_{\mathrm{D,z'}}(\omega)\ \hat{r}', \tag{103}$$

the electric field is expressed as

$$\begin{aligned}\mathbf{E}^{'(0)}(\mathbf{r}',\omega)=&\frac{p^{(0)}_{\mathrm{D,z'}}(\omega)}{4\pi\epsilon_0}ik_0^3\cdot\mathbf{N}^{(\mathrm{III})}_{10}(k_0r',\theta',\phi')\\&+\frac{p^{(0)}_{\mathrm{D,x'}}(\omega)}{4\pi\epsilon_0}ik_0^3\cdot\frac{1}{2}\left[\mathbf{N}^{(\mathrm{III})}_{1(-1)}(k_0r',\theta',\phi')-\mathbf{N}^{(\mathrm{III})}_{11}(k_0r',\theta',\phi')\right]\\&+\frac{p^{(0)}_{\mathrm{D,y'}}(\omega)}{4\pi\epsilon_0}ik_0^3\cdot\frac{i}{2}\left[\mathbf{N}^{(\mathrm{III})}_{1(-1)}(k_0r',\theta',\phi')+\mathbf{N}^{(\mathrm{III})}_{11}(k_0r',\theta',\phi')\right].\end{aligned} \tag{104}$$

Finally, to express the original electric field, a coordinate transformation back to the primary coordinate is needed

$$\mathbf{E}'^{(0)}(\mathbf{r}',\omega) \longmapsto \mathbf{E}^{(0)}(\mathbf{r},\omega) = \mathscr{R}(\theta,\phi,\theta',\phi') \cdot \mathbf{E}'^{(0)}(\mathbf{r}-\mathbf{r}_{\mathrm{D}},\omega). \tag{105}$$

To check whether Eq. (104) is correct, we continue the simplification. The first vector spherical function in Eq. (104) can be explicitly expressed as

$$\begin{aligned}
\mathbf{N}_{10}^{(\mathrm{III})}(k_0r',\theta',\phi') &= \frac{h_1^{(1)}(k_0r')}{k_0r'} \cdot 2P_1^0(\cos\theta')\ \hat{r}' + \frac{1}{k_0r'}\frac{\mathrm{d}\xi_1(k_0r')}{\mathrm{d}(k_0r')} \cdot \tau_{10}(\theta')\ \hat{\theta}' \\
&= \frac{h_1^{(1)}(k_0r')}{k_0r'} \cdot 2\cos\theta'\ \hat{r}' - \frac{1}{k_0r'}\frac{\mathrm{d}\xi_1(k_0r')}{\mathrm{d}(k_0r')} \cdot \sin\theta'\ \hat{\theta}' \\
&= \frac{h_1^{(1)}(k_0r')}{k_0r'} \cdot (3\cos\theta'\ \hat{r}' - \cos\theta'\ \hat{r}') - \frac{1}{k_0r'}\frac{\mathrm{d}\xi_1(k_0r')}{\mathrm{d}(k_0r')} \cdot \sin\theta'\ \hat{\theta}',
\end{aligned} \tag{106}$$

where $P_1^0(\cos\theta') = \cos\theta'$ and $\tau_{10}(\theta') = -\sin\theta'$. Here, we use the following identities to express the radial functions,

$$\frac{h_n^{(1)}(k_0r')}{k_0r'} = -\frac{e^{ik_0r'}}{(k_0r')^2}\left(1+\frac{i}{k_0r'}\right) = \frac{e^{ik_0r'}}{ik_0^3r'}\left(\frac{1}{r'^2}-\frac{ik_0}{r'}\right), \tag{107a}$$

$$\frac{1}{k_0r'}\frac{\mathrm{d}\xi_1(k_0r')}{\mathrm{d}(k_0r')} = -\frac{ie^{ik_0r'}}{k_0r'}\left(1+\frac{i}{k_0r'}-\frac{1}{(k_0r')^2}\right) = \frac{e^{ik_0r'}}{ik_0^3r'}\left(k_0^2+\frac{ik_0}{r'}-\frac{1}{r'^2}\right), \tag{107b}$$

and apply to Eq. (106), the vector spherical function becomes

$$\begin{aligned}
\mathbf{N}_{10}^{(\mathrm{III})}(k_0r',\theta',\phi') &= \frac{h_1^{(1)}(k_0r')}{k_0r'} \cdot (3\cos\theta'\ \hat{r}' - \cos\theta'\ \hat{r}') - \frac{1}{k_0r'}\frac{\mathrm{d}\xi_1(k_0r')}{\mathrm{d}(k_0r')} \cdot \sin\theta'\ \hat{\theta}' \\
&= \frac{e^{ik_0r'}}{ik_0^3r'}\left\{\left(\frac{1}{r'^2}-\frac{ik_0}{r'}\right)\left[3\cos\theta'\ \hat{r}' - \left(\cos\theta'\ \hat{r}' - \sin\theta'\ \hat{\theta}'\right)\right] - k_0^2 \cdot \sin\theta'\ \hat{\theta}'\right\} \\
&= \frac{e^{ik_0r'}}{ik_0^3r'}\left\{\left(\frac{1}{r'^2}-\frac{ik_0}{r'}\right)[3\hat{r}'(\hat{r}'\cdot\hat{z}') - \hat{z}'] + k_0^2\ \hat{\theta}'(\hat{\theta}'\cdot\hat{z}')\right\} \\
&= \frac{e^{ik_0r'}}{ik_0^3r'}\left\{\left(\frac{1}{r'^2}-\frac{ik_0}{r'}\right)[3\hat{r}'(\hat{r}'\cdot\hat{z}') - \hat{z}'] + k_0^2\,[\hat{z}' - \hat{r}'(\hat{r}'\cdot\hat{z}')]\right\}
\end{aligned} \tag{108}$$

In the derivation of Eq. (108), we use the relation, $\hat{z}' = \hat{r}'(\hat{r}' \cdot \hat{z}') + \hat{\theta}'(\hat{\theta}' \cdot \hat{z}') + \hat{\phi}'(\hat{\phi}' \cdot \hat{z}') = \cos\theta'\ \hat{r}' - \sin\theta'\ \hat{\theta}'$. Combining with Eq. (97), we get

$$\begin{aligned}\mathbf{E}^{'(0)}_{\mathrm{z-dip}}(\mathbf{r}', \omega) &= \frac{p_{\mathrm{D}}^{(0)}(\omega)}{4\pi\epsilon_0} ik_0^3 \mathbf{N}_{10}^{(\mathrm{III})}(k_0 r', \theta', \phi') \\ &= \frac{p_{\mathrm{D}}^{(0)}(\omega)}{4\pi\epsilon_0} \frac{e^{ik_0 r'}}{r'} \left\{ \left(\frac{1}{r'^2} - \frac{ik_0}{r'} \right) [3\hat{r}'(\hat{r}' \cdot \hat{z}') - \hat{z}'] + k_0^2 [\hat{z}' - \hat{r}'(\hat{r}' \cdot \hat{z}')] \right\}, \end{aligned} \tag{109}$$

which is the electric field of a z-polarized dipole. In the same way, the vector spherical functions of the x-polarized dipole and the z-polarized dipole can be expressed as

$$\begin{aligned}&\frac{1}{2}\left[\mathbf{N}_{1(-1)}^{(\mathrm{III})}(k_0 r', \theta', \phi') - \mathbf{N}_{11}^{(\mathrm{III})}(k_0 r', \theta', \phi')\right] \\ &= \left[\frac{h_1^{(1)}(k_0 r')}{k_0 r'} \cdot 2\sin\theta'\cos\phi'\ \hat{r}' + \frac{1}{k_0 r'} \frac{\mathrm{d}\xi_1(k_0 r')}{\mathrm{d}(k_0 r')} \cdot \left(\cos\theta'\cos\phi'\ \hat{\theta}' - \sin\phi'\ \hat{\phi}' \right) \right] \qquad (110) \\ &= \left\{ \frac{h_1^{(1)}(k_0 r')}{k_0 r'} \cdot 2\hat{r}'(\hat{r}' \cdot \hat{x}') + \frac{1}{k_0 r'} \frac{\mathrm{d}\xi_1(k_0 r')}{\mathrm{d}(k_0 r')} \cdot \left[\hat{\theta}'(\hat{\theta}' \cdot \hat{x}') + \hat{\phi}'(\hat{\phi}' \cdot \hat{x}') \right] \right\} \\ &= \frac{e^{ik_0 r'}}{ik_0^3 r'} \left\{ \left(\frac{1}{r'^2} - \frac{ik_0}{r'} \right) \left[3\hat{r}'(\hat{r}' \cdot \hat{x}') - \hat{x}' \right] + k_0^2 [\hat{x}' - \hat{r}'(\hat{r}' \cdot \hat{x}')] \right\}, \qquad (111) \end{aligned}$$

and

$$\begin{aligned}&\frac{i}{2}\left[\mathbf{N}_{1(-1)}^{(\mathrm{III})}(k_0 r', \theta', \phi') + \mathbf{N}_{11}^{(\mathrm{III})}(k_0 r', \theta', \phi')\right] \\ &= \left[\frac{h_1^{(1)}(k_0 r')}{k_0 r'} \cdot 2\sin\theta'\sin\phi'\ \hat{r} + \frac{1}{k_0 r'} \frac{\mathrm{d}\xi_1(k_0 r')}{\mathrm{d}(k_0 r')} \cdot \left(\cos\theta'\sin\phi'\ \hat{\theta}' + \cos\phi'\ \hat{\phi}' \right) \right] \qquad (112) \\ &= \left\{ \frac{h_1^{(1)}(k_0 r')}{k_0 r'} \cdot 2\hat{r}'(\hat{r}' \cdot \hat{y}') + \frac{1}{k_0 r'} \frac{\mathrm{d}\xi_1(k_0 r')}{\mathrm{d}(k_0 r')} \cdot \left[\hat{\theta}'(\hat{\theta}' \cdot \hat{y}') + \hat{\phi}'(\hat{\phi}' \cdot \hat{y}') \right] \right\} \\ &= \frac{e^{ik_0 r'}}{ik_0^3 r'} \left\{ \left(\frac{1}{r'^2} - \frac{ik_0}{r'} \right) [3\hat{r}'(\hat{r}' \cdot \hat{y}') - \hat{y}'] + k_0^2 [\hat{y}' - \hat{r}'(\hat{r}' \cdot \hat{y}')] \right\}, \qquad (113) \end{aligned}$$

respectively. In Eq. (111) and (113), the angular functions are $P_1^{-1}(\cos\theta) = -P_1^1(\cos\theta) = \sin\theta$, $\tau_{1(-1)}(\theta) = -\tau_{11}(\theta) = \cos\theta$, and $\pi_{1(-1)}(\theta) = \pi_{11}(\theta) = -1$. In addition, the Cartesian unit vectors in the representation of spherical coordinate are $\hat{x}' = \sin\theta'\cos\phi'\ \hat{r}' +$

$\cos\theta'\cos\phi'\ \hat{\theta}' - \sin\phi'\ \hat{\phi}'$ and $\hat{y}' = \sin\theta'\sin\phi'\ \hat{r}' + \cos\theta'\sin\phi'\ \hat{\theta}' + \cos\phi'\ \hat{\phi}'$. The electric field of the x-polarized (y-polarized) dipole is similar to Eq. (109), the only difference is the dipole direction. Finally, applying Eqs. (108), (111), and (113) to Eq. (104), the electric field generated by a dipole under the secondary coordinate reads

$$\mathbf{E}'^{(0)}_{\mathrm{dip}}(\mathbf{r}',\omega) = \frac{p^{(0)}_{\mathrm{D}}(\omega)}{4\pi\epsilon_0}\frac{e^{ik_0 r'}}{r'}\left\{\left(\frac{1}{r'^2} - \frac{ik_0}{r'}\right)[3\hat{r}'(\hat{r}'\cdot\hat{p}') - \hat{p}'] + k_0^2\,[\hat{p}' - \hat{r}'(\hat{r}'\cdot\hat{p}')]\right\}, \tag{114}$$

which is indeed the textbook form of the electric field generated by an electric point dipole. On the basis of the above description, the implementation is presented as follows,

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                                        %
4  %    nr     -> double       : Relative refractive index                        %
5  %    k      -> double       : Modulus of wavevector in vacuum                  %
6  %    rdip   -> double (3x1) : Position vector of the dipole                    %
7  %    vdip   -> double (3x1) : Direction vector of the dipole                   %
8  % Outputs:                                                                       %
9  %    EdipS -> double (3x1) : Electric Field in the spherical coordinate        %
10 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12 function EdipS = EdipField(nr,k,rdip,vdip)
13     % Preallocation
14     NX = zeros(3,1); NY = zeros(3,1); NZ = zeros(3,1);
15     % Define Variables
16     r = rdip(1); theta = rdip(2); phi = rdip(3);
17     % Radial Function [Eq.(99), where 1i*k^3 is canceled]
18     Rad1 = exp(1i*k*r)/r*(r^(-2)-1i*k/r);
19     Rad2 = exp(1i*k*r)/r*(k^2+1i*k/r-r^(-2));
20     % Z-Component [Eq.(98), where 1i*k^3 is canceled]
21     NZ(1) =  Rad1*cos(theta)*2;
22     NZ(2) = -Rad2*sin(theta);
23     % X-Component [Eq.(102), where 1i*k^3 is canceled]
24     NX(1) =  Rad1*sin(theta)*cos(phi)*2;
25     NX(2) =  Rad2*cos(theta)*cos(phi);
26     NX(3) = -Rad2*sin(phi);
27     % Y-Component [Eq.(104), where 1i*k^3 is canceled]
28     NY(1) =  Rad1*sin(theta)*sin(phi)*2;
29     NY(2) =  Rad2*cos(theta)*sin(phi);
30     NY(3) =  Rad2*cos(phi);
31     % Electric Dipole Field (Gaussian Unit)
32     EdipS = (NX*vdip(1) + NY*vdip(2) + NZ*vdip(3))*nr;
33 end
```

Function 1: Computation of electric dipole field

Note that we build the function EdipField in Gaussian unit, which makes difference from the above derivation (SI unit). In addition, the coordinate transform $\mathscr{R}(\theta, \phi, \theta', \phi')$ is accomplished by S2S, as shown in Func. 2. For the sake of convenience, we request that $\phi = \phi'$, implying the dipole locates on the z axis.

```

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Inputs:                                                              %
 %   S2     -> double (3x1): Secondary spherical coordinate             %
 %   S1t    -> double      : Theta in the S1 coordinate                 %
 %   S1p    -> double      : Phi in the S1 coordinate                   %
 % Outputs:                                                             %
 %   S1     -> double (3x1): Primary spherical coordinate               %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function S1 = S2S(S2,S1t,S1p)
    cost = cos(S1t);
    sint = sin(S1t);
    cosp = cos(S1p);
    % Coordinate Transformation Matrix [Eq.(87)]
    R = [cost     -sint    0;
         sint      cost    0;
         0         0       cosp];
    S1 = R*S2;
end
```

Function 2: Coordinate transformation between two spherical coordinates

Numerical Precision of Vector Spherical Functions

The precision of vector spherical functions determines whether computations are converged, where the vector spherical functions are the product of spherical Bessel functions and vector spherical harmonics,

$$\underline{\mathbf{M}}_{nm}^{(j)}(k_i r,\theta,\phi) = z_n(k_i r)\left[i\underline{\pi}_{nm}(\theta)\hat{\theta} - \underline{\tau}_{nm}(\theta)\hat{\phi}\right]\frac{1}{\sqrt{2\pi}}e^{im\phi}, \tag{115}$$

$$\begin{aligned}\underline{\mathbf{N}}_{nm}^{(j)}(k_i r,\theta,\phi) = {} & \frac{z_n(k_i r)}{k_i r}\cdot n(n+1)\underline{P}_n^m(\cos\theta)\frac{1}{\sqrt{2\pi}}e^{im\phi}\hat{r} \\ & + \frac{1}{k_i r}\frac{\mathrm{d}Z_n(k_i r)}{\mathrm{d}(k_i r)}\cdot\left[\underline{\tau}_{nm}(\theta)\hat{\theta} + i\underline{\pi}_{nm}(\theta)\hat{\phi}\right]\frac{1}{\sqrt{2\pi}}e^{im\phi}.\end{aligned} \tag{116}$$

Note that $j = \mathrm{I}$ for $z_n(k_i r) = j_n(k_i r)$ and $Z_n(k_i r) = \psi_n(k_i r)$, and $j = \mathrm{III}$ for $z_n(k_i r) = h_n^{(1)}(k_i r)$ and $Z_n(k_i r) = \xi_n(k_i r)$. Here, we construct two MATLAB functions to compute the radial and angular functions. In the radial part, we need to cope with spherical Bessel (Hankel) functions $j_n(k_i r)$ $[h_n^{(1)}(k_i r)]$ and the derivatives of Raccati-Bessel (-Hankel) functions $\psi_n'(k_i r)$ $[\xi_n'(k_i r)]$. The angular part includes exponential functions and the following functions,

$$\underline{P}_n^m(\cos\theta) = \left[\frac{(2n+1)(n-|m|!)}{2n(n+1)(n+|m|!)}\right]^{1/2} P_n^m(\cos\theta) \tag{117a}$$

$$\underline{\pi}_{nm}(\theta) = \left[\frac{(2n+1)(n-|m|!)}{2n(n+1)(n+|m|!)}\right]^{1/2} \pi_{nm}(\theta) \tag{117b}$$

$$\underline{\tau}_{nm}(\theta) = \left[\frac{(2n+1)(n-|m|!)}{2n(n+1)(n+|m|!)}\right]^{1/2} \tau_{nm}(\theta). \tag{117c}$$

Radial Functions

We need to compute the following three radial functions: (1) spherical Bessel (Hankel) functions, (2) Riccati-Bessel (Hankel) functions, and (3) derivatives of Riccati-Bessel (Hankel) functions. Apart from the built-in functions in MATLAB, we use the algorithm[4] to calculate these functions, where a MATLAB version source code is presented in Func. 3. In `SphBessel`,

the two additional sub-functions, `sbesselc` and `rcbesselc`, are called on line 14, 44, and 54. `sbesselc` and `rcbesselc` are form the reference[4] that are originally coding in Fortran language. Here, we translate them to the MATLAB version and present in Funcs. 4 and 5. Note that the three auxiliary functions, `MSTA1`, `MSTA2`, and `envj`, are called in `sbesselc` (Func. 4) and `rcbesselc` (Func. 5). These auxiliary functions are presented in Funcs. 6 to 8.

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                               %
4  %   kr         -> double       : Input argument                         %
5  %   nmax       -> double       : Maximum expansion order                %
6  %   array      -> double       : Array mode (0 or 1)                    %
7  %   type       -> string       : Type of functions ('bessel','hankel1') %
8  % Outputs:                                                              %
9  %   Rad        -> struct array: Set of related Bessel (Hankel) functions%
10 %     .j1      -> double (1xn): Spherical Bessel functions              %
11 %     .psi     -> double (1xn): Riccati-Bessel functions                %
12 %     .dpsi    -> double (1xn): Derivatives of psi                      %
13 %     .raddpsi -> double (1xn): dpsi/kr                                 %
14 %     .h1      -> double (1xn): Spherical Hankel functions              %
15 %     .xi      -> double (1xn): Riccati-Hankel functions                %
16 %     .dxi     -> double (1xn): Derivatives of xi                       %
17 %     .raddxi  -> double (1xn): dxi/kr                                  %
18 % Calling functions:                                                    %
19 %   sbesselc                                                            %
20 %   rcbesselc                                                           %
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22
23 function Rad = SphBessel(kr,nmax,array,type)
24     if strcmp(type,'bessel') == 1
25         if kr == 0
26             if nmax == 0
27                 z1 = 1;
28                 dZ = 1;
29             else
30                 z1 = 0;
31                 Z = 0;
32                 dZ = 0;
33                 raddZ = 0;
34             end
35         else
36             [csj,~] = sbesselc(kr,nmax);
37             if array == 1
38                 z1 = csj(2:nmax+1);
39                 z_2 = csj(1:nmax);
40                 lindex = 1:nmax;
41             else
```

```
                z1 = csj(nmax+1);
                z_2 = csj(nmax);
                lindex = nmax;
            end
            % Riccati-Bessel Functions
            Z = kr*z1;
            % Derivative of Riccati-Bessel Function
            dZ = kr.*z_2-lindex.*z1;
            raddZ = dZ/kr;
        end
        Rad.j1 = z1; Rad.psi = Z; Rad.dpsi = dZ; Rad.raddpsi = raddZ;
    elseif strcmp(type,'hankel1') == 1
        if kr == 0
            if nmax == 0
                z1 = 1-1i*e300;
                Z = -1i;
                dZ = 1;
            else
                z1 = 0-1i*1e300;
                Z = 0-1i*1e300;
                dZ = 1i*1e300;
                raddZ = 0;
            end
        else
            % Spherical Bessel Functions
            [csj,csy] = sbesselc(kr,nmax);
            if or(numel(csj) < (nmax + 1),numel(csy) < (nmax + 1))
                error('Please decrease the expansion order "n".');
            end
            if array == 1
                z1 = csj(2:nmax+1)+1i*csy(2:nmax+1);
            else
                z1 = csj(nmax+1)+1i*csy(nmax+1);
            end
            % Riccati-Bessel Functions and their Derivatives
            [rcj,rcy,drcj,drcy] = rcbesselc(kr,nmax);
            if array == 0
                Z = rcj(nmax+1) + 1i*rcy(nmax+1);
                dZ = drcj(nmax+1) + 1i*drcy(nmax+1);
                raddZ=dZ/kr;
            else
                rcj(1)=[];rcy(1)=[];
                drcj(1)=[];drcy(1)=[];
                Z = rcj + 1i*rcy;
                dZ = drcj + 1i*drcy;
                raddZ=dZ/kr;
            end
        end
        Rad.h1 = z1; Rad.xi = Z; Rad.dxi = dZ; Rad.raddxi = raddZ;
    end
end
```

Function 3: Computation of spherical Bessel functions and their associated functions

```
function [csj,csy] = sbesselc(z,n)
    a0 = abs(z);
    nm = n;
    if a0 < 1e-60
        csj = ones([1,n+1])*0;
        csy = ones([1,n+1])*(-1e300);
        csy(1) = 1e0;
        return
    end
    csj = zeros([1,n+1]);
    csj(1) = sin(z)/z;
    csj(2) = (csj(1)-cos(z))/z;
    if n >= 2
        csa = csj(1); csb = csj(2);
        m = MSTA1(a0,200);
        if m < n
            nm = m;
        else
            m = MSTA2(a0,n,15);
        end
        cf0 = 0.0; cf1 = 1.0-100;
        for k = (m):-1:0
            j = k+1;
            cf = (2.0*k+3.0)*cf1/z-cf0;
            if k <= nm
                csj(j)=cf;
            end
            cf0=cf1; cf1=cf;
        end
        if abs(csa) > abs(csb)
            cs = csa/cf;
        else
            cs = csb/cf0;
        end
        for k = 0:min(nm,n)
            j = k+1;
            csj(j) = cs*csj(j);
        end
    end
    csy=1e200;
    csy(1) = -cos(z)/z;
    csy(2) = (csy(1)-sin(z))/z;
    for k = 2:min(nm,n)
        j = k+1;
        if abs(csj(j-1)) >= abs(csj(j-2))
            csy(j) = (csj(j)*csy(j-1)-1.0/z^2)/csj(j-1);
        else
            csy(j) = (csj(j)*csy(j-2)-(2.0*k-1.0)/z^3)/csj(j-2);
        end
    end
end
```

Function 4: Computation of spherical Bessel (Hankel) functions

```
function [rcj,rcy,drcj,drcy] = rcbesselc(z,n)
    rcj  = zeros(1,n+1); rcy  = zeros(1,n+1);
    drcj = zeros(1,n+1); drcy = zeros(1,n+1); NM=n;
    if abs(z) < 1e-60
        rcy  = -1.0e300*ones(1,n+1); rcy(1)  = -1.0;
        drcy =  1.0e300*ones(1,n+1); drcy(1) =  0.0; drcj(1) = 1;
    else
        rcj(1) =  sin(z); rcj(2) = rcj(1)/z - cos(z);
        rcy(1) = -cos(z); rcy(2) = rcy(1)/z - sin(z);
        rcj0 = rcj(1); rcj1 = rcj(2); RF0 = rcy(1); RF1 = rcy(2);
        for Ky = 3:n+1
            RF2 = (2.0*(Ky-1)-1.0)*RF1/z-RF0;
            if abs(RF2) > 1.0e300
                continue
            end
            rcy(Ky) = RF2; RF0 = RF1; RF1 = RF2;
        end
        drcy(1) = sin(z); drcy(2) = -rcy(2)/z + rcy(1); NMy = Ky-1;
        for Ky = 3:NMy+1
            drcy(Ky) = -(Ky-1)*rcy(Ky)/z + rcy(Ky-1);
        end
        if n >= 2
            M = MSTA1(z,200);
            if M < n
                NM = M;
            else
                M = MSTA2(z,n,15);
            end
            F0 = 0.0; F1 = 1.0e-100;
            for K = M+1:-1:1
                F = (2.0*(K-1)+3.0)*F1/z - F0;
                if K <= NM+1
                    rcj (K) = F;
                end
                F0 = F1; F1 = F;
            end
            if abs(rcj0) > abs(rcj1)
                CS = rcj0/F;
            else
                CS = rcj1/F0;
            end
            for K = 1:NM+1
                rcj(K) = CS*rcj(K);
            end
        end
        drcj(1) = cos(z); drcj(2) = -rcj(2)/z + rcj0;
        for K = 3:NM+1
            drcj(K) = -(K-1)*rcj(K)/z + rcj(K-1);
        end
    end
end
```

Function 5: Computation of Riccati-Bessel (-Hankel) functions

```
function result = MSTA1(z,mp)
    a0 = abs(z);
    n0 = fix(1.1*a0)+1;  n1 = n0+5;
    f0 = envj(n0,a0)-mp; f1 = envj(n1,a0)-mp;
    for i = 1:20
        nn = n1-(n1-n0)/(1.0-f0/f1);
        nn = fix(nn); % type conversion: nn should be int
        f = envj(nn,a0)-mp;
        if abs(nn-n1) < 1
            break
        end
        n0 = n1; n1 = nn;
        f0 = f1; f1 = f;
    end
    result = nn;
end
```

Function 6: Auxiliary function `MSTA1`

```
function result = MSTA2(z,n,mp)
    a0 = abs(z);
    hmp = 0.5*mp;
    ejn = envj(n,a0);
    if ejn <= hmp
        obj = mp;
        n0 = fix(1.1*a0);
    else
        obj = hmp+ejn;
        n0 = n;
    end
    f0 = envj(n0,a0)-obj; n1 = n0+5;
    f1 = envj(n1,a0)-obj;
    for i = 1:20
        nn = fix(n1-(n1-n0)/(1.0-f0/f1)); %nn should be int
        f = envj(nn,a0)-obj;
        if abs(nn-n1) < 1
           break
        end
        n0 = n1; n1 = nn;
        f0 = f1; f1 = f;
    end
    result = nn+10;
end
```

Function 7: Auxiliary function `MSTA2`

```
function result=envj(n,z)
    n = max(1,abs(n));
    result = 0.5*log10(6.28*n)-n*log10(1.36*z/n);
end
```

Function 8: Auxiliary function `envj`

Angular Functions

As shown in Eq. (117), the angular functions are related to associated Legendre polynomials,

$$\underline{P}_n^m(\cos\theta) = \left[\frac{(2n+1)(n-|m|)!}{2n(n+1)(n+|m|)!}\right]^{1/2} P_n^m(\cos\theta), \tag{118a}$$

$$\underline{\pi}_{nm}(\theta) = \left[\frac{(2n+1)(n-|m|)!}{2n(n+1)(n+|m|)!}\right]^{1/2} \frac{m}{\sin\theta} P_n^m(\cos\theta), \tag{118b}$$

$$\underline{\tau}_{nm}(\theta) = \left[\frac{(2n+1)(n-|m|)!}{2n(n+1)(n+|m|)!}\right]^{1/2} \frac{\mathrm{d}}{\mathrm{d}\theta}\left[P_n^m(\cos\theta)\right]. \tag{118c}$$

Naively, we can use the recurrence relations of associated Legendre polynomials to compute these functions, but the function error would cumulatively grow if n is large. To keep the numerical precision, we utilize the relations based on Wigner D functions $[D^n_{m'm}(\theta)]$,

$$\underline{P}_n^m(\cos\theta) = \left[\frac{2n+1}{2n(n+1)}\right]^{1/2} D^n_{m0}(0,\theta,0) \tag{119a}$$

$$\underline{\pi}_{nm}(\theta) = -\left[\frac{2n+1}{8}\right]^{1/2} \left[D^n_{m,1}(0,\theta,0) + D^n_{m,-1}(0,\theta,0)\right], \tag{119b}$$

$$\underline{\tau}_{nm}(\theta) = -\left[\frac{2n+1}{8}\right]^{1/2} \left[D^n_{m,1}(0,\theta,0) - D^n_{m,-1}(0,\theta,0)\right]. \tag{119c}$$

Based on Eq. (119a), we will prove that Eqs. (119b) and (119c) are correct after the introduction of Wigner D functions, which is rarely discussed in present literature. Function (9) demonstrates the computation of Eqs. (119a) to (119c). `NormTauPiP` requests three inputs, `nmax`, `theta`, and `order`. Literally, `nmax` indicates the highest expansion order and `theta` is the variable of Wigner D functions. `order` is a string which makes the output order become 'normal' order or 'reversed' order. Also, it can be found that `NormTauPiP` outputs a structure array, `NAng`, including `NTau`, `NPi`, and `NP`. The three arrays save the output of $\underline{\tau}_{nm}(\theta)$, $\underline{\pi}_{nm}(\theta)$, and $\underline{P}_{nm}(\theta)$, respectively. On line 9 in `NormTauPiP`, the computation of Wigner d functions is done by `Wigner_d`, as shwon in Func. 10. To get the numerically high-precision Wigner D functions, a algorithm based on numerical diagonalization is adopted.[5]

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                                      %
4  %   nmax  -> double      : Maximum expansion order                           %
5  %   theta -> double      : Polar angle (rad)                                 %
6  %   order -> string      : Ordering of tables ('normal' or 'reversed')       %
7  % Outputs:                                                                     %
8  %   NAng  -> struct array: Normalized Tau, Pi, and P functions               %
9  %    .NTau -> double [nx(2n+1)]  : Normalized Tau array                      %
10 %    .NPi  -> double [nx(2n+1)]  : Normalized Pi array                       %
11 %    .NP   -> double [nx(2n+1)]  : Normalized P array                        %
12 % Calling functions:                                                          %
13 %   Wigner_d                                                                 %
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 function NAng = NormTauPiP(nmax,theta,order)
17     % Preallocation
18     NTau = zeros(nmax,2*nmax+1);
19     NPi = zeros(nmax,2*nmax+1);
20     NP = zeros(nmax,2*nmax+1);
21     for indn = 1:nmax
22         % Calling Wigner d Matrix of Order n
23         dn = Wigner_d(indn,theta);
24         % Setting the Order
25         if strcmp(order,'normal') == 1
26             dnp1 = dn(:,indn+2)'; % d_(m,+1)^n
27             dn01 = dn(:,indn+1)'; % d_(m,0)^n
28             dnn1 = dn(:,indn)';   % d_(m,-1)^n
29         elseif strcmp(order,'reversed') == 1
30             dnp1 = fliplr(dn(:,indn+2)'); % d_(m,+1)^n
31             dn01 = fliplr(dn(:,indn+1)'); % d_(m,0)^n
32             dnn1 = fliplr(dn(:,indn)');   % d_(m,-1)^n
33         end
34         % Normalization Constants
35         NormTauPi = sqrt((2*indn+1)/8);
36         NormP = sqrt((2*indn+1)/2./indn./(indn+1));
37         % Output Functions
38         NPi(indn,1:2*indn+1) = -NormTauPi.*(dnp1 + dnn1);
39         NTau(indn,1:2*indn+1) = -NormTauPi.*(dnp1 - dnn1);
40         NP(indn,1:2*indn+1) = NormP.*dn01;
41         % Correction to the Floating Numbers
42         NPi(abs(NPi)<1e-15) = 0;
43         NTau(abs(NTau)<1e-15) = 0;
44         NP(abs(NP)<1e-15) = 0;
45         % Output a Structure Array
46         NAng.NPi = NPi;
47         NAng.NTau = NTau;
48         NAng.NP = NP;
49     end
50 end
```

Function 9: Computation of $\underline{\tau}_{nm}(\theta)$, $\underline{\pi}_{nm}(\theta)$, and $\underline{P}_n^m(\cos\theta)$

Wigner D functions are well-known in angular momentum theory. These functions describe an Euler rotation, and the order of fixed axes is z-y-z. Wigner D functions are defined as[4]

$$D^{j}_{m',m}(\alpha,\beta,\gamma) \equiv \langle j,m'|\, \hat{R}(\alpha,\beta,\gamma)\, |j,m\rangle = \langle j,m'|\, e^{-i\alpha\hat{J}_z} e^{-i\beta\hat{J}_y} e^{-i\gamma\hat{J}_z}\, |j,m\rangle\,, \tag{120}$$

where $\hat{R}(\alpha,\beta,\gamma)$ refers to a rotation operator. $\hat{J}_z$ and $\hat{J}_y$ (also, $\hat{J}_x$) are the generators of the Lie group of SO(3), and the three generators form a Lie algebra $\mathfrak{so}(3)$,

$$\left[\hat{J}_i, \hat{J}_j\right] = i\epsilon_{ijk}\hat{J}_k, \qquad i,j,k = \{\mathrm{x},\mathrm{y},\mathrm{z}\}. \tag{121}$$

ϵ_{ijk} is a skew-symmetric tensor (or in other words, Levi-Civita symbol). This algebra is isomorphic to the orbital angular momentum operators in quantum mechanics. When we do the following operations on the eigenstate $|j,m\rangle$, it yields:

$$\hat{J}_{\mathrm{z}}\, |j,m\rangle = m\, |j,m\rangle\,, \qquad \hat{J}^2\, |j,m\rangle = j(j+1)\, |j,m\rangle\,, \qquad \hat{J}^2 = \hat{J}^2_{\mathrm{x}} + \hat{J}^2_{\mathrm{y}} + \hat{J}^2_{\mathrm{z}}. \tag{122}$$

It is obvious that

$$\begin{aligned} \langle j,m'|\, e^{-i\alpha\hat{J}_z} e^{-i\beta\hat{J}_y} e^{-i\gamma\hat{J}_z}\, |j,m\rangle &= e^{-im'\alpha}\, \langle j,m'|\, e^{-i\beta\hat{J}_y}\, |j,m\rangle\, e^{-im\gamma} \\ &\equiv e^{-im'\alpha} d^{j}_{m',m}(\beta) e^{-im\gamma}, \end{aligned} \tag{123}$$

where we define that $d^{j}_{m',m}(\beta) \equiv \langle j,m'|\, e^{-i\beta\hat{J}_y}\, |j,m\rangle$. Thus, the equation $D^{j}_{m',m}(0,\beta,0) = d^{j}_{m',m}(\beta)$ holds. Here, we would like to emphasize that m is the eigenvalues of $\hat{J}_{\mathrm{z}}$ by convention. In addition, we can also define another complete eigenbasis about $\hat{J}_{\mathrm{y}}$,

$$\mathbf{I} = \sum_{\mu-j}^{j} |j,\mu\rangle\, \langle j,\mu|\,, \tag{124}$$

[4]We use the same symbol m when defining $\underline{\pi}_{nm}(\theta)$, $\underline{\tau}_{nm}(\theta)$, and Wigner D (d) functions since all of them are associated with the degree of freedom of the azimuthal angle. Please do not be confused when we introduce the properties of Wigner D (d) functions with the convention symbols.

and μ denotes the eigenvalue along y axis. Inserting this complete relation, the equation becomes:

$$d^j_{m',m}(\theta) = \sum_\mu \langle j, m'|j, \mu\rangle \, e^{-i\theta \hat{J}_\mathrm{y}} \, \langle j, \mu|j, m\rangle = \sum_\mu e^{-i\theta\mu} \, \langle j, m'|j, \mu\rangle \, \langle j, \mu|j, m\rangle. \tag{125}$$

In the matrix representation, we have

$$\mathbf{d}(\theta) = \begin{bmatrix} d^j_{-m,-m} & d^j_{-m,-m+1} & \cdots & d^j_{-m,m} \\ d^j_{-m+1,-m} & d^j_{-m+1,-m+1} & \cdots & d^j_{-m,m} \\ \vdots & \vdots & \ddots & \vdots \\ d^j_{m,-m} & d^j_{m,-m+1} & \cdots & d^j_{m,m} \end{bmatrix}_\mathrm{z} = \mathbf{P} \begin{bmatrix} e^{i\mu\theta} & 0 & \cdots & 0 \\ 0 & e^{i(\mu-1)\theta} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & e^{-i\mu\theta} \end{bmatrix}_\mathrm{y} \mathbf{P}^{-1} \tag{126}$$

According to Eq. (126), if we get the modal matrix $\mathbf{P}$, we are able to calculate Wigner d functions. To solve the modal matrix $\mathbf{P}$, we define a pair of ladder operators, $\hat{J}_+ \equiv \hat{J}_\mathrm{x} + i\hat{J}_\mathrm{y}$ and $\hat{J}_- \equiv \hat{J}_\mathrm{x} - i\hat{J}_\mathrm{y}$. When these operators act on the eigenstate $|j, m\rangle$, they return

$$\hat{J}_\pm \, |j, m\rangle = C_\pm(j, m) \, |j, m \pm 1\rangle \, , \tag{127}$$

where $C_\pm(j, m) = \sqrt{(j \mp m)(j \pm m + 1)}$. The only step we need to do is solve the eigenvalue problem for $\hat{J}_\mathrm{y}$ in the z-basis representation. The form can obtain from the following formula:

$$\begin{aligned} [\hat{J}_\mathrm{y}]_\mathrm{z} &= \frac{1}{2i}(\hat{J}_+ - \hat{J}_-)_\mathrm{z} \\ &= \frac{1}{2i} \begin{bmatrix} 0 & -C_-(j, -j+1) & 0 & \cdots & 0 \\ C_+(j, -j) & 0 & -C_-(j, -j+2) & \cdots & 0 \\ 0 & C_+(j, -j+1) & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -C_-(j, j) \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_\mathrm{z} \end{aligned} \tag{128}$$

After the diagonalization, we get

$$[\hat{J}_{\mathrm{y}}]_{\mathrm{y}} = \mathbf{P}^{-1}[\hat{J}_{\mathrm{y}}]_{\mathrm{z}}\mathbf{P} = \begin{bmatrix} -\mu & 0 & 0 & \cdots & 0 \\ 0 & -\mu+1 & 0 & \cdots & 0 \\ 0 & 0 & -\mu+2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & \mu \end{bmatrix}_{y} . \tag{129}$$

Applying $\mathbf{P}$ to Eq. (126), we can acquire the Wigner d function. In Func. 10, we show the source code of the algorithm.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs:                                                                      %
%   j     -> double      : j-dimension SO(3) irreducible representation%
%   theta -> double      : Polar angle (rad)                                   %
% Outputs:                                                                     %
%   d     -> double (jxj): Wigner d matrix                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d = Wigner_d(j, theta)
    % Calculation of J+
    m = -j:j-1;
    J = diag(sqrt((j-m).*(j+m+1)), -1);
    % Create the Spectral Decomposition Matrix J_y at z Representation
    Jy = (J-J')/2i;
    % Diagonalization
    [V, D] = eig(Jy);
    % Unitary Transformation
    d = V*diag(exp(-1i*theta*diag(D)))*V';
    % Check the Quality of the Transformation
    if max(max(abs(imag(d)))) > 1e-12
        warn_mes = 'Wigner_d may not give reliable results.';
        dispstat(sprintf(warn_mes),'keepthis','timestamp');
    end
    % Change Data Type (double complex -> double real)
    d = real(d);
end
```

Function 10: Computation of Wigner d functions

Vector Spherical Functions

By calling the functions, `SphBessel` and `NormTauPiP`, we are able to construct the basis functions now, which is accomplished by Func. 11. In `VectSphField`, components in $\mathbf{F} = \{\underline{\mathbf{M}}_{nm}^{(j)}, \underline{\mathbf{N}}_{nm}^{(j)}\}$ follow the two-dimensional arrangement if we set the output of `NormTauPiP` to be 'normal' order,

$$[\mathbf{F}]_{nm}^{i} = \begin{bmatrix} F_{1,-1}^{i} & F_{1,0}^{i} & F_{1,1}^{i} & 0 & 0 & \dots & 0 \\ F_{2,-2}^{i} & F_{2,-1}^{i} & F_{2,0}^{i} & F_{2,1}^{i} & F_{1,2}^{i} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ F_{n-1,-n+1}^{i} & F_{n-1,-n+2}^{i} & F_{n-1,-n+3}^{i} & F_{n-1,-n+4}^{i} & F_{n-1,-n+5}^{i} & \dots & 0 \\ F_{n,-n}^{i} & F_{n,-n+1}^{i} & F_{n,-n+2}^{i} & F_{n,-n+3}^{i} & F_{n,-n+4}^{i} & \dots & F_{n,n}^{i} \end{bmatrix}. \tag{130}$$

Index i indicates the i-th component of spherical coordinates. The elements of $\mathbf{F}$ in each row store the same order of n; for each n, a total of $2n+1$ choices for m. Thus, if $n_{\max}$ is the highest expansion order to n, a total of $2(n_{\max} - n)$ zeros in the rest of each row. Note that although the arrangement in Eq. (130) is not conducive to computation efficiency, it is a friendly arrangement for humans to debug. On the other hand, if we set the output of `NormTauPiP` to be 'reversed' order, the reversed array $\tilde{\mathbf{F}}$ becomes

$$[\tilde{\mathbf{F}}]_{nm}^{i} = \begin{bmatrix} F_{1,1}^{i} & F_{1,0}^{i} & F_{1,-1}^{i} & 0 & 0 & \dots & 0 \\ F_{2,2}^{i} & F_{2,1}^{i} & F_{2,0}^{i} & F_{2,-1}^{i} & F_{1,-2}^{i} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ F_{n-1,n-1}^{i} & F_{n-1,n-2}^{i} & F_{n-1,n-3}^{i} & F_{n-1,n-4}^{i} & F_{n-1,n-5}^{i} & \dots & 0 \\ F_{n,n}^{i} & F_{n,n-1}^{i} & F_{n,n-2}^{i} & F_{n,n-3}^{i} & F_{n,n-4}^{i} & \dots & F_{n,-n}^{i} \end{bmatrix}. \tag{131}$$

The reversed vector spherical fields are called when calculating the dyadic Green's functions [in practice, expansion coefficients of the electric point dipole, Eq. (72)].

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs:                                                                  %
%   kr        -> double            : Dimensionless Radial variable         %
%   nmax      -> double            : Maximum expansion order               %
%   Rad       -> struct array**    : Set of radial functions               %
%     .j1     -> double (1xn)      : Spherical Bessel functions            %
%     .raddpsi -> double (1xn)     : dpsi/kr (See more in SphBessel)       %
%     .h1     -> double (1xn)      : Spherical Bessel functions            %
%     .raddxi -> double (1xn)      : dxi/kr (See more in SphBessel)        %
%   NAng      -> struct array      : Normalized Tau, Pi, and P funcs.      %
%     .NTau   -> double [nx(2n+1)] : Normalized Tau array                  %
%     .NPi    -> double [nx(2n+1)] : Normalized Pi array                   %
%     .NP     -> double [nx(2n+1)] : Normalized P array                    %
%   emphi     -> double [nx(2n+1)] : Array of e^(i*m*phi)                  %
% Outputs:                                                                 %
%   VSF       -> struct array      : Vector spherical functions            %
%     .M      -> double [nx(2n+1)x3]: Vector spherical function M          %
%     .N      -> double [nx(2n+1)x3]: Vector spherical function N          %
%                                                                          %
% **:                                                                      %
%  The functions, j1 (raddpsi) and h1 (raddxi), in Rad should not          %
%  appear simultaneously.                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function VSF = VectSphFunc(kr,nmax,Rad,NAng,emphi)
    % Preallocation
    VSF.M = zeros(nmax,2*nmax+1,3);
    VSF.N = zeros(nmax,2*nmax+1,3);
    % Extract Radial Functions
    if isfield(Rad,'h1') == 1
        z1 = Rad.h1;
    elseif isfield(Rad,'j1') == 1
        z1 = Rad.j1;
    end
    if isfield(Rad,'raddxi') == 1
        raddz = Rad.raddxi;
    elseif isfield(Rad,'raddpsi') == 1
        raddz = Rad.raddpsi;
    end
    % Construct the Array of Each Order
    n = (1:nmax)';
    %M Field
    VSF.M(:,:,2) = 1i.*transpose(z1).*NAng.NPi.*emphi;
    VSF.M(:,:,3) = -transpose(z1).*NAng.NTau.*emphi;
    %N Field
    VSF.N(:,:,1) = transpose(z1)/kr.*n.*(n+1).*NAng.NP.*emphi;
    VSF.N(:,:,2) = transpose(raddz).*NAng.NTau.*emphi;
    VSF.N(:,:,3) = 1i.*transpose(raddz).*NAng.NPi.*emphi;
end
```

Function 11: Generation of Vector Spherical Functions

Computation of Mie Coefficients

The computations of Mie coefficients for a single sphere can be implemeted by Eq. (77) once the MATLAB function `SphBessel` is constructed. The source code `MieSingle` computes the Mie coefficients for a single sphere, as shown in Func. 12.

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                                     %
4  %   nr      -> double (1x2): Relative refractive index                       %
5  %   ks      -> double      : Dimensionless boundary (k*sphere_radius)        %
6  %   n       -> double      : expansion order                                 %
7  % Outputs:                                                                    %
8  %   Coeffs  -> struct array: Mie coefficients                                %
9  %    .alpha -> double (1xn): Mie coefficient alpha                           %
10 %    .beta  -> double (1xn): Mie coefficient beta                            %
11 %    .gamma -> double (1xn): Mie coefficient gamma                           %
12 %    .delta -> double (1xn): Mie coefficient delta                           %
13 % Calling functions:                                                         %
14 %   SphBessel                                                               %
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 function Coeffs = MieSingle(nr,ks,nmax)
18     % Checking Input
19     if or(ne(max(size(nr)),2) ,ne(max(size(ks)),1) ) == 1
20         errmes = 'Error input size of "nr" or "ks" from "MieSingle"';
21         dispstat(sprintf(errmes),'keepthis','timestamp');
22     end
23     % Defining Variables
24     n0 = nr(1); n1 = nr(2);
25     n0kr1 = n0*ks; n1kr1 = n1*ks;
26     % Generating Radial Functions
27     n0Rad = SphBessel(n0kr1,nmax,1,'bessel');
28     n0psi = n0Rad.psi; n0dpsi = n0Rad.dpsi;
29     n0Rad = SphBessel(n0kr1,nmax,1,'hankel1');
30     n0xi  = n0Rad.xi ; n0dxi  = n0Rad.dxi ;
31     n1Rad = SphBessel(n1kr1,nmax,1,'bessel');
32     n1psi = n1Rad.psi; n1dpsi = n1Rad.dpsi;
33     % Coefficients
34     Coeffs.alpha = -(n1*n0dpsi.*n1psi - n0*n0psi.*n1dpsi)./...
35               (n1*n0dxi.*n1psi - n0*n0xi.*n1dpsi);
36     Coeffs.beta  = -(n0*n0dpsi.*n1psi - n1*n0psi.*n1dpsi)./...
37                 (n0*n0dxi.*n1psi - n1*n0xi.*n1dpsi);
38     Coeffs.gamma = n1*(n0dpsi.*n0xi - n0psi.*n0dxi)./...
39              (n1*n1dpsi.*n0xi - n0*n1psi.*n0dxi);
40     Coeffs.delta = n1*(n0dpsi.*n0xi - n0psi.*n0dxi)./...
41              (n0*n1dpsi.*n0xi - n1*n1psi.*n0dxi);
42 end
```

Function 12: Mie coefficients of a single sphere

On the contrary, the computations of Mie coefficients for a core/shell sphere need an additional treatment to improve the numerical stability. As described in Eq. (86), we introduce the logarithmic derivatives of Riccati-Bessel (-Hankel) in the expression of Mie coefficients for a core/shell sphere,

$$\mathscr{D}\psi_n(z) = \frac{\mathrm{d}}{\mathrm{d}z}\ln\psi_n(z) = \frac{\psi_n'(z)}{\psi_n(z)},$$
$$\mathscr{D}\xi_n(z) = \frac{\mathrm{d}}{\mathrm{d}z}\ln\xi_n(z) = \frac{\xi_n'(z)}{\xi_n(z)}.$$

To compute the two functions, we adopt the algorithm from previous work.[3,6,7] The source code is shown in Func. 13. By calling the MATLAB function `Dlog`, the computation of Mie coefficiens for a core/shell sphere becomes straightforward, as shown in Func. 14.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs:                                                                     %
%   z      -> double       : Input argument                                   %
%   n      -> double       : Expansion order                                  %
% Outputs:                                                                    %
%   D1     -> double (1xn): Logarithmic derivatives (Ricatti-Bessel)          %
%   D3     -> double (1xn): Logarithmic derivatives (Ricatti-Hankel)          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [D1,D3] = Dlog(z,n)
    nex = n+15;
    D1 = zeros(1,nex);
    D3 = zeros(1,nex);
    for nn = nex:-1:2
        D1(nn-1) = nn/z - 1/(D1(nn)+nn/z);
    end
    psixi=zeros(nex,1);
    psi0xi0 = (1-exp(2i*z))/2;
    D30 = 1i;
    D10 = cot(z);
    psixi(1) = psi0xi0*(1/z-D10)*(1/z-D30);
    D3(1) = D1(1) + 1i/psixi(1);
    for nn=2:nex
        psixi(nn) = psixi(nn-1)*(nn/z-D1(nn-1))*(nn/z-D3(nn-1));
        D3(nn) = D1(nn) + 1i/psixi(nn);
    end
    D1(n+1:n+15) = [];
    D3(n+1:n+15) = [];
end
```

Function 13: Logarithmic Derivatives of Riccati-Bessel (-Hankel) functions

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                                      %
4  %   nr      -> double (1x3): Relative refractive index                        %
5  %   ks      -> double      : Dimensionless boundary (k*sphere_radius)         %
6  %   nmax    -> double      : Maximum expansion order                          %
7  % Outputs:                                                                     %
8  %   Coeffs  -> struct array: Mie coefficients                                 %
9  %    .alpha -> double (1xn): Mie coefficient alpha                            %
10 %    .beta  -> double (1xn): Mie coefficient beta                             %
11 % Calling functions:                                                          %
12 %   SphBessel                                                                 %
13 %   Dlog                                                                      %
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16 function Coeffs = MieCoreShell(nr,ks,nmax)
17     % Checking Input
18     if or(ne(max(size(nr)),3) ,ne(max(size(ks)),2) ) == 1
19         disp('Error input size of "nr" or "ks" from "MieCoreShell"');
20     end
21     % Defining Variables
22     n0 = nr(1); n1 = nr(2); n2 = nr(3);
23     if n0 == n1
24         n1 = n1+1e-7;
25     end
26     n0kr1=n0*ks(1); n1kr1=n1*ks(1); n1kr2=n1*ks(2); n2kr2=n2*ks(2);
27     % Calling Radial Functions (out)
28     n0kr1Rad = SphBessel(n0kr1,nmax,1,'bessel') ; n0kr1psi = n0kr1Rad.psi;
29     n0kr1Rad = SphBessel(n0kr1,nmax,1,'hankel1'); n0kr1xi  = n0kr1Rad.xi;
30     % Calling Radial Functions (shell)
31     n1kr1Rad = SphBessel(n1kr1,nmax,1,'hankel1'); n1kr1xi  = n1kr1Rad.xi;
32     n1kr2Rad = SphBessel(n1kr2,nmax,1,'hankel1'); n1kr2xi  = n1kr2Rad.xi;
33     n1kr1Rad = SphBessel(n1kr1,nmax,1,'bessel') ; n1kr1psi = n1kr1Rad.psi;
34     n1kr2Rad = SphBessel(n1kr2,nmax,1,'bessel') ; n1kr2psi = n1kr2Rad.psi;
35     % Calling Dlog
36     [n1kr2D1,n1kr2D3]=Dlog(n1kr2,nmax);[n1kr1D1,n1kr1D3]=Dlog(n1kr1,nmax);
37     [n0kr1D1,n0kr1D3]=Dlog(n0kr1,nmax);[n2kr2D1,~]=Dlog(n2kr2,nmax);
38     % Factors
39     f1=n1kr2xi./n1kr2psi; f2=n1kr1xi./n1kr1psi; f3=n0kr1psi./n0kr1xi;
40     A  = (n2*n1kr2D3 - n1*n2kr2D1)./(n1*n2kr2D1 - n2*n1kr2D1).*f1;
41     B  = (n2*n2kr2D1 - n1*n1kr2D3)./(n1*n1kr2D1 - n2*n2kr2D1).*f1;
42     A1 = (n1*n0kr1D1 - n0*n1kr1D3)./(n0*n1kr1D1 - n1*n0kr1D1).*f2;
43     A2 = (n0*n1kr1D3 - n1*n0kr1D3)./(n1*n0kr1D3 - n0*n1kr1D1).*f2;
44     B1 = (n1*n1kr1D3 - n0*n0kr1D1)./(n0*n0kr1D1 - n1*n1kr1D1).*f2;
45     B2 = (n0*n0kr1D3 - n1*n1kr1D3)./(n1*n1kr1D1 - n0*n0kr1D3).*f2;
46     % Coefficients
47     Coeffs.alpha = (A1-A)./(A2-A).*f3.*...
48                    (n0*n1kr1D1 - n1*n0kr1D1)./(n1*n0kr1D3 - n0*n1kr1D1);
49     Coeffs.beta  = (B1-B)./(B2-B).*f3.*...
50                    (n0*n0kr1D1 - n1*n1kr1D1)./(n1*n1kr1D1 - n0*n0kr1D3);
51 end
```

Function 14: Mie coefficients of a core/shell sphere

Expansion Coefficients of Electric Point Dipole

In this section, we deal with the expansion coefficients of a electric point dipole, where the equation is explicitly shown in Eq. (71). It is worthwhile to mention again that the electric field produced by the electric point dipole do not computed by Eq. (72). We compute these coefficients for the scattering electric fields in Eqs. (75) and (83). In addition, we adopt Gaussian unit to express electric fields in the source code (line 25) in Func. 15.

```
1
2   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3   % Inputs:                                                                 %
4   %   Settings -> struct array     : Calculation parameters                 %
5   %     .nmax   -> double           : Maximum expansion order                %
6   %     .nr     -> double (1xp)**   : Relative refractive index              %
7   %     .k0     -> double           : modulus of wavevector in vacuum        %
8   %     .DPos   -> double (3x1)     : Donor position                         %
9   %       .Sph  -> double (3x1)     : Presented in a spherical coordinate    %
10  %   Rad       -> struct array     : Set of radial functions                %
11  %     .h1     -> double (1xn)     : Spherical Bessel functions             %
12  %     .raddxi -> double (1xn)     : dxi/kr (See more in SphBessel)         %
13  %   NAng      -> struct array     : Normalized Tau, Pi, and P funcs.       %
14  %     .NTau   -> double [nx(2n+1)]: Normalized Tau array                   %
15  %     .NPi    -> double [nx(2n+1)]: Normalized Pi array                    %
16  %     .NP     -> double [nx(2n+1)]: Normalized P array                     %
17  % Outputs:                                                                %
18  %   Source    -> struct array     : Source Expansion coefficients          %
19  %     .p      -> double [nx(2n+1)]: Coefficient p                          %
20  %     .q      -> double [nx(2n+1)]: Coefficient q                          %
21  % Calling functions:                                                      %
22  %   SphBessel                                                             %
23  %   VectSphField                                                          %
24  %   TenCont (Tensor Contraction)                                          %
25  %                                                                         %
26  % **:                                                                     %
27  %  The value of p depends on the spherical scatterer:                     %
28  %   Single sphere     -> p = 2                                            %
29  %   Core/shell sphere -> p = 3                                            %
30  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
31
32  function Source = SourCoeff(Settings,type)
33      % Variables
34      nmax = Settings.nmax;
35      n0 = Settings.nr(1);
36      kr = n0*Settings.k0*Settings.DPos.Sph(1);
37      % Preallocation
38      m = -inf*ones(nmax,2*nmax+1);
39      % Generate Azimuthal Function
40      for ii = 1:nmax
41              m(ii,1:2*ii+1) = ii:-1:-ii;
```

```
    end
    if Settings.DPos.Sph(3) == 0
        % For Speed-Up
        emphi = sqrt(1/2/pi);
    else
        emphi = sqrt(1/2/pi)*exp(1i*m*Settings.DPos.Sph(3));
        emphi(isnan(emphi)) = 0;
    end
    % Generate N and M Functions
    VSF = VectSphFunc(kr,nmax,Settings.DRad,Settings.DNAng,emphi);
    % Calculate Prefactor
    if strcmp(type,"Green's function only") == 1
        prefactor = 1i*(n0*Settings.k0)*(-1).^m;
    elseif strcmp(type,'dipole') == 1
        % Prefactor from the Green's Function and a Dipole (Gaussian Unit)
        prefactor = 4*pi*1i*(n0*Settings.k0)^3*(-1).^m;
    end
    % Output
    Source.p = prefactor.*TenCont(VSF.N,Settings.DOri.Sph,[3,1]);
    Source.q = prefactor.*TenCont(VSF.M,Settings.DOri.Sph,[3,1]);
end
```

Function 15: Expansion coefficients of an electric point dipole

Here, an additional function `TenCont` computes the inner product of two tensors, as shown in Func.16.

```
function result = TenCont(A,B,dim)
    % Matrix Size
    sizeA = size(A);
    sizeB = size(B);
    % Error of dimension
    if size(dim) > 2
        disp('Wrong Assignment of Dimension in TenCont');
    end
    % Size of the Target Column
    sizetar = max(sizeB);
    % Alert of Illegal Operation
    if sum(sizeB) > sizetar+1
        dispstat(sprintf('Illegal Tensor Contraction'),'keepthis',...
            'timestamp');
    end
    % Erasing the Contribution of the Target Column
    sizeA(dim(1)) = [];
    % Reshaping Matrix and Contraction
    result = reshape(reshape(A,[prod(sizeA),sizetar])*B,sizeA);
end
```

Function 16: Contraction of two tenors for a specific index.

Modules of Calculating Electric Fields in Different Regions

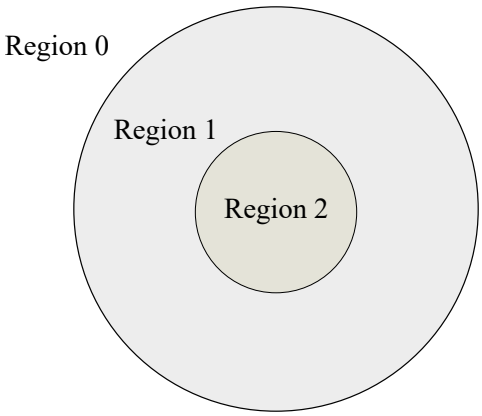

Figure 4: Schematic illustration of space division. Computations of electric fields for a specific position is based on the located region.

On the basis of aforementioned MATLAB functions (subroutines), we are now able to construct modules to compute electric fields. Because dielectric functions are assumed to be piecewise homogeneous, the computation of electric fields can be divided into two (three) concentric-sphere regions, as shown in Figure 4. Hence, we construct calculation modules for each region, i.e., `EFieldR0`, `EFieldR1`, and so on.[5] To briefly explain the details, we present the flowchart of the module `EFieldR0`, as shwon in Fig. 5. At the beginning of `EFieldR0`, we make a series of decisions to check whether the required variables (structure arrays, objects in MATLAB) exist or not. Processes of checking variables allow us to prevent redundant calculations in a for-loop and compute the needed variables in advance. Here, `DRad` and `ARad` denote the radial functions of the donor and the acceptor individually; `DNAng` and `ANAng` denote the normalized angular functions of the donor and the acceptor, repectively. Also, `emphi`, `Source`, and `Scat`, respresent the normalized azimuthal functions of the acceptor, the expansion coefficients of the dipole source, and the scattering coefficients, respectively.[6]

[5]For the case of the core/shell sphere, the corresponding modules have not been completely constructed. Therefore, the readers can extend the code to their needs according to this scheme.

[6]Because the code is constructed to calculate the generalized spectral overlap in the resonance energy transfer theory, we use the language of donor and acceptor to name the variables.

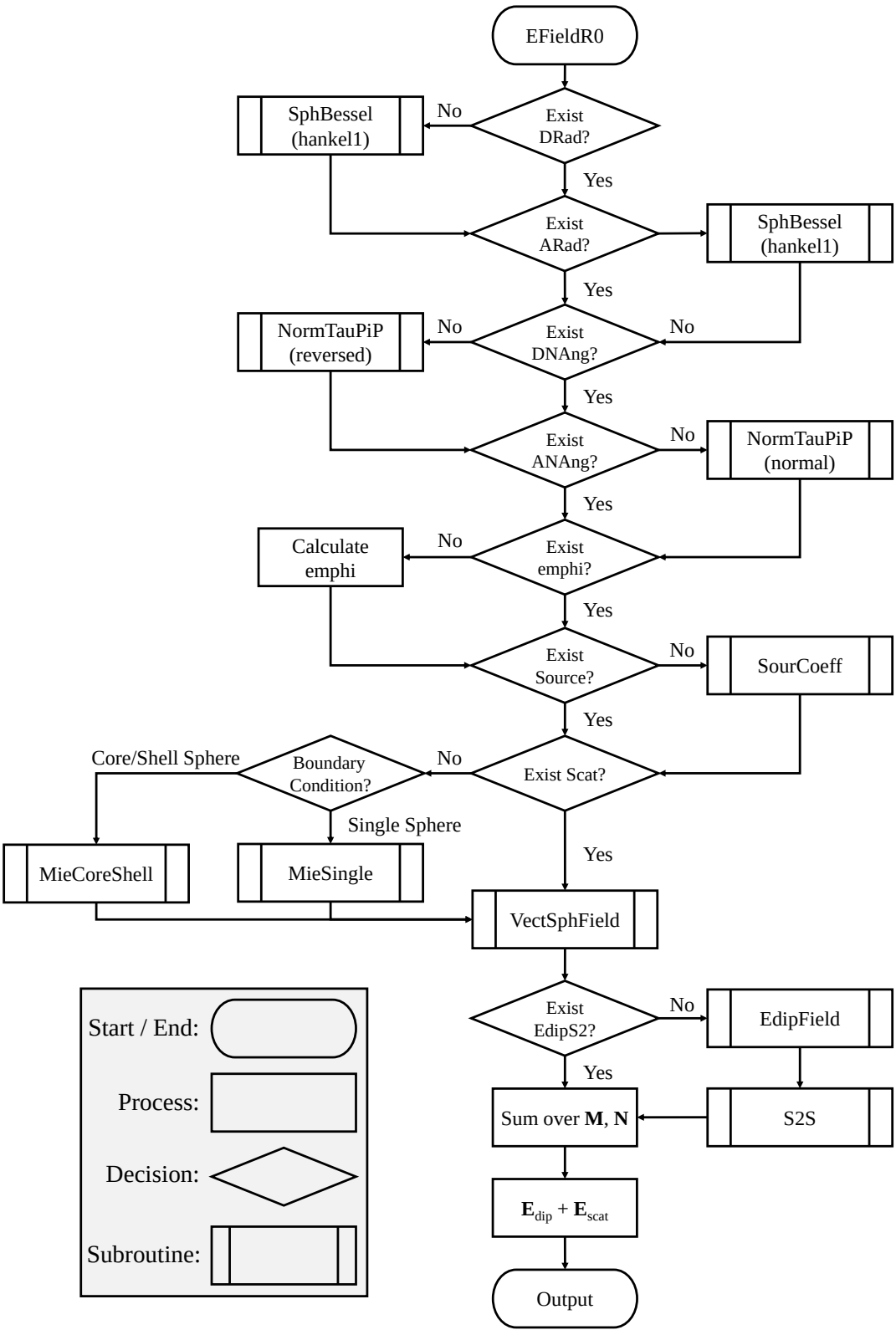

Figure 5: Flowchart of module EFieldR0

If variables have not been calculated, the code will call the corresponding functions to calculate them. In `EFieldR0`, an additional step is needed to compute the electric field generated by the electric dipole (in other words, incident electric field) because the dipole is placed in the zeroth region, as shown in the final decision. Once the variables are prepared, we sum over all the vector spherical functions and the contribution from the electric point dipole ($\mathbf{E}_{\text{dip}}$), and output the results. Another demonstration module, `EFieldR1`, can be found in Appendix J.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs:                                                                    %
% ! Settings -> struct array     : Calculation parameters                   %
% !  .nmax   -> double           : Maximum expansion order                  %
% !  .nr     -> double (1xp)**   : Relative refractive index                %
% !  .k0     -> double           : modulus of wavevector in vacuum          %
% !  .k0s    -> double [1x(p-1)] : k0*r_i (r_i = radius of boundary)        %
% !  .BC     -> string           : Boundary condition                       %
% !  .DPos   -> double (3x1)     : Donor position                           %
% !   .Sph   -> double (3x1)     : Presented in a spherical coordinate      %
% !   .Cart  -> double (3x1)     : Presented in a Cartesian coordinate      %
% !  .APos   -> double (3x1)     : Acceptor position                        %
% !   .Sph   -> double (3x1)     : Presented in a spherical coordinate      %
% !   .Sph2  -> double (3x1)     : Presented in S2 coordinate               %
% !   .Cart  -> double (3x1)     : Presented in a Cartesian coordinate      %
% !  .DOri   -> double (3x1)     : Orientation of the Donor dipole          %
% !   .Cart  -> double (3x1)     : Presented in a Cartesian coordinate      %
%    .DRad   -> struct array     : Set of radial functions (donor)          %
%     .h1    -> double (1xn)     : Spherical Bessel functions               %
%     .raddxi-> double (1xn)     : dxi/kr (See more in SphBessel)           %
%    .ARad   -> struct array     : Set of radial functions (acceptor)       %
%     .h1    -> double (1xn)     : Spherical Bessel functions               %
%     .raddxi-> double (1xn)     : dxi/kr (See more in SphBessel)           %
%    .DNAng  -> struct array     : Normalized Tau, Pi, and P (donor)        %
%     .NTau  -> double [nx(2n+1)]: Normalized Tau array                     %
%     .NPi   -> double [nx(2n+1)]: Normalized Pi array                      %
%     .NP    -> double [nx(2n+1)]: Normalized P array                       %
%    .ANAng  -> struct array     : Normalized Tau, Pi, and P (acceptor)     %
%     .NTau  -> double [nx(2n+1)]: Normalized Tau array                     %
%     .NPi   -> double [nx(2n+1)]: Normalized Pi array                      %
%     .NP    -> double [nx(2n+1)]: Normalized P array                       %
%    .emphi  -> double [nx(2n+1)]: Normalized azimuthal functions           %
%    .Source -> struct array     : Source Expansion coefficients            %
%     .p     -> double [nx(2n+1)]: Coefficient p                           %
%     .q     -> double [nx(2n+1)]: Coefficient q                           %
%    .Scat   -> struct array     : Mie coefficients                         %
%     .alpha -> double (1xn)     : Mie coefficient alpha                    %
%     .beta  -> double (1xn)     : Mie coefficient beta                     %
```

```
%    .EdipS1 -> double (3x1)     : EdipField in S1 coordinate          %
%    .EdipS2 -> double (3x1)     : EdipField in S2 coordinate          %
% Outputs:                                                            %
%   Output   -> struct array     : Storage of output data              %
%    .Etot   -> double (3x1)     : Total electric field at APos        %
%    .Int    -> double           : Electric field intensity at APos    %
%    .Edip   -> double (3x1)     : Electric dipole field at APos       %
%    .NEtot  -> double (3x1)     : Normalized Etot at APos (Etot/Edip) %
% Temporary data:                                                     %
%   Temp     -> struct array     : Storage of temporary data           %
%    .AVSF   -> struct array     : Vector spherical function (acceptor)%
%     .M     -> double [nx(2n+1)]: Vector spherical function M         %
%     .N     -> double [nx(2n+1)]: Vector spherical function N         %
%    .ScatM  -> double (3x1)     : Scattering field (M part)           %
%    .ScatN  -> double (3x1)     : Scattering field (N part)           %
%                                                                      %
% !: Variables with "!" is required to make the module work.          %
% **:                                                                 %
%  The value of p depends on the spherical scatterer:                 %
%   Single sphere     -> p = 2                                        %
%   Core/shell sphere -> p = 3                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Output = EFieldR0(Settings)
    % Variables
    nmax = Settings.nmax;
    rhoD = Settings.nr(1)*Settings.k0*Settings.DPos.Sph(1);
    rhoA = Settings.nr(1)*Settings.k0*Settings.APos.Sph(1);
    % Radial Functions
    if isfield(Settings,'DRad') == 0
        Settings.DRad = SphBessel(rhoD,nmax,1,'hankel1');
    end
    if isfield(Settings,'ARad') == 0
        Settings.ARad = SphBessel(rhoA,nmax,1,'hankel1');
    end
    % Angular Functions
    if isfield(Settings,'DNAng') == 0
        Settings.DNAng = NormTauPiP(nmax,Settings.DPos.Sph(2),'reversed');
    end
    if isfield(Settings,'ANAng') == 0
        Settings.ANAng = NormTauPiP(nmax,Settings.APos.Sph(2),'normal');
    end
    % Azimuthal Functions
    if isfield(Settings,'emphi') == 0
        if Settings.APos.Sph(3) == 0
            % Speed-Up
            emphi = sqrt(1/2/pi);
        else
            % Setting exp(-inf) = 0 for Useless Array Elements
            m = -inf*ones(nmax,2*nmax+1);
            for ii = 1:nmax
                    m(ii,1:2*ii+1) = -ii:1:ii;
            end
            emphi = sqrt(1/2/pi)*exp(1i*m*Settings.APos.Sph(3));
```

```
            % Change exp(-inf) = NaN to Zero
            emphi(isnan(emphi)) = 0;
        end
    end
    % Source Coefficients
    if isfield(Settings,'Source') == 0
        Settings.Source = SourCoeff(Settings,'dipole');
    end
    % Mie Coefficients
    if isfield(Settings,'Scat') == 0
        if strcmp(Settings.BC,'sphere') == 1
            Settings.Scat = MieSingle(Settings.nr,Settings.k0s,nmax);
        elseif strcmp(Settings.BC,'coreshell') == 1
            Settings.Scat = MieCoreShell(Settings.nr,Settings.k0s,nmax);
        end
        Settings.Scat.a=Settings.Source.p.*transpose(Settings.Scat.alpha);
        Settings.Scat.b=Settings.Source.q.*transpose(Settings.Scat.beta);
    end
    % Generating M and N Fields
    Temp.AVSF = VectSphFunc(rhoA,nmax,Settings.ARad,Settings.ANAng,emphi);
    % Donor Dipole Field
    if isfield(Settings,'EdipS1') == 0
        if isfield(Settings,'EdipS2') == 0
            if isfield(Settings.APos,'Sph2') == 0
                Settings.APos.Sph2 = ...
                    C2S(Settings.APos.Cart-Settings.DPos.Cart);
            end
            % Field in the Secondary Coordinate
            Settings.EdipS2 = EdipField(Settings.nr(1),Settings.k0,...
                Settings.APos.Sph2,Settings.DOri.Cart);
        end
        % Transforming to the Primary Coordinate
        Settings.EdipS1 = S2S(Settings.EdipS2, ...
            (Settings.APos.Sph2(2)-Settings.APos.Sph(2)),0);
    end
    % Summing All order of the Scattering Field
    Temp.ScatM = reshape(sum(Temp.AVSF.M.*Settings.Scat.b,[1,2]),[3,1]);
    Temp.ScatN = reshape(sum(Temp.AVSF.N.*Settings.Scat.a,[1,2]),[3,1]);
    % Total Electric Field at the Acceptor Position
    Output.Etot = Temp.ScatM + Temp.ScatN + Settings.EdipS1;
    % Total Intensity at the Acceptor Position
    Output.Int = norm(Temp.ScatM + Temp.ScatN + Settings.EdipS1).^2;
    % Dipole Field
    Output.Edip =  Settings.EdipS1;
    % Etot / Edip
    Output.NEtot = Output.Etot./Settings.EdipS1;
    % Other Additional Outputs
        % Note: Feel Free to Add What You Want!
        % Use the Structure Array to Output Data
        % Example: output.testAPos = Settings.APos.Sph2;
end
```

Function 17: Module of calculating electric fields in region 0

Total Code Structure

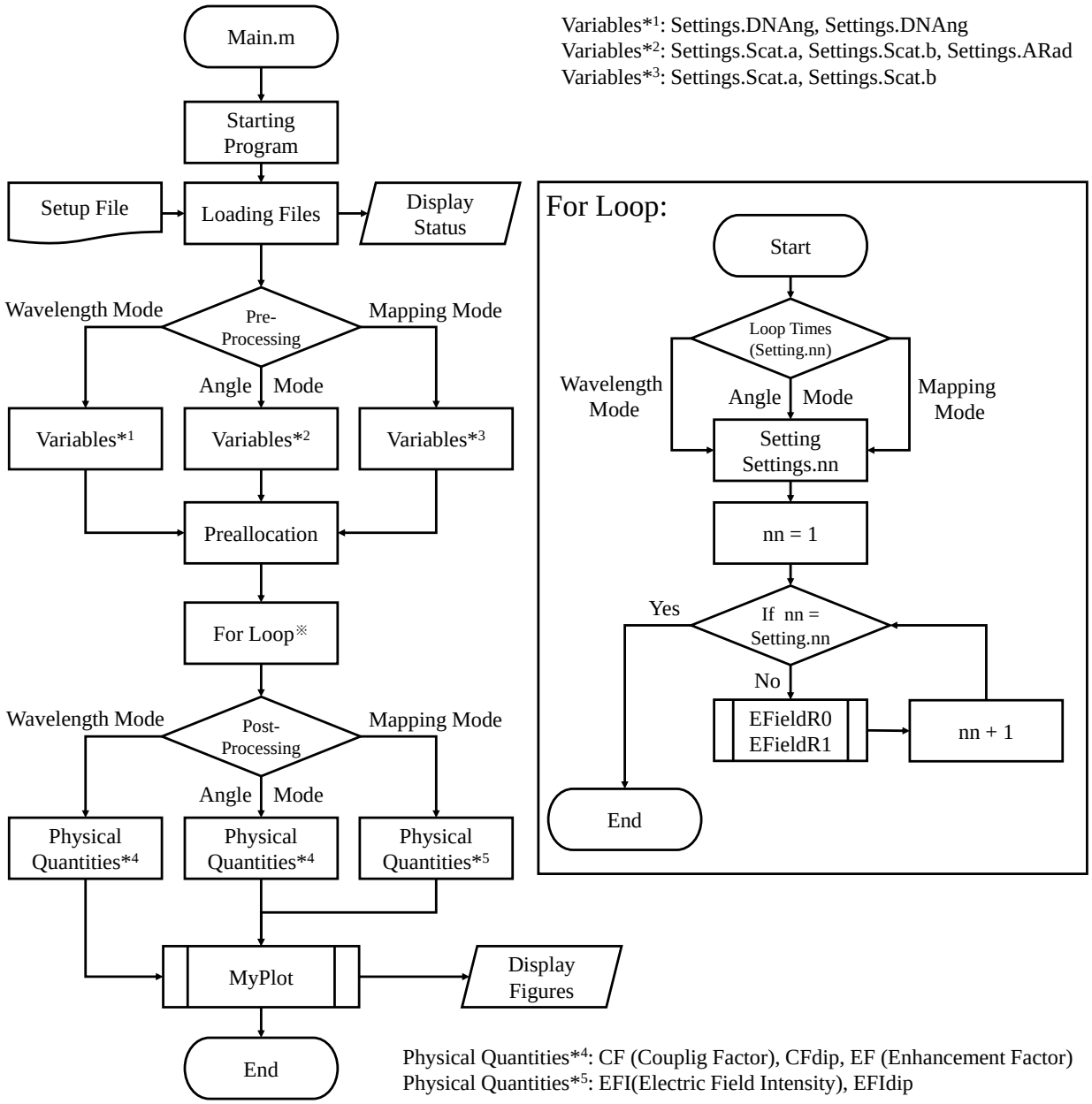

Figure 6: Computation flowchart of generalized Mie theory

The calculation of generalized Mie theory is accomplished by `Main.m`, which performs loading the setup file, selecting the calculation mode, computing physical quantities, and outputting of figures, as shown in Fig. 6. To initiate the program, we need to input the setting file of calculation details. The file path and the file name should be filled in line 17 and 18. In

addition, the size of the output figures can be tuned by a dimensionless scaling factor in line 19, in order to prevent the figures from exceeding screen size. Also note that non-built-in functions are called in `main.m` and the source code can be found in Appendix L.

MATLAB program for calculating electric fields based on generalized Mie theory

```
%% Code of Generalized Mie Theory
% Version: 2.4 (2022.04.29)

% Changes in v2.3
% (1) Fix the error when nmax = 3
% (2) Add a new mode for calculating Purcell factors

% Changes in v2.4
% (1) Redefine variables and their naming

% By Ming-Wei Lee

%% Starting Program
% Clean the Workspace
clear
% File to be Calculated
FilePath = '.\InputFile\'; % Folder Path of Input Files
FileName = 'Demo_MappingMode'; % File Name
% Output Figure Size (value = 0~1)
Resize = 1;

%% Information of initiate a job
tic
dispstat('','init'); % One time only initialization
dispstat(sprintf('Beginning the program...'),'keepthis','timestamp');

%% Loading the Input File
run(append(FilePath,FileName,'.m'));
% Information of the Input File
dispstat(append('The file is imported:'),'keepthis','timestamp');
fprintf(2,append(FileName,'\n'));
% Information of the Using Mode
dispstat(sprintf(append('Using mode: ', Settings.ModeName)),...
    'keepthis','timestamp');
% Information of the Using Structure
dispstat(sprintf(append('Using structure: ',Settings.BC)),...
    'keepthis','timestamp');

%% Pre-Processing (Reducing Computation Time)
% Transforming Coordinate
Settings.DPos.Sph = C2S(Settings.DPos.Cart);
Settings.DOri.Sph = ...
    VecTrans(Settings.DOri.Cart,Settings.DPos.Sph(2:3),'C2S');
% Pre-Calculation of Fixed Variables for Each Mode
if strcmp(Settings.ModeName,'wavelength') == 1
```

```
46      % Times of the 'for loop'
47      Settings.nn = size(Settings.nr,1);
48      % Coordinate Transformation
49      Settings.APos.Sph = C2S(Settings.APos.Cart);
50      Settings.AOri.Sph = ...
51          VecTrans(Settings.AOri.Cart,Settings.APos.Sph(2:3),'C2S');
52      Settings.APos.Sph2 = C2S(Settings.APos.Cart-Settings.DPos.Cart);
53      % Angular Functions
54      Settings.DNAng = ...
55          NormTauPiP(Settings.nmax,Settings.DPos.Sph(2),'reversed');
56      Settings.ANAng = ...
57          NormTauPiP(Settings.nmax,Settings.APos.Sph(2),'normal');
58  elseif strcmp(Settings.ModeName,'angle') == 1
59      % Times of the 'for loop'
60      Settings.nn = size(Settings.APos.Sph,2);
61      % Coordinate Transformation
62      Settings.AOri.Sph = ...
63          VecTrans(Settings.AOri.Cart,Settings.APos.Sph(2:3),'C2S');
64      % Radial Functions
65      rhoD = Settings.nr(1)*Settings.k0*Settings.DPos.Sph(1);
66      Settings.DRad = SphBessel(rhoD,Settings.nmax,1,'hankel1');
67      % Angular Functions
68      Settings.DNAng = ...
69          NormTauPiP(Settings.nmax,Settings.DPos.Sph(2),'reversed');
70      % Source Coefficients
71      Settings.Source = SourCoeff(Settings,'dipole');
72      if Settings.APos.Sph(1) >= Settings.rbc(1)
73          % Scattering Coefficients
74          if strcmp(Settings.BC,'sphere') == 1
75              Settings.Scat = ...
76                  MieSingle(Settings.nr,Settings.k0s,Settings.nmax);
77          elseif strcmp(Settings.BC,'coreshell') == 1
78              Settings.Scat = ...
79                  MieCoreShell(Settings.nr,Settings.k0s,Settings.nmax);
80          end
81          Settings.Scat.a = ...
82              Settings.Source.p.*transpose(Settings.Scat.alpha);
83          Settings.Scat.b = ...
84              Settings.Source.q.*transpose(Settings.Scat.beta);
85      else
86          % Layer1 Coefficients
87          if strcmp(Settings.BC,'sphere') == 1
88                  Settings.Layer1 = ...
89                      MieSingle(Settings.nr,Settings.k0s,Settings.nmax);
90              elseif strcmp(Settings.BC,'coreshell') == 1
91                  str1 = 'Core/shell mapping is not yet fully supported.\n';
92                  str2 = 'Overwrite E-field in the inner region by zero.\n';
93                  fprintf(2,str1);
94                  fprintf(2,str2);
95                  Settings.Layer1.gamma = 0;
96                  Settings.Layer1.delta = 0;
97          end
98          Settings.Layer1.d = ...
99              Settings.Source.p.*transpose(Settings.Layer1.delta);
```

```
100         Settings.Layer1.c = ...
101             Settings.Source.q.*transpose(Settings.Layer1.gamma);
102     end
103     % Radial Functions of the Acceptor
104     rhoA = Settings.nr(1)*Settings.k0*Settings.APos.Sph(1);
105     Settings.ARad = SphBessel(rhoA,Settings.nmax,1,'hankel1');
106 elseif strcmp(Settings.ModeName,'mapping') == 1
107     % Times of the 'for loop'
108     Settings.nn = size(Settings.APos.Cart,2);
109     % Coordinate Transformation
110     Settings.APos.Sph = C2S(Settings.APos.Cart);
111     % Radial Functions
112     rhoD = Settings.nr(1)*Settings.k0*Settings.DPos.Sph(1);
113     Settings.DRad = SphBessel(rhoD,Settings.nmax,1,'hankel1');
114     % Angular Functions
115     Settings.DNAng = ...
116         NormTauPiP(Settings.nmax,Settings.DPos.Sph(2),'reversed');
117     % Source Coefficients
118     Settings.Source = SourCoeff(Settings,'dipole');
119     % Scattering Coefficients
120     if strcmp(Settings.BC,'sphere') == 1
121         Settings.Scat = ...
122             MieSingle(Settings.nr,Settings.k0s,Settings.nmax);
123     elseif strcmp(Settings.BC,'coreshell') == 1
124         Settings.Scat = ...
125             MieCoreShell(Settings.nr,Settings.k0s,Settings.nmax);
126     end
127     % Layer1 Coefficients
128     if strcmp(Settings.BC,'sphere') == 1
129             Settings.Layer1 = ...
130                 MieSingle(Settings.nr,Settings.k0s,Settings.nmax);
131         elseif strcmp(Settings.BC,'coreshell') == 1
132             str1 = 'Core/shell mapping is not yet fully supported.\n';
133             str2 = 'Overwrite E-field in the inner region by zero.\n';
134             fprintf(2,str1);
135             fprintf(2,str2);
136             Settings.Layer1.gamma = 0;
137             Settings.Layer1.delta = 0;
138     end
139     Settings.Scat.a = ...
140         Settings.Source.p.*transpose(Settings.Scat.alpha);
141     Settings.Scat.b = ...
142         Settings.Source.q.*transpose(Settings.Scat.beta);
143     Settings.Layer1.d = ...
144         Settings.Source.p.*transpose(Settings.Layer1.delta);
145     Settings.Layer1.c = ...
146         Settings.Source.q.*transpose(Settings.Layer1.gamma);
147 elseif strcmp(Settings.ModeName,'Purcell') == 1
148     % Times of the 'for loop'
149     Settings.nn = size(Settings.nr,1);
150     % Angular Functions
151     Settings.DNAng = ...
152         NormTauPiP(Settings.nmax,Settings.DPos.Sph(2),'reversed');
153     Settings.DNAngN = ...
```

```
        NormTauPiP(Settings.nmax,Settings.DPos.Sph(2),'normal');
end

%% Preallocation
if strcmp(Settings.ModeName,'Purcell') == 1
    EScat = zeros(Settings.nn,3);
    Purcell = zeros(Settings.nn,1);
else
    Etot = zeros(Settings.nn,3);
    NormEtot = zeros(Settings.nn,3);
    Edip = zeros(Settings.nn,3);
end

%% Main Loop
if strcmp(Settings.ModeName,'wavelength') == 1
    for ii = 1:Settings.nn
        Settings.k0 = k0(ii);
        Settings.nr = nr(ii,:);
        Settings.k0s = k0s(ii,:);
        % Determing which Function is Called by the Acceptor Position
        if Settings.APos.Sph(1) >= Settings.rbc(1)
            Output = EFieldR0(Settings);
        else
            Output = EFieldR1(Settings);
        end
        Etot(ii,:) = transpose(Output.Etot);
        Edip(ii,:) = transpose(Output.Edip);
        NormEtot(ii,:) = transpose(Output.NEtot);
        % Information
        dispstat(sprintf('Progress: %.2f%%',(ii/Settings.nn)*100),...
            'timestamp');
    end
elseif strcmp(Settings.ModeName,'angle') == 1
    for ii = 1:Settings.nn
        Settings.APos.Cart = [Ax(ii); Ay(ii); Az(ii)];
        Settings.APos.Sph = [Ar; Atheta(ii); Aphi];
        % Determing which Function is Called by the Acceptor Position
        if Settings.APos.Sph(1) >= Settings.rbc(1)
            Output = EFieldR0(Settings);
        else
            Output = EFieldR1(Settings);
        end
        Etot(ii,:) = transpose(Output.Etot);
        Edip(ii,:) = transpose(Output.Edip);
        % Information
        dispstat(sprintf('Progress: %.2f%%',(ii/Settings.nn)*100),...
            'timestamp');
    end
elseif strcmp(Settings.ModeName,'mapping') == 1
    tmp1 = Settings.APos.Cart;
    tmp2 = Settings.APos.Sph;
    for ii = 1:Settings.nn
        Settings.APos.Cart =tmp1(:,ii);
        Settings.APos.Sph =tmp2(:,ii);
```

```
208         if Settings.APos.Sph(1) >= Settings.rbc(1)
209             Output = EFieldR0(Settings);
210         else
211             Output = EFieldR1(Settings);
212         end
213         Etot(ii,:) = transpose(Output.Etot);
214         Edip(ii,:) = transpose(Output.Edip);
215         % Information
216         dispstat(sprintf('Progress: %.2f%%',(ii/Settings.nn)*100),...
217             'timestamp');
218     end
219 elseif strcmp(Settings.ModeName,'Purcell') == 1
220     for ii = 1:Settings.nn
221         Settings.k0 = k0(ii);
222         Settings.nr = nr(ii,:);
223         Settings.k0s = k0s(ii,:);
224         % Calculation of Purcell Factor
225         Output = PurcellR0(Settings);
226         EScat(ii,:) = transpose(Output.EScat);
227         Purcell(ii,:) = transpose(Output.Purcell);
228         % Information
229         dispstat(sprintf('Progress: %.2f%%',(ii/Settings.nn)*100),...
230             'timestamp');
231     end
232 end
233
234 %% Output Warnings
235 if isfield(Output,'error1') ==1
236         fprintf(2,append(Output.error1,'\n'));
237 end
238
239 %% Post-Processing
240 if strcmp(Settings.ModeName,'wavelength') == 1
241   % Coupling Factor
242     CF = abs(Etot*Settings.AOri.Sph).^2;
243     % Coupling Factor along R Direction (Vacuum)
244     CFdip = abs(Edip*Settings.AOri.Sph).^2;
245     % Setting 0/0 to 0 for Etot/Edip
246     NormEtot(isnan(NormEtot)) = 0;
247     % Enhancement Factor
248     EF = abs(NormEtot*Settings.AOri.Sph).^2;
249 elseif strcmp(Settings.ModeName,'angle') == 1
250     % Coupling Factor along R Direction (Spheres)
251     CF = abs(Etot*[1;0;0]).^2;
252     % Coupling Factor along R Direction (Vacuum)
253     CFdip = abs(Edip*[1;0;0]/sqrt(1)).^2;
254 elseif strcmp(Settings.ModeName,'mapping') == 1
255     % Electric Field Intensity (Spheres)
256     EFI = vecnorm(Etot,2,2).^2;
257     % Reshape the Array
258     EFImap = reshape(EFI,size(Az));
259     % Electric Field Intensity (Vacuum)
260     EFIdip = vecnorm(Edip,2,2).^2;
261     % Reshape the Array
```

```
    EFIdipmap = reshape(EFIdip,size(Az));
end

%% Plotting Figures
if strcmp(Settings.ModeName,'wavelength') == 1
    fplot.x = 1./lambda*1e4;
    fplot.y = CF*1e24;
    MyPlot(fplot,Resize,0);
    fplot.y = CFdip*1e24;
    fplot.colorstyle = 'r-';
    MyPlot(fplot,Resize,1);
    fplot.y = EF;
    fplot.colorstyle = 'k-';
    fplot.range = [-inf,inf,1e-2,1e5];
    MyPlot(fplot,Resize,0);
elseif strcmp(Settings.ModeName,'angle') == 1
  fplot.x = Ar*Atheta/Settings.lambda;
    fplot.y = CF*1e24;
    MyPlot(fplot,Resize,0);
    fplot.y = CFdip*1e24;
    fplot.colorstyle = 'b--';
    MyPlot(fplot,Resize,1);
elseif strcmp(Settings.ModeName,'mapping') == 1
    contourf(Ax*1e3,Az*1e3,log10(EFImap*1e24),300,'linestyle','none');
    colormap jet
    colorbar
    hold on
    x = Settings.rbc(1)*linspace(-1,1,101);
    y = sqrt(Settings.rbc(1)^2 - x.^2);
    plot(x*1e3,-y*1e3,'-k','linewidth',2);
    plot(x*1e3,y*1e3,'-k','linewidth',2);
elseif strcmp(Settings.ModeName,'Purcell') == 1
    fplot.x = lambda*1e3;
    fplot.y = Purcell;
    MyPlot(fplot,Resize,0);
end

%% Output information
dispstat('Computation is Finished.','keepprev','timestamp');
toc
```

Setup an Input File

In this subsection, we describe how to construct a setting file to perform a job. In the current version, `main.m` supports four calculation modes, including wavelength, angle, field mapping, and Purcell factor. For example, the setting under the wavelength mode is shown at the bottom. First, we need to assign the position and the orientation of the donor and acceptor dipole in lines 4-15. Second, the maximum expansion order is given in line 17. Third, we set the properties of the dielectric environment in lines 19-51, including the type of the scatterer (`'sphere'` or `'coreshell'`), the frequency-dependent dielectric functions, and the radius of the scatterer. In addition, the default output of figures is adjustable in lines 55-62 if the modification is needed. Finally, the parameters set in lines 4-52 will be packaged to a structure array in lines 68-93. Setting files for the other modes can be found in Appendix L.

```
1  %% Input for wavelength mode (feel free to change variables)
2  % Dipole position (micrometer)
3      % Donor dipole
4      Dx =  0; %Do not change the value! Extremely Important!
5      Dy =  0; %Do not change the value! Extremely Important!
6      Dz =  0.100;
7      % Acceptor dipole
8      Ax =  0;
9      Ay =  0;
10     Az = -0.100;
11 % Dipole orientation
12     % Donor dipole (Cartesian coordinate, [x;y;z])
13     Doc = [0;0;1];
14     % Acceptor dipole (Cartesian coordinate, [x;y;z])
15     Aoc = [0;0;1];
16 % Largest expansion order (n) in a calculation
17 nmax = 70;
18 % Boundary conditions ('sphere' or 'coreshell')
19 BC = 'sphere';
20     % Calling dielectric data
21     excelre = ...
22        xlsread('.\DielectricFunction\dielectric function.xlsx','Ag_JPCL');
23     epsilonmat = excelre(:,2) + 1i*excelre(:,3);
24     % Wavelength (microns)
25     lambda = excelre(:,1)*1e-3;
26     %Wavenumber (microns)
27     k0 = 2*pi./lambda;
28     % Setting conditions
29     if strcmp(BC,'sphere') == 1
```

```
        % Preallocation
        nr = zeros(size(epsilonmat,1),2);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = sqrt(epsilonmat);
        % Radius of the sphere (unit: micron)
        rbc = 0.085;
        % Dimensionless radial variable
        k0s = k0*rbc;
    elseif strcmp(BC,'coreshell') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),3);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = 2;
        % Relative refractive index (Region 2)
        nr(:,3) = sqrt(epsilonmat);
        % Radius of the core and the shell [shell, core] (unit: micron)
        rbc = [0.070, 0.060];
        % Dimensionless radial variable
        k0s = k0*rbc;
    end

%% Settings for Function "myplot" (Default of Exporting Figures )
fplot.colorstyle = '-k';
fplot.range = [-inf,inf,1e28,1e36];
fplot.yscale = 'log';
fplot.xlabel = '$\mathrm{Wavenumber}~(\mathrm{cm}^{-1})$';
fplot.ylabel = '$\mathrm{Coupling~Factor}~(\mathrm{cm}^{-6})$';
fplot.subaxis = 1;
fplot.subrange = [-inf,inf,-inf,inf];
fplot.subxlabel = '$\mathrm{Wavelength}~(\mathrm{nm})$';
% plot.subylabel = '$\mathrm{Coupling~Factor}~(\mathrm{cm}^{-6})$';

%% ------- Do not change the following settings ------- %%
% Creating a mode structure array
    % Mode name
    Settings.ModeName = 'wavelength';
    % Donor position
        % Cartesian coordinate
        Settings.DPos.Cart = [Dx;Dy;Dz];
    % Acceptor position
        % Cartesian coordinate
        Settings.APos.Cart = [Ax;Ay;Az];
    % Donor orientation
        % Cartesian coordinate
        Settings.DOri.Cart = Doc;
    % Acceptor orientation
        % Cartesian coordinate
        Settings.AOri.Cart = Aoc;
    % Expansion number
    Settings.nmax = nmax;
```

```
    % Type of boundary condition
    Settings.BC = BC;
    % Range of wavelength and wavenumber (data point)
    Settings.lambda = lambda;
    Settings.k0 = k0;
    % Dielectric function of the environment
    Settings.nr = nr;
    % Radial boundary condition
    Settings.rbc = rbc;
    % Radial dimensionless variable for boundary conditions
    Settings.k0s = k0s;

clearvars -except Settings lambda k0 nr k0s FilePath FileName fplot Resize
```

Appendix

A. Derivation of Eqs. (5) and (6)

To derive the decoupled differential equation of Maxwell's equations for electric fields, we take a curl on the Faraday's law:

$$\begin{aligned}
\nabla \times \nabla \times \mathbf{E}(\mathbf{r},\omega) &= i\omega \nabla \times \mathbf{B}(\mathbf{r},\omega) \\
&= i\omega\mu_0 \nabla \times \mathbf{H}(\mathbf{r},\omega) \\
&= \omega^2 \mu_0 \mathbf{D}(\mathbf{r},\omega) \\
&= \omega^2 \mu_0 \left[\epsilon_0 \epsilon_{\mathrm{r}}(\mathbf{r},\omega)\mathbf{E}(\mathbf{r},\omega) + \mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega)\right] \\
&= \frac{\omega^2}{c^2}\epsilon_{\mathrm{r}}(\mathbf{r},\omega)\mathbf{E}(\mathbf{r},\omega) + \frac{\omega^2}{\epsilon_0 c^2}\mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega). \qquad (133)
\end{aligned}$$

Utilizing the vector identity $\nabla \times \nabla \times \ = \nabla\nabla\cdot \ - \nabla^2$, the differential equation becomes:

$$\left[\frac{\omega^2\epsilon_{\mathrm{r}}(\mathbf{r},\omega)}{c^2} - \nabla \times \nabla \times\right]\mathbf{E}(\mathbf{r},\omega) = \left[\nabla^2 + \frac{\omega^2\epsilon_{\mathrm{r}}(\mathbf{r},\omega)}{c^2}\right]\mathbf{E}(\mathbf{r},\omega) = -\frac{\omega^2}{\epsilon_0 c^2}\mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega). \qquad (134)$$

Using the same trick on the Maxwell-Ampère equation, we reach the result that

$$\begin{aligned}
\nabla \times \nabla \times \mathbf{H}(\mathbf{r},\omega) &= -i\omega \nabla \times \mathbf{D}(\mathbf{r},\omega) \\
&= -i\omega \nabla \times \left[\epsilon_0 \epsilon_{\mathrm{r}}(\mathbf{r},\omega)\mathbf{E}(\mathbf{r},\omega) + \mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega)\right] \\
&= -i\omega\epsilon_0 \nabla\epsilon_{\mathrm{r}}(\mathbf{r},\omega) \times \mathbf{E}(\mathbf{r},\omega) - i\omega\epsilon_0\epsilon_{\mathrm{r}}(\mathbf{r},\omega)\nabla \times \mathbf{E}(\mathbf{r},\omega) - i\omega \nabla \times \mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega) \\
&\cong \omega^2 \mu_0 \epsilon_0 \epsilon_{\mathrm{r}}(\mathbf{r},\omega)\mathbf{H}(\mathbf{r},\omega) - i\omega \nabla \times \mathbf{P}_{\mathrm{D}}(\mathbf{r},\omega). \qquad (135)
\end{aligned}$$

In Eq. (135), we assume that the dielectric function is piecewise-homogeneous for each region. In other words, the dielectric function in i-th region is independent of position, $\epsilon_{\mathrm{r}}(\mathbf{r},\omega) \rightarrow \epsilon_{\mathrm{r},i}(\omega)$. Therefore, $\nabla\epsilon_{\mathrm{r}}(\mathbf{r},\omega) = 0$ except for $\mathbf{r}$ at boundary. Similarly, we get

another inhomogeneous differential equation for the magnetizing fields in the i-th region:

$$\left[\frac{\omega^2\epsilon_{\mathrm{r},i}(\mathbf{r},\omega)}{c^2} - \nabla\times\nabla\times\right]\mathbf{H}^{(i)}(\mathbf{r},\omega) = \left[\nabla^2 + \frac{\omega^2\epsilon_{\mathrm{r},i}(\mathbf{r},\omega)}{c^2}\right]\mathbf{H}^{(i)}(\mathbf{r},\omega)$$
$$= i\omega\nabla\times\sum_j \mathbf{P}_{\mathrm{D}}^{(j)}(\mathbf{r},\omega), \tag{136}$$

where the superscript (i) denotes the i-th region. In the derivation, the requirement of piecewise-homogeneous dielectric function is not necessary to get the result in Eq. (133). For the sake of consistency, we impose the additional condition to dielectric functions, which are presented in the tutorial [Eqs. (5) and (6)].

B. Derivation from Eq. (9) to Eq. (14)

Here, we aim to prove that the vector differential equation in Eq. (9) can be reduced to the scalar differential equation in Eq. (14). To get the result, we need to verify that

$$\left[k_i^2(\omega) - \nabla \times \nabla \times\right] \mathscr{T}_{\mathbf{K}} \left\{\phi^{(i)}(\mathbf{r},\omega)\right\} = 0 = \mathscr{T}_{\mathbf{K}} \left\{\left[k_i^2(\omega) + \nabla^2\right] \phi^{(i)}(\mathbf{r},\omega)\right\}, \tag{137}$$

where $\mathbf{K} = \{\mathbf{M}, \mathbf{N}\}$. Recall that $\mathscr{T}_{\mathbf{K}}\{\cdot\}$ transforms a scalar field to a transverse vector field [definitions can be found in Eq. (12) and Eq. (13)]. The thing we need to do is change the order of differential operators. From the definition, the differential equation of $\mathbf{M}^{(i)}(\mathbf{r},\omega)$ is

$$\left[k_i^2(\omega) - \nabla \times \nabla \times\right] \nabla \times \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right] = 0. \tag{138}$$

It is obvious to obtain that $k_i^2(\omega)\nabla \times \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right] = \nabla \times \left[\mathbf{r}k_i^2(\omega)\phi^{(i)}(\mathbf{r},\omega)\right]$ because the curl operator does not operate on $k_i(\omega)$. Moreover, the triple-curl operator can be simplified to

$$\begin{aligned}
\nabla \times \nabla \times \nabla \times \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right] &= \nabla \times \left\{\nabla\nabla \cdot \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right] - \nabla^2 \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right]\right\} && (139)\\
&= -\nabla \times \nabla^2 \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right] && (140)\\
&= -\nabla \times \left[\mathbf{r}\nabla^2\phi^{(i)}(\mathbf{r},\omega) + 2\nabla\phi^{(i)}(\mathbf{r},\omega)\right] && (141)\\
&= -\nabla \times \left[\mathbf{r}\nabla^2\phi^{(i)}(\mathbf{r},\omega)\right]. && (142)
\end{aligned}$$

In the first equation, we use the fact that $\nabla \times \nabla\times = \nabla\nabla \cdot -\nabla^2$. Also, because the curl of the gradient of any scalar function is identical to zero, only $\nabla \times \nabla^2 \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right]$ contributes a nonzero result. Finally, it shows that

$$\left[k_i^2(\omega) - \nabla \times \nabla \times\right] \nabla \times \left[\mathbf{r}\phi^{(i)}(\mathbf{r},\omega)\right] = \nabla \times \left\{\mathbf{r}\left[k_i^2(\omega) + \nabla^2\right] \phi^{(i)}(\mathbf{r},\omega)\right\} = 0. \tag{143}$$

The derivation of $\mathbf{N}^{(i)}(\mathbf{r},\omega)$ is similar to that of $\mathbf{M}^{(i)}(\mathbf{r},\omega)$. Readers can verify themselves.

C. Scalar Helmholtz Equation and its Eigenfunctions

The scalar Helmholtz equation is defined as $[\nabla^2 + k_i^2]\,\phi^{(i)}(\mathbf{r},\omega) = 0$. Recall that $k_i \equiv \underline{n}_i k_0$ and $k_0 = \omega/c$. Expanding Laplacian operator in the spherical coordinate, we get the equation:

$$\left[\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\hat{J}^2 + k_i^2\right]\phi^{(i)}(\mathbf{r},\omega) = 0, \tag{144}$$

where $\hat{J}^2$ is the angular part of Laplacian,

$$\hat{J}^2 = \frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}, \tag{145}$$

By using the separation of variables, i.e., $\phi^{(i)}(\mathbf{r},\omega) \sim R^{(i)}(r,\omega)Y(\theta,\phi)$, we divide the differential equation into the radical part

$$\left[\frac{\mathrm{d}}{\mathrm{d}r}\left(r^2\frac{\mathrm{d}}{\mathrm{d}r}\right) + (k_i r)^2 - n(n+1)\right]R^{(i)}(r,\omega) = 0, \tag{146}$$

and the angular part $\hat{J}^2 Y(\theta,\phi) + n(n+1)Y(\theta,\phi) = 0$. Furthermore, the angular function $Y(\theta,\phi)$ can be separated again to $\Theta(\theta)\Phi(\phi)$ and get the two equations,

$$\left[\frac{1}{\sin\theta}\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\sin\theta\frac{\mathrm{d}}{\mathrm{d}\theta}\right) + n(n+1) - \frac{m^2}{\sin^2\theta}\right]\Theta(\theta) = 0 \tag{147}$$

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}\phi^2} + m^2\right)\Phi(\phi) = 0. \tag{148}$$

The solution of Eq. (148), the azimuthal part, with the boundary condition $\Phi(\phi) = \Phi(\phi+2\pi)$ is $\Phi(\phi) = e^{im\phi},\ m \in \mathbb{Z}$. For the polar part [Eq. (147)], the solutions are associated Legendre polynomials,

$$\Theta(\theta) = P_n^m(\cos\theta), \qquad n \geq |m|. \tag{149}$$

Here, associated Legendre polynomials can be defined by Legendre polynomials $[P_n(\cos\theta)]$,

$$P_n^m(\cos\theta) = (-1)^m \sin^m\theta \frac{\mathrm{d}^m}{\mathrm{d}(\cos\theta)^m} P_n(\cos\theta), \qquad m \geq 0. \tag{150}$$

Because $m \in \mathbb{Z}$, which is requested by the azimuthal differential equation, we need to define associated Legendre polynomials for negative arguments of m. Customarily, we can define them using Rodrigues' formula; however, we do not adopt this scheme due to the redundant computation of normalization. According to Eq. (147), it is obvious that associated Legendre polynomials do not change when $m \to -m$. Intuitively thinking, we can set $P_n^{-m}(\cos\theta) = P_n^m(\cos\theta)$. However, in practice, we choose the convention in angular momentum theory, which includes the Condon-Shortley phase, $(-1)^m$,

$$P_n^{-m}(\cos\theta) \equiv (-1)^m P_n^m(\cos\theta), \qquad m \geq 0. \tag{151}$$

The additional phase on associated Legendre polynomials makes Eq. (127) hold, i.e.,

$$\langle\theta,\phi|\,\hat{J}_\pm\,|n,m\rangle = \langle\theta,\phi|\,C_\pm(n,m)\,|n,m\pm1\rangle\,. \tag{152}$$

This requirement allows us to compute associated Legendre polynomials via spectral methods, which provides high-precision numerical results when n is large. As for the radial-part function, the differential equation can be reduced to the well-known spherical Bessel differential equation via the transformation,

$$z(k_i r) = R^{(i)}(r,\omega)\cdot\sqrt{k_i r}. \tag{153}$$

The differential equation becomes:

$$\left\{k_i r\cdot\frac{\mathrm{d}}{\mathrm{d}(k_i r)}\left[(k_i r)\cdot\frac{\mathrm{d}}{\mathrm{d}(k_i r)}\right] + \left[(k_i r)^2 - \left(n+\frac{1}{2}\right)^2\right]\right\} z(k_i r) = 0, \tag{154}$$

The function $z(k_i r)$ denotes the independent solutions of the differential equation, spherical Bessel function of the first kind and the second kind, $\{j_n(k_i r), y_n(k_i r)\}$, and the spherical Hankel function of the first kind and the second kind, $\{h_n^{(1)}(k_i r), h_n^{(2)}(k_i r)\}$. Notice that both of the set are defined in the complex domain since the input variable $k_i r$ may be a complex number. In addition, the two spherical Hankel functions are defined by the complex linear combination of the two spherical Bessel functions:

$$h_n^{(1)}(\rho) = j_n(\rho) + iy_n(\rho), \tag{155}$$

$$h_n^{(1)}(\rho) = j_n(\rho) - iy_n(\rho). \tag{156}$$

D. Auxiliary Equations for proving the orthogonality of Vector Spherical Functions

In the discussion of orthogonality, we encounter the following two integrals with respect to solid angle,

$$\int i\left[\tau_{n'm'}(\theta)\pi_{nm}(\theta)+\pi_{n'm'}(\theta)\tau_{nm}(\theta)\right]e^{i(m-m')\phi}\,\mathrm{d}\Omega=0$$

$$\int\left[\tau_{n'm'}(\theta)\tau_{nm}(\theta)+\pi_{n'm'}(\theta)\pi_{nm}(\theta)\right]e^{i(m-m')\phi}\,\mathrm{d}\Omega=n(n+1)f_{nm}\delta_{nn'}\delta_{mm'}. \tag{157}$$

Here, we are going to prove the two equations. First, it is easy to obtain that the integral of the azimuthal part returns a Kronecker delta. In other words, cases for $m'\neq m$ is always identical to zero. Hence, we only need to consider the polar-part integrals for integrands that $m'=m$. For integrands of crossing terms of τ and π functions, the polar-part integral becomes

$$\begin{aligned}
&\int_0^\pi\left[\tau_{n'm}(\theta)\pi_{nm}(\theta)+\pi_{n'm}(\theta)\tau_{nm}(\theta)\right]\sin\theta\mathrm{d}\theta\\
&=m\int_0^\pi\left[P_n^m(\cos\theta)\frac{\mathrm{d}P_{n'}^m(\cos\theta)}{\mathrm{d}\theta}+P_{n'}^m(\cos\theta)\frac{\mathrm{d}P_n^m(\cos\theta)}{\mathrm{d}\theta}\right]\mathrm{d}\theta &(158)\\
&=mP_n^m(\cos\theta)P_{n'}^m(\cos\theta)|_0^\pi=0 &(159)
\end{aligned}$$

In Eq. (158), we apply the definitions of τ and π functions. Using the chain rule and the fundamental theorem of calculus, we obtain the result in Eq. (159). Equation (159) is equivalent to zero because $P_n^m(-1)=0=P_n^m(1)$ except for $m=0$. For the case of $m=0$, the equation still equals zero because the associated Legendre polynomials are multiplied by m beforehand. For the integrands of the summation of τ-square and π-square we use the

identity,

$$\begin{aligned}2\sin\theta\left[\tau_{n'm'}(\theta)\tau_{nm}(\theta)+\pi_{n'm'}(\theta)\pi_{nm}(\theta)\right] =& 2\sin\theta\left[\frac{\mathrm{d}P^m_{n'}}{\mathrm{d}\theta}\frac{\mathrm{d}P^m_n}{\mathrm{d}\theta}+m^2\frac{P^m_{n'}}{\sin\theta}\frac{P^m_n}{\sin\theta}\right]\\ =& 2\sin\theta\ n(n+1)P^m_{n'}P^m_n\\ &+\frac{\mathrm{d}}{\mathrm{d}\theta}\left[\sin\theta\ P^m_{n'}\ \frac{\mathrm{d}P^m_n}{\mathrm{d}\theta}+\sin\theta\ P^m_n\ \frac{\mathrm{d}P^m_{n'}}{\mathrm{d}\theta}\right], \quad (160)\end{aligned}$$

which makes the integral become

$$\begin{aligned}&\int\left[\tau_{n'm'}(\theta)\tau_{nm}(\theta)+\pi_{n'm'}(\theta)\pi_{nm}(\theta)\right]e^{i(m-m')\phi}\ \mathrm{d}\Omega\\ &=n(n+1)\int_0^\pi P^m_{n'}(\cos\theta)P^m_n(\cos\theta)\sin\theta\mathrm{d}\theta\int_0^{2\pi}e^{i(m-m')\phi}\ \mathrm{d}\phi=n(n+1)f_{nm}\delta_{nn'}\delta_{mm'}. \quad (161)\end{aligned}$$

Note that Eq. (160) can be derived from the definition of associated Legendre differential equation, as depicted in Eq. (147).

E. Expansion of Dyadic Delta Function in a Spherical Coordinate

According to the orthogonality of (partially) normalized vector spherical functions, $\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}$, $\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}$, and $\underline{\mathbf{N}}^{(\mathrm{I})}_{nm}$, the normalized vector spherical functions form a complete basis. Thus, a dyadic delta function can be expanded as follows,

$$\begin{aligned}\bar{\bar{\mathbf{I}}}\delta(\mathbf{r}-\mathbf{r}') = \int_0^\infty k^2 \mathrm{d}k \sum_{nm} \Big[& \underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \otimes \underline{\mathbf{A}}_{nm}(kr',\theta',\phi') \\ & + \underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \otimes \underline{\mathbf{B}}_{nm}(kr',\theta',\phi') \\ & + \underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \otimes \underline{\mathbf{C}}_{nm}(kr',\theta',\phi') \Big]. \end{aligned} \tag{162}$$

To solve the unknown vector fields: $\underline{\mathbf{A}}_{nm}(kr',\theta',\phi')$, $\underline{\mathbf{B}}_{nm}(kr',\theta',\phi')$, and $\underline{\mathbf{C}}_{nm}(kr',\theta',\phi')$, we evaluate the following integrals:

$$\mathrm{LHS}_{(\mathbf{A})} = \int \mathrm{d}^3\mathbf{r}\ (-1)^{m'} \underline{\mathbf{L}}^{(\mathrm{I})}_{n'(-m)'}(k'r,\theta,\phi) \cdot \bar{\bar{\mathbf{I}}}\delta(\mathbf{r}-\mathbf{r}') = (-1)^{m'} \underline{\mathbf{L}}^{(\mathrm{I})}_{n'(-m)'}(k'r',\theta',\phi') \tag{163}$$

$$\begin{aligned}\mathrm{RHS}_{(\mathbf{A})} =& \int \mathrm{d}^3\mathbf{r} \int_0^\infty k^2 \mathrm{d}k \sum_{nm}\ (-1)^{m'} \underline{\mathbf{L}}^{(\mathrm{I})}_{n'(-m)'}(k'r,\theta,\phi) \cdot \underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi) \otimes \underline{\mathbf{A}}_{nm}(kr',\theta',\phi') \\ =& \int_0^\infty k^2 \mathrm{d}k\ \underline{\mathbf{A}}_{nm}(kr',\theta',\phi') \cdot \frac{\pi\delta(k-k')}{2k^2} \delta_{nn'}\delta_{mm'} \\ =& \frac{\pi}{2}\ \underline{\mathbf{A}}_{n'm'}(k'r',\theta',\phi') \end{aligned} \tag{164}$$

According to the integral above, we get the relation,

$$\underline{\mathbf{A}}_{nm}(kr',\theta',\phi') = \frac{2}{\pi}\ (-1)^m \underline{\mathbf{L}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi'). \tag{165}$$

In the same way, the unknown vector fields, $\underline{\mathbf{B}}_{nm}(kr',\theta',\phi')$, and $\underline{\mathbf{C}}_{nm}(kr',\theta',\phi')$, are

$$\underline{\mathbf{B}}_{nm}(kr',\theta',\phi') = \frac{2}{\pi}\ (-1)^m \underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi'), \tag{166}$$

$$\underline{\mathbf{C}}_{nm}(kr',\theta',\phi') = \frac{2}{\pi}\ (-1)^m \underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi'). \tag{167}$$

Finally, the expansion of a dyadic delta function in a spherical coordinate becomes

$$\begin{aligned}\bar{\bar{\mathbf{I}}}\delta(\mathbf{r}-\mathbf{r}') = \frac{2}{\pi}\int_0^\infty k^2 \mathrm{d}k \sum_{nm}(-1)^m \Bigg[\quad & \underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{L}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \\ & + \underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \\ & + \underline{\mathbf{N}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\mathrm{I})}_{n(-m)}(kr',\theta',\phi') \Bigg]. \end{aligned} \tag{168}$$

F. Contour Integrals in the Free-Space Dyadic Green's Function

In discussing the spherical expansion of dyadic Green's functions, we need to cope with integrals associated with spherical Bessel functions. Here, we provide a complete derivation of these integrals. We start from the integral associated with $\underline{\mathbf{M}}^{(\mathrm{I})}_{nm}(kr,\theta,\phi)$ [Eq. (55)],

$$\begin{aligned}
&\int_0^\infty \frac{k^2}{k^2-k_0^2} j_n(kr') j_n(kr)\ \mathrm{d}k \\
&\quad = \frac{1}{2}\int_0^\infty \frac{k^2}{k^2-k_0^2}\left[h_n^{(1)}(kr') + h_n^{(2)}(kr')\right] j_n(kr)\ \mathrm{d}k \\
&\quad = \frac{1}{2}\left[\int_0^\infty \frac{k^2}{k^2-k_0^2} h_n^{(1)}(kr') j_n(kr)\ \mathrm{d}k - \int_0^{-\infty} \frac{k^2}{k^2-k_0^2} h_n^{(1)}(kr') j_n(kr)\ \mathrm{d}k\right] \\
&\quad = \frac{1}{2}\int_{-\infty}^\infty \frac{k^2}{k^2-k_0^2} h_n^{(1)}(kr') j_n(kr)\ \mathrm{d}k,
\end{aligned} \tag{169}$$

where we apply the identities, $j_n(kr') = h_n^{(1)}(kr') + h_n^{(2)}(kr')$ and $h_n^{(2)}(-kr') = (-1)^n h_n^{(1)}(kr')$, to change the interval of the second integral. Next, we use contour integration to calculate the integral. We set the whole close path to be the real axis plus a semicircular path at $k \to \infty$ in the upper half-plane [Fig. 7 (a)],

$$\begin{aligned}
&\int_{-\infty}^\infty \frac{k^2}{k^2-k_0^2} h_n^{(1)}(kr') j_n(kr)\ \mathrm{d}k \\
&\quad = \lim_{\epsilon\to 0^+}\int_{-\infty}^\infty \frac{k^2 h_n^{(1)}(kr') j_n(kr)}{(k+k_0+i\epsilon)(k-k_0-i\epsilon)}\ \mathrm{d}k \\
&\quad = \lim_{\epsilon\to 0^+}\left[\oint_C \frac{\tilde{k}^2 h_n^{(1)}(\tilde{k}r') j_n(\tilde{k}r)\ \mathrm{d}\tilde{k}}{(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)} - \lim_{k\to\infty}\int_0^\pi \frac{\tilde{k}^2 h_n^{(1)}(\tilde{k}r') j_n(\tilde{k}r)\ i\tilde{k}\mathrm{d}\theta}{(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)}\right],
\end{aligned} \tag{170}$$

where $\tilde{k} = ke^{i\theta}$ and ϵ denotes an infinitesimal dissipation. According to the residue theorem, the contour integral of path C_1 becomes

$$\begin{aligned}
\lim_{\epsilon\to 0^+}\oint_{C_1} \frac{\tilde{k}^2 h_n^{(1)}(\tilde{k}r') j_n(\tilde{k}r)\ \mathrm{d}\tilde{k}}{(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)} &= \lim_{\epsilon\to 0^+} 2\pi i\cdot \mathrm{Res}\left[\frac{\tilde{k}^2 h_n^{(1)}(\tilde{k}r') j_n(\tilde{k}r)}{(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)}, k_0+i\epsilon\right] \\
&= \pi i k_0 h_n^{(1)}(k_0 r') j_n(k_0 r)
\end{aligned} \tag{171}$$

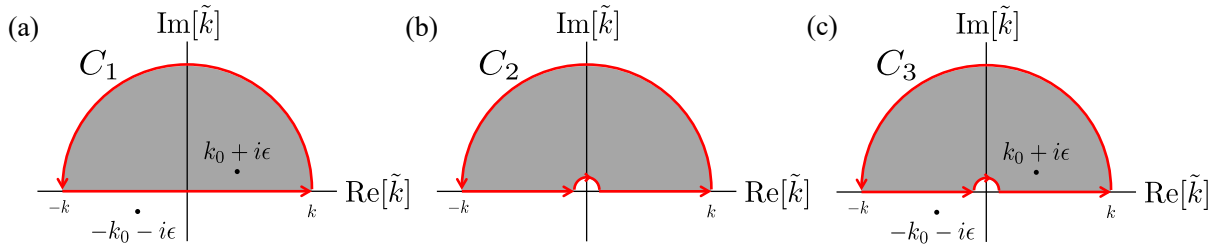

Figure 7: The paths of contour integrals in (a) Eq. (170), (b) Eq. (177), and (c) Eq. (181)

According to the asymptotic expressions of spherical Bessel (Hankel) functions, $j_n(\tilde{k}r) \sim [e^{i(\tilde{k}r-n\pi/2)} - e^{-i(\tilde{k}r-n\pi/2)}]/(2i\tilde{k}r)$ and $h_n^{(1)}(\tilde{k}r') \sim e^{i\tilde{k}r'}/(i^{n+1}\tilde{k}r')$, the integral of semicircular path in the upper half-plane becomes

$$\lim_{\epsilon\to 0^+}\lim_{k\to\infty}\int_0^\pi \frac{\tilde{k}^2 h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\; i\tilde{k}\mathrm{d}\theta}{(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)} = \lim_{\epsilon\to 0^+}\lim_{k\to\infty}\int_0^\pi \frac{\tilde{k}\left[e^{i\tilde{k}(r'+r)-in\pi/2} - e^{i\tilde{k}(r'-r)+in\pi/2}\right]\;\mathrm{d}\theta}{i^{-n-1}2rr'(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)}$$
$$= 0, \qquad r < r'. \tag{172}$$

Moreover, by using Jordan's lemma, if $r < r'$, the integral along the semicircular path in the upper half-plane becomes zero as $k \to \infty$. Hence, we get the result

$$\int_0^\infty \frac{k^2}{k^2-k_0^2} j_n(kr')j_n(kr)\;\mathrm{d}k = \frac{1}{2}\int_{-\infty}^\infty \frac{k^2}{k^2-k_0^2} h_n^{(1)}(kr')j_n(kr)\;\mathrm{d}k$$
$$= \frac{\pi}{2}\cdot ik_0 h_n^{(1)}(k_0r')j_n(k_0r), \qquad r < r'. \tag{173}$$

To evaluate the behavior in the region $r > r'$, we can use the same trick to replace $j_n(kr)$ by $[h_n^{(1)}(kr) + h_n^{(1)}(kr)]/2$ and do the same operation described above. Finally, we get the equation,

$$\int_0^\infty \frac{k^2}{k^2-k_0^2} j_n(kr)j_n(kr')\;\mathrm{d}k = \frac{\pi}{2}\cdot ik_0 h_n^{(1)}(k_0r_>)j_n(k_0r_<), \tag{174}$$

where $r_> = \max(\mathbf{r}, \mathbf{r}')$ and $r_< = \min(\mathbf{r}, \mathbf{r}')$. Second, we evaluate the integral in Eq. (50) which is associated with $\underline{\mathbf{L}}^{(\mathrm{I})}_{nm}(kr, \theta, \phi)$,

$$\begin{aligned}\int_0^\infty j_n(kr')j_n(kr)\ \mathrm{d}k &= \frac{1}{2}\int_0^\infty \left[h_n^{(1)}(kr') + h_n^{(2)}(kr')\right] j_n(kr)\ \mathrm{d}k \\ &= \frac{1}{2}\lim_{\delta\to 0}\left[\int_\delta^\infty h_n^{(1)}(kr')j_n(kr)\ \mathrm{d}k - \int_{-\delta}^{-\infty} h_n^{(1)}(kr')j_n(kr)\ \mathrm{d}k\right] \\ &= \frac{1}{2}\mathrm{PV}\int_{-\infty}^{\infty} h_n^{(1)}(kr')j_n(kr)\ \mathrm{d}k. \end{aligned} \tag{175}$$

The integral is defined by the Cauchy principal value because a pole exists at $k = 0$. The pole can be identified by the asymptotic expansion of spherical Bessel (Hankel) functions,

$$h_n^{(1)}(kr')j_n(kr) \sim \frac{(2n-1)!!}{i(kr')^{n+1}}\frac{(kr)^n}{(2n+1)!!} = \frac{1}{ik(2n+1)}\frac{r^n}{(r')^{n+1}}, \qquad k \to 0. \tag{176}$$

To calculate the integral, we evaluate a contour integral along the path illustrated in Fig. 7 (b), where an infinitesimally-semicircular path is to bypass the pole.

$$\begin{aligned}\mathrm{PV}\int_{-\infty}^{\infty} h_n^{(1)}(kr')j_n(kr)\ \mathrm{d}k &= \oint_C h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\ \mathrm{d}\tilde{k} \\ &\quad - \lim_{k\to\infty}\int_0^\pi h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\ i\tilde{k}\mathrm{d}\theta - \lim_{k\to 0}\int_\pi^0 h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\ i\tilde{k}\mathrm{d}\theta \\ &= \lim_{k\to 0}\int_0^\pi h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\ i\tilde{k}\mathrm{d}\theta. \end{aligned} \tag{177}$$

For the same reason (Jordan's lemma), the integral along the semicircular path at $k \to \infty$ is zero when $r < r'$. In addition, because the integrand is analytic in the upper half-plane, the Cauchy principal value can be evaluated by the integral of infinitesimally semicircular path, as shown in Eq. (177). According to the integral,

$$\lim_{k\to 0}\int_0^\pi h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\ i\tilde{k}\mathrm{d}\theta = \lim_{k\to 0}\int_0^\pi \frac{1}{i\tilde{k}(2n+1)}\frac{r^n}{(r')^{n+1}}\ i\tilde{k}\mathrm{d}\theta = \frac{\pi}{(2n+1)}\frac{r^n}{(r')^{n+1}}, \tag{178}$$

we derive the closed-form expression for general cases (the same technique described former),

$$\int_0^\infty j_n(kr')j_n(kr)\mathrm{d}k = \frac{1}{2}\mathrm{PV}\int_{-\infty}^{\infty} h_n^{(1)}(kr')j_n(kr) = \frac{\pi}{2(2n+1)}\frac{r_<^n}{r_>^{n+1}}. \tag{179}$$

The last integral, which is associated with $\underline{\mathbf{N}}_{nm}^{(\mathrm{I})}(kr,\theta,\phi)$, can be described by the Cauchy principal value,

$$\int_0^\infty \frac{j_n(kr')j_n(kr)}{k^2-k_0^2}\,\mathrm{d}k = \frac{1}{2}\mathrm{PV}\int_{-\infty}^{\infty} \frac{h_n^{(1)}(kr')j_n(kr)}{k^2-k_0^2}\,\mathrm{d}k. \tag{180}$$

We choose the path the same as the second integral to evaluate this Cauchy principal value; however, the integrand is no longer analytic in the upper half-plane because we add an infinitesimal imaginary part to k_0 to bypass the poles on the real line.

$$\begin{aligned}\mathrm{PV}\int_{-\infty}^{\infty} \frac{h_n^{(1)}(kr')j_n(kr)}{k^2-k_0^2}\,\mathrm{d}k = &\lim_{\epsilon\to 0}\oint_C \frac{h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)}{(\tilde{k}-k_0-i\epsilon)(\tilde{k}+k_0+i\epsilon)}\,\mathrm{d}\tilde{k}\\ &-\lim_{\epsilon\to 0}\lim_{k\to\infty}\int_0^\pi \frac{h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)}{(\tilde{k}-k_0-i\epsilon)(\tilde{k}+k_0+i\epsilon)}\,i\tilde{k}\mathrm{d}\theta\\ &-\lim_{\epsilon\to 0}\lim_{k\to 0}\int_\pi^0 \frac{h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)}{(\tilde{k}-k_0-i\epsilon)(\tilde{k}+k_0+i\epsilon)}\,i\tilde{k}\mathrm{d}\theta.\end{aligned} \tag{181}$$

According to the residue theorem, the closed path contour integral becomes

$$\lim_{\epsilon\to 0}\oint_C \frac{h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)\,\mathrm{d}\tilde{k}}{(\tilde{k}+k_0+i\epsilon)(\tilde{k}-k_0-i\epsilon)} = \frac{\pi i}{k_0}h_n^{(1)}(k_0r')j_n(k_0r) \tag{182}$$

In addition, the integral along the infinitesimal semicircular path becomes

$$\lim_{\epsilon\to 0}\lim_{k\to 0}\int_\pi^0 \frac{h_n^{(1)}(\tilde{k}r')j_n(\tilde{k}r)}{(\tilde{k}-k_0-i\epsilon)(\tilde{k}+k_0+i\epsilon)}\,i\tilde{k}\mathrm{d}\theta = \frac{\pi}{k_0^2(2n+1)}\frac{r^n}{(r')^{n+1}}. \tag{183}$$

Again, the integral along the semicircular path at $k\to\infty$ becomes zero. Eventually, the

closed-form expression of Eq. (181) is

$$\mathrm{PV}\int_{-\infty}^{\infty}\frac{h_n^{(1)}(kr')j_n(kr)}{k^2-k_0^2}\,\mathrm{d}k=\frac{\pi i}{k_0}h_n^{(1)}(k_0r')j_n(k_0r)-\frac{\pi}{k_0^2(2n+1)}\frac{r^n}{(r')^{n+1}}, \tag{184}$$

and the target integral in Eq. (180) becomes

$$\int_{0}^{\infty}\frac{j_n(kr')j_n(kr)}{k^2-k_0^2}\,\mathrm{d}k=\frac{\pi i}{2k_0}h_n^{(1)}(k_0r')j_n(k_0r)-\frac{\pi}{2k_0^2(2n+1)}\frac{r^n}{(r')^{n+1}}. \tag{185}$$

G. Limit of Vector Spherical Functions

From the definition, the vector spherical functions read

$$\begin{aligned}
\underline{\mathbf{M}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) &= \frac{1}{\sqrt{n(n+1)f_{nm}}} j_n(k_i r) e^{im\phi}\left[i\pi_{nm}(\theta)\hat{\theta} - \tau_{nm}(\theta)\hat{\phi}\right], \\
\underline{\mathbf{N}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) &= \frac{1}{\sqrt{n(n+1)f_{nm}}} \frac{j_n(k_i r)}{k_i r} \cdot n(n+1) P_n^m(\cos\theta) e^{im\phi}\hat{r} \\
&\quad + \frac{1}{\sqrt{n(n+1)f_{nm}}} \frac{1}{k_i r} \frac{\mathrm{d}\psi_n(k_i r)}{\mathrm{d}(k_i r)} \cdot e^{im\phi}\left[\tau_{nm}(\theta)\hat{\theta} + i\pi_{nm}(\theta)\hat{\phi}\right].
\end{aligned} \tag{186}$$

We would like to evaluate the behavior of $\underline{\mathbf{M}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$ and $\underline{\mathbf{M}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi)$ when $\mathbf{r}\to 0$,

$$\lim_{\mathbf{r}\to 0} \underline{\mathbf{M}}_{nm}^{(\mathrm{I})}(\rho,\theta,\phi) = \frac{1}{\sqrt{n(n+1)f_{nm}}} \lim_{r\to 0^+} j_n(kr) \cdot \lim_{\theta\to 0}\left[\tau_{nm}(\theta)\hat{\theta} + i\pi_{nm}(\theta)\hat{\phi}\right],$$

To evaluate the limit of the radial part, we express the spherical Bessel function by its asymptotic form,

$$\lim_{r\to 0^+} j_n(k_i r) = \lim_{r\to 0^+} \frac{(kr)^n}{(2n+1)!!} = 0, \qquad n \neq 0, \tag{187}$$

which can be found in the NIST handbook of mathematical functions.[8] It is obvious to obtain the limit has the only no agreed-upon (nonzero?) value when $n = 0$. Moreover, it can be shown that

$$\lim_{\theta\to 0} \pi_{00}(\theta) = \lim_{\theta\to 0} \frac{0}{\sin\theta} = \lim_{\theta\to 0} \frac{0}{\cos\theta} = 0, \tag{188}$$

$$\lim_{\theta\to 0} \tau_{00}(\theta) = \lim_{\theta\to 0} \frac{\mathrm{d}}{\mathrm{d}\theta} P_0^0(\cos\theta) = 0, \tag{189}$$

where we use L'Hôpital's rule when evaluating π_{00}. Thus, we have

$$\lim_{\mathbf{r}\to 0} \underline{\mathbf{M}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) = 0, \qquad \forall\ n \in \mathbb{N}_0,\ |m| \leq n, \tag{190}$$

where $\mathbb{N}_0$ denotes the set of all natural numbers including zero. Also, for $\underline{\mathbf{N}}_{nm}^{(\mathrm{I})}(\rho,\theta,\phi)$,

$$\begin{aligned}\lim_{\mathbf{r}\to 0}\underline{\mathbf{N}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) =& \frac{1}{\sqrt{n(n+1)f_{nm}}} n(n+1)\cdot\lim_{r\to 0^+}\frac{j_n(k_i r)}{k_i r}\cdot\lim_{\theta\to 0}P_n^m(\cos\theta)\hat{r}\\ &+\frac{1}{\sqrt{n(n+1)f_{nm}}}\cdot\lim_{r\to 0^+}\frac{1}{k_i r}\frac{\mathrm{d}\psi_n(k_i r)}{\mathrm{d}(k_i r)}\cdot\lim_{\theta\to 0}\left[\tau_{nm}(\theta)\hat{\theta}+i\pi_{nm}(\theta)\hat{\phi}\right]. \quad (191)\end{aligned}$$

In the same way, we evaluate the limits via the asymptotic form,

$$\lim_{r\to 0^+}\frac{j_n(k_i r)}{k_i r}=\lim_{r\to 0^+}\frac{(k_i r)^{n-1}}{(2n+1)!!}=\frac{1}{3},\qquad n=1, \quad (192)$$

and

$$\lim_{r\to 0^+}\frac{1}{k_i r}\frac{\mathrm{d}}{\mathrm{d}(k_i r)}\left[k_i r\cdot j_n(k_i r)\right]=\lim_{r\to 0^+}\left[\frac{j_n(k_i r)}{k_i r}+j_n'(k_i r)\right]=\frac{2}{3},\qquad n=1.$$

It is worth mentioning that we define $\lim_{r\to 0^+} r^0\equiv 1$. Finally, we obtain that

$$\lim_{\mathbf{r}\to 0}\underline{\mathbf{N}}_{nm}^{(\mathrm{I})}(k_i r,\theta,\phi) \quad (193)$$

$$=\begin{cases}\dfrac{1}{\sqrt{n(n+1)f_{nm}}}\dfrac{2}{3}\left[P_n^m(1)\hat{r}+\tau_{nm}(0)\hat{\theta}+i\pi_{nm}(0)\hat{\phi}\right] & n=1,|m|\le n\\ 0 & n>1,|m|\le n\end{cases}. \quad (194)$$

H. Derivation of Eqs. (119b) and (119c)

Finally, we provide the derivation of Eqs. (119b) and (119c). First, by definition, $\underline{\pi}_{nm}(\theta)$ can be written by

$$\begin{aligned}\underline{\tau}_{nm}(\theta) &= \left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2} \frac{\mathrm{d}}{\mathrm{d}\theta}\left[D^n_{m0}(0,\theta,0)\right] = \left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2} \frac{\mathrm{d}}{\mathrm{d}\theta}\left\langle n,m\right| e^{-i\theta\hat{J}_y}\left|n,0\right\rangle \\ &= \left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2} (-i)\left\langle n,m\right| e^{-i\theta\hat{J}_y}\hat{J}_y\left|n,0\right\rangle . \end{aligned} \tag{195}$$

Using ladder operators to describe $\hat{J}_y$, the equation becomes

$$\begin{aligned}\underline{\tau}_{nm}(\theta) &= -\frac{1}{2}\left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2}\left[\left\langle n,m\right| e^{-i\theta\hat{J}_y}\hat{J}_+\left|n,0\right\rangle - \left\langle n,m\right| e^{-i\theta\hat{J}_y}\hat{J}_-\left|n,0\right\rangle\right] \\ &= -\frac{1}{2}\left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2}\left[C_+(n,0)d^n_{m,1}(\theta) - C_-(n,0)d^n_{m,-1}(\theta)\right] \\ &= -\left[\frac{2n+1}{8}\right]^{1/2}\left[D^n_{m,1}(0,\theta,0) - D^n_{m,-1}(0,\theta,0)\right]. \end{aligned} \tag{196}$$

On the other hand, the derivation of $\underline{\pi}_{nm}(\theta)$ is a little bit complicated. By definition, we can rewrite $\underline{\pi}_{nm}(\theta)$ to

$$\begin{aligned}\underline{\pi}_{nm}(\theta) &= \left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2} \frac{m}{\sin\theta} d^n_{m,0}(\theta) = \left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2} \frac{1}{\sin\theta}\left\langle n,m\right| \hat{J}_z e^{-i\theta\hat{J}_y}\left|n,0\right\rangle \\ &= \left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2} \frac{1}{\sin\theta}\left\langle n,m\right| e^{-i\theta\hat{J}_y}e^{i\theta\hat{J}_y}\hat{J}_z e^{-i\theta\hat{J}_y}\left|n,0\right\rangle . \end{aligned} \tag{197}$$

Here, we need to prove that $e^{i\theta\hat{J}_y}\hat{J}_z e^{-i\theta\hat{J}_y} = \cos\theta\hat{J}_z - \sin\theta(\hat{J}_+ + \hat{J}_-)/2$ first. From the lemma in Baker-Campbell-Hausdorff formula, $e^{\hat{X}}\hat{Y}e^{-\hat{X}} = \hat{Y} + [\hat{X},\hat{Y}] + [\hat{X},[\hat{X},\hat{Y}]]/2! +$

$[\hat{X},[\hat{X},[\hat{X},\hat{Y}]]/3!+\cdots$, it becomes

$$\begin{aligned} e^{i\theta\hat{J}_y}\hat{J}_z e^{-i\theta\hat{J}_y} &= \hat{J}_z + [i\theta\hat{J}_y,\hat{J}_z] + [i\theta\hat{J}_y,[i\theta\hat{J}_y,\hat{J}_z]]/2! + [i\theta\hat{J}_y,[i\theta\hat{J}_y,[i\theta\hat{J}_y,\hat{J}_z]]]/3! + \cdots \\ &= \hat{J}_z + (i\theta)i\hat{J}_x + \frac{(i\theta)^2}{2!}\hat{J}_z + \frac{(i\theta)^3}{3!}i\hat{J}_x + \cdots \\ &= \left[\sum_{n=0}^{\infty}\frac{(i\theta)^{2n}}{(2n)!}\right]\hat{J}_z + \left[\sum_{n=0}^{\infty}\frac{(i\theta)^{2n+1}}{(2n+1)!}\right]i\hat{J}_x = \cosh i\theta\hat{J}_z + i\sinh i\theta\hat{J}_x \\ &= \cos\theta\hat{J}_z - \sin\theta(\hat{J}_+ + \hat{J}_-)/2 \end{aligned} \tag{198}$$

Therefore, Eq. (197) becomes

$$\underline{\pi}_{nm}(\theta) = -\frac{1}{2}\left[\frac{(2n+1)}{2n(n+1)}\right]^{1/2}\langle n,m|\, e^{-i\theta\hat{J}_y}(\hat{J}_+ + \hat{J}_-)\,|n,0\rangle \tag{199}$$

$$= -\left[\frac{2n+1}{8}\right]^{1/2}\left[D^n_{m,1}(0,\theta,0) + D^n_{m,-1}(0,\theta,0)\right] \tag{200}$$

I. A Simple Proof of the Lemma in the BCH Formula

In this subsection, we try to give a simple proof of the lemma in the BCH formula,

$$e^{\hat{X}}\hat{Y}e^{-\hat{X}} = \hat{Y} + [\hat{X},\hat{Y}] + [\hat{X},[\hat{X},\hat{Y}]]/2! + [\hat{X},[\hat{X},[\hat{X},\hat{Y}]]/3! + \cdots . \tag{201}$$

First, we express the exponential functions by Taylor series,

$$e^{\hat{X}}\hat{Y}e^{-\hat{X}} = \left(1 + \hat{X} + \frac{\hat{X}^2}{2!} + \frac{\hat{X}^3}{3!} + \cdots\right)\hat{Y}\left(1 - \hat{X} + \frac{\hat{X}^2}{2!} - \frac{\hat{X}^3}{3!} + \cdots\right). \tag{202}$$

By collecting the same powers of $\hat{X}$, the equation can be rewritten as

$$\begin{aligned} e^{\hat{X}}\hat{Y}e^{-\hat{X}} &= \sum_{n=0}^{\infty}\left\{\sum_{m=0}^{n}\left[\frac{\hat{X}^m}{m!}\hat{Y}\frac{(-\hat{X})^{n-m}}{(n-m)!}\right]\right\} \\ &= \sum_{n=0}^{\infty}\left\{\frac{1}{n!}\sum_{m=0}^{n}\left[\frac{n!}{m!(n-m)!}(-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m}\right]\right\} \\ &= \sum_{n=0}^{\infty}\left\{\frac{1}{n!}\sum_{m=0}^{n}\left[C_m^n(-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m}\right]\right\} \end{aligned} \tag{203}$$

To simplify the result, we need to use the recurrence relation (will be proved after),

$$\sum_{m=0}^{n} C_m^n(-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} = \sum_{m=0}^{n-1} C_m^{n-1}(-1)^{n-m-1}\hat{X}^m\left[\hat{X},\hat{Y}\right]\hat{X}^{n-m-1}, \tag{204}$$

which indicates that

$$\begin{aligned} \sum_{m=0}^{n} C_m^n(-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} &= \sum_{m=0}^{n-1} C_m^{n-1}(-1)^{n-m-1}\hat{X}^m\left[\hat{X},\hat{Y}\right]\hat{X}^{n-m-1} \\ &= \sum_{m=0}^{n-2} C_m^{n-2}(-1)^{n-m-2}\hat{X}^m\left[\hat{X},\left[\hat{X},\hat{Y}\right]\right]\hat{X}^{n-m-2} \\ &\vdots \\ &= \underbrace{\left[\hat{X},\cdots\left[\hat{X},\left[\hat{X},\hat{Y}\right]\right]\cdots\right]}_{n}. \end{aligned} \tag{205}$$

Therefore, Eq. (203) becomes

$$e^{\hat{X}}\hat{Y}e^{-\hat{X}} = \sum_{n=0}^{\infty}\left\{\frac{1}{n!}\underbrace{\left[\hat{X},\cdots\left[\hat{X},\left[\hat{X},\hat{Y}\right]\right]\cdots\right]}_{n}\right\} \tag{206}$$

$$= \hat{Y} + [\hat{X},\hat{Y}] + \frac{1}{2!}[\hat{X},[\hat{X},\hat{Y}]] + \frac{1}{3!}[\hat{X},[\hat{X},[\hat{X},\hat{Y}]] + \cdots \tag{207}$$

Postscript: the derivation of the recurrence relation is as follows,

$$\sum_{m=0}^{n} C_m^n (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m}$$

$$= (-1)^n\hat{Y}\hat{X}^n + \sum_{m=1}^{n-1} C_m^n (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} + \hat{X}^n\hat{Y} \tag{208}$$

$$= (-1)^n\hat{Y}\hat{X}^n + \sum_{m=1}^{n-1} C_{m-1}^{n-1} (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} \tag{209}$$

$$+ \sum_{m=1}^{n-1} C_m^{n-1} (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} + \hat{X}^n\hat{Y}$$

$$= (-1)^n\hat{Y}\hat{X}^n + (-1)^{n-1}\hat{X}\hat{Y}\hat{X}^{n-1} + \sum_{m=2}^{n-1} C_{m-1}^{n-1} (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} \tag{210}$$

$$+ \sum_{m=1}^{n-2} C_m^{n-1} (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} - \hat{X}^{n-1}\hat{Y}\hat{X} + \hat{X}^n\hat{Y}$$

$$= (-1)^{n-1}\left[\hat{X},\hat{Y}\right]\hat{X}^{n-1} + \sum_{m'=1}^{n-2} C_{m'}^{n-1} (-1)^{n-m'-1}\hat{X}^{m'+1}\hat{Y}\hat{X}^{n-m'-1} \tag{211}$$

$$+ \sum_{m=1}^{n-2} C_m^{n-1} (-1)^{n-m}\hat{X}^m\hat{Y}\hat{X}^{n-m} + \hat{X}^{n-1}\left[\hat{X},\hat{Y}\right]$$

$$= (-1)^{n-1}\left[\hat{X},\hat{Y}\right]\hat{X}^{n-1} + \sum_{m=1}^{n-2} C_m^{n-1} (-1)^{n-m-1}\hat{X}^m\left[\hat{X},\hat{Y}\right]\hat{X}^{n-m-1} + \hat{X}^{n-1}\left[\hat{X},\hat{Y}\right] \tag{212}$$

$$= \sum_{m=0}^{n-1} C_m^{n-1} (-1)^{n-m-1}\hat{X}^m\left[\hat{X},\hat{Y}\right]\hat{X}^{n-m-1} \tag{213}$$

In Eq. (211), we let $m' = m-1$, then the range of summation becomes $m' = 1$ to $m' = n-1$.

J. Source Code of Other Modules

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                                     %
4  % ! Settings -> struct array     : Calculation parameters                    %
5  % !  .nmax    -> double           : Maximum expansion order                  %
6  % !  .nr      -> double (1xp)**   : Relative refractive index               %
7  % !  .k0      -> double           : modulus of wavevector in vacuum          %
8  % !  .k0s     -> double [1x(p-1)] : k0*r_i (r_i = radius of boundary)        %
9  % !  .BC      -> string           : Boundary condition                       %
10 % !  .DPos    -> double (3x1)     : Donor position                           %
11 % !   .Sph    -> double (3x1)     : Presented in a spherical coordinate      %
12 % !   .Cart   -> double (3x1)     : Presented in a Cartesian coordinate      %
13 % !  .APos    -> double (3x1)     : Acceptor position                        %
14 % !   .Sph    -> double (3x1)     : Presented in a spherical coordinate      %
15 % !   .Sph2   -> double (3x1)     : Presented in S2 coordinate               %
16 % !   .Cart   -> double (3x1)     : Presented in a Cartesian coordinate      %
17 % !  .DOri    -> double (3x1)     : Orientation of the Donor dipole          %
18 % !   .Cart   -> double (3x1)     : Presented in a Cartesian coordinate      %
19 %    .DRad    -> struct array     : Set of radial functions (donor)          %
20 %     .h1     -> double (1xn)     : Spherical Bessel functions               %
21 %     .raddxi-> double (1xn)      : dxi/kr (See more in SphBessel)           %
22 %    .ARad    -> struct array     : Set of radial functions (acceptor)       %
23 %     .j1     -> double (1xn)     : Spherical Bessel functions               %
24 %     .raddpsi> double (1xn)      : dxi/kr (See more in SphBessel)           %
25 %    .DNAng   -> struct array     : Normalized Tau, Pi, and P (donor)        %
26 %     .NTau   -> double [nx(2n+1)]: Normalized Tau array                     %
27 %     .NPi    -> double [nx(2n+1)]: Normalized Pi array                      %
28 %     .NP     -> double [nx(2n+1)]: Normalized P array                       %
29 %    .ANAng   -> struct array     : Normalized Tau, Pi, and P (acceptor)     %
30 %     .NTau   -> double [nx(2n+1)]: Normalized Tau array                     %
31 %     .NPi    -> double [nx(2n+1)]: Normalized Pi array                      %
32 %     .NP     -> double [nx(2n+1)]: Normalized P array                       %
33 %    .emphi   -> double [nx(2n+1)]: Normalized azimuthal functions           %
34 %    .Source  -> struct array     : Source Expansion coefficients            %
35 %     .p      -> double [nx(2n+1)]: Coefficient p                            %
36 %     .q      -> double [nx(2n+1)]: Coefficient q                            %
37 %    .Scat    -> struct array     : Mie coefficients                         %
38 %     .gamma  -> double (1xn)     : Mie coefficient gamma                    %
39 %     .delta  -> double (1xn)     : Mie coefficient delta                    %
40 %    .EdipS1  -> double (3x1)     : EdipField in S1 coordinate               %
41 %    .EdipS2  -> double (3x1)     : EdipField in S2 coordinate               %
42 % Outputs:                                                                   %
43 %   Output    -> struct array     : Storage of output data                   %
44 %    .Etot    -> double (3x1)     : Total electric field at APos             %
45 %    .Int     -> double           : Electric field intensity at APos         %
46 %    .Edip    -> double (3x1)     : Electric dipole field at APos            %
47 %    .NEtot   -> double (3x1)     : Normalized Etot at APos (Etot/Edip)      %
48 % Temporary data:                                                            %
49 %   Temp      -> struct array     : Storage of temporary data                %
50 %    .AVSF    -> struct array     : Vector spherical function (acceptor)     %
51 %     .M      -> double [nx(2n+1)]: Vector spherical function M              %
```

```
 %     .N     -> double [nx(2n+1)]: Vector spherical function N          %
 %    .Layer1M-> double (3x1)     : Electric field in layer 1 (M part)  %
 %    .Layer1N-> double (3x1)     : Electric field in layer 1 (N part)  %
 %                                                                      %
 % !: Variables with "!" is required to make the module work.          %
 % **:                                                                 %
 %  The value of p depends on the spherical scatterer:                 %
 %   Single sphere     -> p = 2                                        %
 %   Core/shell sphere -> p = 3                                        %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Output = EFieldR1(Settings)
    % Variables
    nmax = Settings.nmax;
    rhoD = Settings.nr(1)*Settings.k0*Settings.DPos.Sph(1);
    rhoA = Settings.nr(2)*Settings.k0*Settings.APos.Sph(1);
    % Radial Functions
    if isfield(Settings,'DRad')  == 0
        Settings.DRad = SphBessel(rhoD,nmax,1,'hankel1');
    end
    if isfield(Settings,'ARad') == 0
            Settings.ARad = SphBessel(rhoA,nmax,1,'bessel');
    else
        if isfield(Settings.ARad,'j1') == 0
            Settings.ARad = SphBessel(rhoA,nmax,1,'bessel');
        end
    end
    % Angular Functions
    if isfield(Settings,'DNAng') == 0
        Settings.DNAng = NormTauPiP(nmax,Settings.DPos.Sph(2),'reversed');
    end
    if isfield(Settings,'ANAng') == 0
        Settings.ANAng = NormTauPiP(nmax,Settings.APos.Sph(2),'normal');
    end
    % Azimuthal Functions
    if isfield(Settings,'emphi') == 0
        if Settings.APos.Sph(3) == 0
            % For Speed-Up
            emphi = sqrt(1/2/pi);
        else
            % Setting exp(-inf) = 0 for Useless Array Elements
            m = -inf*ones(nmax,2*nmax+1);
            for ii = 1:nmax
                    m(ii,1:2*ii+1) = -ii:1:ii;
            end
            emphi = sqrt(1/2/pi)*exp(1i*m*Settings.APos.Sph(3));
            % Change exp(-inf) = NaN to Zero
            emphi(isnan(emphi)) = 0;
        end
    end
    % Mie Coefficients
    if isfield(Settings,'Source') == 0
        Settings.Source = SourCoeff(Settings,'dipole');
    end
```

```
    if isfield(Settings,'layer1') == 0
        if strcmp(Settings.BC,'sphere') == 1
            Settings.Layer1 = MieSingle(Settings.nr,Settings.k0s,nmax);
        elseif strcmp(Settings.BC,'coreshell') == 1
            % Alert of the Unsupported Function
            Output.error1 = ...
                'EFieldR1 for coreshell structures is not supported yet.';
            Settings.Layer1.gamma = 0;
            Settings.Layer1.delta = 0;
        end
        Settings.Layer1.d = ...
            Settings.Source.p.*transpose(Settings.Layer1.delta);
        Settings.Layer1.c = ...
            Settings.Source.q.*transpose(Settings.Layer1.gamma);
    end
    % Generating M and N Fields
    Temp.AVSF = VectSphFunc(rhoA,nmax,Settings.ARad,Settings.ANAng,emphi);
    % Donor Dipole Field
    if isfield(Settings,'EdipS2') == 0
        if isfield(Settings.APos,'Sph2') == 0
            Settings.APos.Sph2 = ...
                C2S(Settings.APos.Cart-Settings.DPos.Cart);
        end
        % Field in the Secondary Coordinate
        Settings.EdipS2 = EdipField(Settings.nr(1),Settings.k0,...
            Settings.APos.Sph2,Settings.DOri.Cart);
        % Transforming to the Primary Coordinate
        Settings.EdipS1 = S2S(Settings.EdipS2,...
            (Settings.APos.Sph2(2)-Settings.APos.Sph(2)),0);
    end
    % Summing All Order of the Field in Layer 1
    Temp.Layer1M = ...
        reshape(sum(Temp.AVSF.M.*Settings.Layer1.c,[1,2]),[3,1]);
    Temp.Layer1N = ...
        reshape(sum(Temp.AVSF.N.*Settings.Layer1.d,[1,2]),[3,1]);
    % Total Electric Field at the Acceptor Position
    Output.Etot = Temp.Layer1M + Temp.Layer1N;
    % Total intensity at the Acceptor position
    Output.Int = norm(Temp.Layer1M + Temp.Layer1N).^2;
    % Dipole field
    Output.Edip =  Settings.EdipS1;
    % Etot / Edip
    Output.NEtot = Output.Etot./Settings.EdipS1;
    % Other Additional Outputs
        % Note: Feel Free to Add What You Want!
        % Use the Structure Array to Output Data
        % Example: Output.testAPos = Settings.APos.Sph2;
end
```

Function 18: Module of calculating electric fields in region 1

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Inputs:                                                                      %
4  % ! Settings -> struct array     : Calculation parameters                     %
5  % !  .nmax    -> double           : Maximum expansion order                  %
6  % !  .nr      -> double (1xp)**   : Relative refractive index                %
7  % !  .k0      -> double           : modulus of wavevector in vacuum          %
8  % !  .k0s     -> double [1x(p-1)] : k0*r_i (r_i = radius of boundary)        %
9  % !  .BC      -> string           : Boundary condition                       %
10 % !  .DPos    -> double (3x1)     : Donor position                           %
11 % !    .Sph   -> double (3x1)     : Presented in a spherical coordinate      %
12 % !    .Cart  -> double (3x1)     : Presented in a Cartesian coordinate      %
13 % !  .DOri    -> double (3x1)     : Orientation of the Donor dipole          %
14 % !    .Cart  -> double (3x1)     : Presented in a Cartesian coordinate      %
15 %    .DRad    -> struct array     : Set of radial functions (donor)          %
16 %     .h1     -> double (1xn)     : Spherical Bessel functions               %
17 %     .raddxi -> double (1xn)     : dxi/kr (See more in SphBessel)           %
18 %    .DNAng   -> struct array     : Normalized Tau, Pi, and P (reversed)     %
19 %     .NTau   -> double [nx(2n+1)]: Normalized Tau array                     %
20 %     .NPi    -> double [nx(2n+1)]: Normalized Pi array                      %
21 %     .NP     -> double [nx(2n+1)]: Normalized P array                       %
22 %    .DNAngN  -> struct array     : Normalized Tau, Pi, and P (normal)       %
23 %     .NTau   -> double [nx(2n+1)]: Normalized Tau array                     %
24 %     .NPi    -> double [nx(2n+1)]: Normalized Pi array                      %
25 %     .NP     -> double [nx(2n+1)]: Normalized P array                       %
26 %    .emphi   -> double [nx(2n+1)]: Normalized azimuthal functions           %
27 %    .Source  -> struct array     : Source Expansion coefficients            %
28 %     .p      -> double [nx(2n+1)]: Coefficient p                            %
29 %     .q      -> double [nx(2n+1)]: Coefficient q                            %
30 %    .Scat    -> struct array     : Mie coefficients                         %
31 %     .alpha  -> double (1xn)     : Mie coefficient alpha                    %
32 %     .beta   -> double (1xn)     : Mie coefficient beta                     %
33 % Outputs:                                                                   %
34 %   Output    -> struct array     : Storage of output data                   %
35 %    .Purcell -> double           : Purcell factor at DPos                   %
36 % Temporary data:                                                           %
37 %   Temp      -> struct array     : Storage of temporary data                %
38 %    .DVSF    -> struct array     : Vector spherical function (acceptor)     %
39 %     .M      -> double [nx(2n+1)]: Vector spherical function M              %
40 %     .N      -> double [nx(2n+1)]: Vector spherical function N              %
41 %    .ScatM   -> double (3x1)     : Scattering contribution (M part)         %
42 %    .ScatN   -> double (3x1)     : Scattering contribution (N part)         %
43 %                                                                            %
44 % !: Variables with "!" is required to make the module work.                 %
45 % **:                                                                        %
46 %  The value of p depends on the spherical scatterer:                        %
47 %   Single sphere     -> p = 2                                               %
48 %   Core/shell sphere -> p = 3                                               %
49 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50
51 function Output = PurcellR0(Settings)
52     % Variables
53     nmax = Settings.nmax;
```

```
    rhoD = Settings.nr(1)*Settings.k0*Settings.DPos.Sph(1);
    % Radial Functions for Donor
    if isfield(Settings,'DRad')  == 0
        Settings.DRad = SphBessel(rhoD,nmax,1,'hankel1');
    end
    % Angular Functions
    if isfield(Settings,'DNAng') == 0
        Settings.DNAng = NormTauPiP(nmax,Settings.DPos.Sph(2),'reversed');
    end
    if isfield(Settings,'DNAngN') == 0
        Settings.DNAngN = NormTauPiP(nmax,Settings.DPos.Sph(2),'normal');
    end
    % Mie Coefficients
    if isfield(Settings,'Source') == 0
        Settings.Source = SourCoeff(Settings,"Green's function only");
    end
    if isfield(Settings,'Scat') == 0
        if strcmp(Settings.BC,'sphere') == 1
            Settings.Scat = MieSingle(Settings.nr,Settings.k0s,nmax);
        elseif strcmp(Settings.BC,'coreshell') == 1
            Settings.Scat = MieCoreShell(Settings.nr,Settings.k0s,nmax);
        end
        Settings.Scat.a = ...
            Settings.Source.p.*transpose(Settings.Scat.alpha);
        Settings.Scat.b = ...
            Settings.Source.q.*transpose(Settings.Scat.beta);
    end
    % Azimuthal Functions
    if isfield(Settings,'emphi') == 0
        emphi = sqrt(1/2/pi);
    end
    % Generating M and N Fields
    Temp.DVSF=VectSphFunc(rhoD,nmax,Settings.DRad,Settings.DNAngN,emphi);
    % Summing All order of the Scattering Field
    Temp.ScatM = reshape(sum(Temp.DVSF.M.*Settings.Scat.b,[1,2]),[3,1]);
    Temp.ScatN = reshape(sum(Temp.DVSF.N.*Settings.Scat.a,[1,2]),[3,1]);
    % Scattering Part at the Donor Position
    Output.EScat = Temp.ScatM + Temp.ScatN;
    % Purcell Factor
    Output.Purcell = 1 + 6*pi/Settings.k0*...
        imag(transpose(Output.EScat)*Settings.DOri.Sph);
end
```

Function 19: Module of calculating Purcell factor in region 0

K. Auxiliary Functions Used in `main.m`

```
function result = TenCont(A,B,dim)
    % Matrix Size
    sizeA = size(A);
    sizeB = size(B);
    % Error of dimension
    if size(dim) > 2
        disp('Wrong Assignment of Dimension in TenCont');
    end
    % Size of the Target Column
    sizetar = max(sizeB);
    % Alert of Illegal Operation
    if sum(sizeB) > sizetar+1
        dispstat(sprintf('Illegal Tensor Contraction'),'keepthis',...
            'timestamp');
    end
    % Erasing the Contribution of the Target Column
    sizeA(dim(1)) = [];
    % Reshaping Matrix and Contraction
    result = reshape(reshape(A,[prod(sizeA),sizetar])*B,sizeA);
end
```

Function 20: Function `TenCont`

```
function rsph = C2S(rcart)
    % Assigning the Cartesian Components
    x = rcart(1,:); y = rcart(2,:); z = rcart(3,:);
    % Radial Distance
    r = sqrt(x.^2+y.^2+z.^2);
    % Polar Angle
    theta = acos(z./r);
    % Assigning Polar Angle when r = 0
    theta(not(any(r,1))) = 0;
    % Azimuthal Angle
    phi = atan(y./x);
    % Assigning Azimuthal Angle when x = 0 and y = 0
    phi(and(not(any(y,1)),not(any(x,1)))) = 0;
    % Assigning Azimuthal Angle when x < 0
    phi(and(x<0,any(y,1))) = pi;
    % Column Form
    rsph = [r;theta;phi];
end
```

Function 21: Function `C2S`

```
function MyPlot(fplot,resize,multi)
    if multi == 0
        % Create a Window at a Decided Position
        figure('Color','white','Position',[150,70,(150+950*resize),...
            (70+830*resize)]);
        % Create an Frame for a Figure
        axes('OuterPosition',[0.005,0.01,.99,.97]);
    elseif multi == 1
        clear sub;
        % Open the Overwrite Mode
        hold on
    end
    % Create a Specific Plot
    plot(fplot.x,fplot.y,fplot.colorstyle,'linewidth',2);
    % Set log Scale
    if isfield(fplot,'yscale') == 1
        set(gca,'YScale', fplot.yscale);
    end
    % Box Setting
    box on
    % Box Linewidth
    set(gca,'linewidth',2);
    % Grid Setting
    grid off
    % Axis Range
    axis(fplot.range);
    % Font Size
    set(gca,'fontsize',30*resize,'fontname','Times New Roman');
    % Label of x Axis
    xlabel(fplot.xlabel,'interpreter','latex');
    % Label of y Axis
    ylabel(fplot.ylabel,'interpreter','latex');
    % Figure Title
    if isfield(fplot,'title') == 1
        title(fplot.title);
    end
end
```

Function 22: Function `MyPlot`

```
function dispstat(TXT,varargin)
% Prints overwritable message to the command line. If you don't want to
% keep this message, call dispstat function with option 'keepthis'. If you
% want to keep the previous message, use option 'keepprev'. First argument
% must be the message.
% IMPORTANT! In the first call, option 'init' must be used for
% initialization purposes.
% Options:
%   'init'      this must be called in the begining. Otherwise, it can
%               overwrite the previous outputs on the command line.
%   'keepthis'  the message will be persistent, wont be overwritable.
%   'keepprev'  the previous message wont be overwritten.
%               New message will start from next line.
%   'timestamp' current time hh:mm:ss will be appended to the beginning of
%               the message.
% Example:
%   clc;
%   fprintf('12345677890\n');
%   dispstat('','init')      %Initialization. Does not print anything.
%   dispstat('Time stamp will be written over this text.'); % First output
%   dispstat('is current time.','timestamp','keepthis'); % Overwrites the
%       previous output but this output wont be overwritten.
%   dispstat(sprintf('*********\nDeveloped by %s\n*********','Kasim')); %
%       does not overwrites the previous output.
%   dispstat('','timestamp','keepprev','keepthis'); % does not overwrites
%       the previous output
%   dispstat('this wont be overwriten','keepthis');
%   dispstat('dummy dummy dummy');
%   dispstat('final stat');
% % Output:
%     12345677890
%     15:15:34 is current time.
%     *********
%     Developed by Kasim
%     *********
%     15:15:34
%     this wont be overwriten
%     final stat
% **********
% **** Options
keepthis = 0; % option for not overwriting
keepprev = 0;
timestamp = 0; % time stamp option
init = 0; % is it initialization step?
if ~ischar(TXT)
    return
end
persistent prevCharCnt;
if isempty(prevCharCnt)
    prevCharCnt = 0;
end
if nargin == 0
    return
```

```
elseif nargin > 1
    for i = 2:nargin
        eval([varargin{i-1} '=1;']);
    end
end
if init == 1
    prevCharCnt = 0;
    return;
end
if isempty(TXT) && timestamp == 0
    return
end
if timestamp == 1
    c = clock; % [year month day hour minute seconds]
    txtTimeStamp = sprintf('%02d:%02d:%02d ',c(4),c(5),round(c(6)));
else
    txtTimeStamp = '';
end
if keepprev == 1
    prevCharCnt = 0;
end
% *************** Make safe for fprintf, replace control charachters
TXT = strrep(TXT,'%','%%');
TXT = strrep(TXT,'\','\\');
% *************** Print
TXT = [txtTimeStamp TXT '\n'];
fprintf([repmat('\b',1, prevCharCnt) TXT]);
nof_extra = length(strfind(TXT,'%%'));
nof_extra = nof_extra + length(strfind(TXT,'\\'));
nof_extra = nof_extra + length(strfind(TXT,'\n'));
prevCharCnt = length(TXT) - nof_extra; %-1 is for \n
if keepthis == 1
    prevCharCnt = 0;
end
end
```

Function 23: Function `dispstat`[9]

```
function vfin = VecTrans(vini,solidangle,type)
    % Solid Angle to Polar Angle and Azimuthal Angle
    theta = solidangle(1); phi = solidangle(2);
    % Calculating cos(theta) and sin(theta)
    if theta == 0
        sint = 0;
        cost = 1;
    elseif theta == pi
        sint=0;
        cost = -1;
    elseif theta == pi/2 || theta ==3*pi/2
        cost = 0;
        sint = 1;
    else
        cost = cos(theta);
        sint = sin(theta);
    end
    % Calculating cos(phi) and sin(phi)
    if phi == 0
        sinp = 0;
        cosp = 1;
    elseif phi == pi/2
        cosp = 0;
        sinp = 1;
    else
        cosp = cos(phi);
        sinp = sin(phi);
    end
    % Transform Matrix
    T = [ sint*cosp       cost*cosp     -sinp;
            sint*sinp       cost*sinp       cosp;
            cost             -sint              0       ];
    % Asigning the Type of Transformation
    if strcmp(type,'C2S') == 1
        T = transpose(T);
    elseif strcmp(type,'S2C') == 0
        disp('Error from function "VecTrans"');
    end
    vfin = T*vini;
end
```

Function 24: Function `VecTrans`

L. Setting Files of Other Calculation Modes

```
%% Input for wavelength mode (feel free to change variables)
% Dipole position (micrometer)
    % Donor dipole
    Dx = 0; %Do not change the value! Extremely Important!
    Dy = 0; %Do not change the value! Extremely Important!
    Dz = 0.100;
    % Acceptor dipole
    unit = 0.001; % micrometer
    points = linspace(-120,120,481)*unit;
    [Ax,Az] = meshgrid(points,points);
    Ay = 0;
% Dipole orientation
    % Donor dipole (Cartesian coordinate, [x;y;z])
    Doc = [1;0;0];
    % Acceptor dipole (Cartesian coordinate, [x;y;z])
    Aoc = [0;0;1];
% Largest expansion order (n) in a calculation
nmax = 70;
% Boundary conditions ('sphere' or 'coreshell')
BC = 'sphere';
    % Calling dielectric data
    excelre = ...
       xlsread('.\DielectricFunction\dielectric function.xlsx','Ag_JPCL');
    epsilonmat = excelre(:,2) + 1i*excelre(:,3);
    % Wavelength (microns)
    lambda = excelre(:,1)*1e-3;
    %Wavenumber (microns)
    k0 = 2*pi./lambda;
    % Desired wavelength (microns)
    wavelength = 0.353;
    % Setting conditions
    if strcmp(BC,'sphere') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),2);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = sqrt(epsilonmat);
        % Radius of the sphere (unit: micron)
        rbc = 0.085;
        % Dimensionless radial variable
        k0s = k0*rbc;
    elseif strcmp(BC,'coreshell') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),3);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = 2.0   ;
        % Relative refractive index (Region 2)
        nr(:,3) = sqrt(epsilonmat);
```

```
        % Radius of the core and the shell [shell, core] (unit: micron)
        rbc = [0.070, 0.060];
        % Dimensionless radial variable
        k0s = k0*rbc;
    end

%% ------- Do not change the following settings ------- %%
% Creating a mode structure array
    % Mode name
    Settings.ModeName = 'mapping';
    % Donor position
        % Cartesian coordinate
        Settings.DPos.Cart = [Dx;Dy;Dz];
    % Acceptor position
        % Cartesian coordinate
        AxReshape = reshape(Ax,[1,numel(Ax)]);
        AyReshape = Ay*ones([1,numel(Ax)]);
        AzReshape = reshape(Az,[1,numel(Az)]);
        Settings.APos.Cart = [AxReshape;AyReshape;AzReshape];
    % Donor orientation
        % Cartesian coordinate
        Settings.DOri.Cart = Doc;
    % Acceptor orientation
        % Cartesian coordinate
        Settings.AOri.Cart = Aoc;
    % Expansion number
    Settings.nmax = nmax;
    % Type of boundary condition
    Settings.BC = BC;
    % Specific wavelength and wavenumber (data point)
    [~,ii] = min(abs(wavelength-lambda));
    Settings.lambda = lambda(ii);
    Settings.k0 = k0(ii);
    % Dielectric function of the environment
    Settings.nr = nr(ii,:);
    % Radial boundary condition
    Settings.rbc = rbc;
    % Radial dimensionless variable for boundary conditions
    Settings.k0s = k0s(ii,:);

clearvars -except Settings Ar Atheta Aphi Ax Ay Az FilePath ...
    FileName Resize
```

Function 25: Setting file of mapping mode

```
%% Input for wavelength mode (feel free to change variables)
% Dipole position (micrometer)
    % Donor dipole
    Dx = 0; %Do not change the value! Extremely Important!
    Dy = 0; %Do not change the value! Extremely Important!
    Dz = 0.08;
    % Acceptor dipole
    Ar = 0.08;
    Atheta = linspace(5,180,176)*(pi/180);
    Aphi = 0*(pi/180);
    Ax = Ar*sin(Atheta)*cos(Aphi);
    Ay = Ar*sin(Atheta)*sin(Aphi);
    Az = Ar*cos(Atheta);
% Dipole orientation
    % Donor dipole (Cartesian coordinate, [x;y;z])
    Doc = [0;0;1];
    % Acceptor dipole (Cartesian coordinate, [x;y;z])
    Aoc = [0;0;1];
% Largest expansion order (n) in a calculation
nmax = 50;
% Boundary conditions ('sphere' or 'coreshell')
BC = 'coreshell';
    % Calling dielectric data
    excelre = ...
       xlsread('.\DielectricFunction\dielectric function.xlsx','Ag_JPCL');
    epsilonmat = excelre(:,2) + 1i*excelre(:,3);
    % Wavelength (microns)
    lambda = excelre(:,1)*1e-3;
    %Wavenumber (microns)
    k0 = 2*pi./lambda;
    % Desired wavelength (microns)
    wavelength = 0.600;
    % Setting conditions
    if strcmp(BC,'sphere') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),2);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = sqrt(epsilonmat);
        % Radius of the sphere (unit: micron)
        rbc = 0.06;
        % Dimensionless radial variable
        k0s = k0*rbc;
    elseif strcmp(BC,'coreshell') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),3);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = 2.0   ;
        % Relative refractive index (Region 2)
        nr(:,3) = sqrt(epsilonmat);
```

```
        % Radius of the core and the shell [shell, core] (unit: micron)
        rbc = [0.070, 0.060];
        % Dimensionless radial variable
        k0s = k0*rbc;
    end

%% Settings for Function "myplot" (Default of Exporting Figures )
fplot.colorstyle = '-k';
fplot.range = [0,inf,1e28,1e36];
fplot.yscale = 'log';
fplot.xlabel = '$\mathrm{Arc~Length/\lambda}$';
fplot.ylabel = '$\mathrm{Coupling~Factor}~(\mathrm{cm}^{-6})$';
fplot.subaxis = 1;
fplot.subrange = [0,pi,1e28,1e36];
fplot.subxlabel = '$\mathrm{Angle}~(\mathrm{rad})$';
% plot.subylabel = '$\mathrm{Coupling~Factor}~(\mathrm{cm}^{-6})$';

%% ------- Do not change the following settings ------- %%
% Creating a mode structure array
    % Mode name
    Settings.ModeName = 'angle';
    % Donor position
        % Cartesian coordinate
        Settings.DPos.Cart = [Dx;Dy;Dz];
    % Acceptor position
        % Cartesian coordinate
        Settings.APos.Cart = [Ax;Ay;Az];
        % Spherical coordinate
        Settings.APos.Sph = ...
            [Ar*ones(size(Atheta));Atheta;Aphi*ones(size(Atheta))];
    % Donor orientation
        % Cartesian coordinate
        Settings.DOri.Cart = Doc;
    % Acceptor orientation
        % Cartesian coordinate
        Settings.AOri.Cart = Aoc;
    % Expansion number
    Settings.nmax = nmax;
    % Type of boundary condition
    Settings.BC = BC;
    % Specific wavelength and wavenumber (data point)
    [~,ii] = min(abs(wavelength-lambda));
    Settings.lambda = lambda(ii);
    Settings.k0 = k0(ii);
    % Dielectric function of the environment
    Settings.nr = nr(ii,:);
    % Radial boundary condition
    Settings.rbc = rbc;
    % Radial dimensionless variable for boundary conditions
    Settings.k0s = k0s(ii,:);
clearvars -except Settings Ar Atheta Aphi Ax Ay Az FilePath FileName ...
    fplot Resize
```

Function 26: Setting file of angle mode

```
%% Input for wavelength mode (feel free to change variables)
% Dipole position (micrometer)
    % Donor dipole
    Dx =  0; %Do not change the value! Extremely Important!
    Dy =  0; %Do not change the value! Extremely Important!
    Dz =  0.010;
% Dipole orientation
    % Donor dipole (Cartesian coordinate, [x;y;z])
    Doc = [1;0;0];
% Largest expansion order (n) in a calculation
nmax = 30;
% Boundary conditions ('sphere' or 'coreshell')
BC = 'sphere';
    % Calling dielectric data
    excelre = ...
       xlsread('.\DielectricFunction\dielectric function.xlsx','Ag_JPCL');
    epsilonmat = excelre(:,2) + 1i*excelre(:,3);
    % Wavelength (microns)
    lambda = excelre(:,1)*1e-3;
    %Wavenumber (microns)
    k0 = 2*pi./lambda;
    % Setting conditions
    if strcmp(BC,'sphere') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),2);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = sqrt(epsilonmat);
        % Radius of the sphere (unit: micron)
        rbc = 0.005;
        % Dimensionless radial variable
        k0s = k0*rbc;
    elseif strcmp(BC,'coreshell') == 1
        % Preallocation
        nr = zeros(size(epsilonmat,1),3);
        % Relative refractive index (Region 0)
        nr(:,1) = 1;
        % Relative refractive index (Region 1)
        nr(:,2) = 2;
        % Relative refractive index (Region 2)
        nr(:,3) = sqrt(epsilonmat);
        % Radius of the core and the shell [shell, core] (unit: micron)
        rbc = [0.070, 0.060];
        % Dimensionless radial variable
        k0s = k0*rbc;
    end

%% Settings for Function "myplot" (Default of Exporting Figures )
fplot.colorstyle = '-k';
fplot.range = [-inf,inf,1e0,1e6];
fplot.yscale = 'log';
%fplot.xlabel = '$\mathrm{Wavenumber}~(\mathrm{cm}^{-1})$';
```

```
fplot.xlabel = '$\mathrm{Wavelength}~(\mathrm{nm})$';
fplot.ylabel = '$\mathrm{Purcell~Factor}$';
fplot.subaxis = 1;
fplot.subrange = [-inf,inf,-inf,inf];
fplot.subxlabel = '$\mathrm{Wavelength}~(\mathrm{nm})$';

%% ------- Do not change the following settings ------- %%
% Creating a mode structure array
    % Mode name
    Settings.ModeName = 'Purcell';
    % Donor position
        % Cartesian coordinate
        Settings.DPos.Cart = [Dx;Dy;Dz];
    % Donor orientation
        % Cartesian coordinate
        Settings.DOri.Cart = Doc;
    % Expansion number
    Settings.nmax = nmax;
    % Type of boundary condition
    Settings.BC = BC;
    % Range of wavelength and wavenumber (data point)
    Settings.lambda = lambda;
    Settings.k0 = k0;
    % Dielectric function of the environment
    Settings.nr = nr;
    % Radial boundary condition
    Settings.rbc = rbc;
    % Radial dimensionless variable for boundary conditions
    Settings.k0s = k0s;

clearvars -except Settings lambda k0 nr k0s FilePath FileName fplot Resize
```

Function 27: Setting file of Purcell Mode

K. Still Constructing

For a source placed in a simple cavity with an infinitely extending shell, the electric field for each region can be written as

$$\mathbf{E}^{(0)}(\mathbf{r},\omega) = \mathbf{E}^{(0)}_{\text{pene}}(\mathbf{r},\omega), \tag{214a}$$

$$\mathbf{E}^{(1)}(\mathbf{r},\omega) = \mathbf{E}^{(1)}_{\text{sour}}(\mathbf{r},\omega) + \mathbf{E}^{(1)}_{\text{refl}}(\mathbf{r},\omega). \tag{214b}$$

Different from the case of the single sphere, only the penetrated field exists, $\mathbf{E}^{(0)}_{\text{pene}}(\mathbf{r},\omega)$, because the source is placed in region 1. In the first region, $\mathbf{E}^{(1)}_{\text{sour}}(\mathbf{r},\omega)$ and $\mathbf{E}^{(1)}_{\text{refl}}(\mathbf{r},\omega)$ describe the source field and the reflection field, respectively. In the same way, we project the fields on the spherical vector functions, $\underline{\mathbf{N}}^{(j)}_{nm}(k_i r,\theta,\phi)$ and $\underline{\mathbf{M}}^{(j)}_{nm}(k_i r,\theta,\phi)$, and yields

$$\mathbf{E}^{(0)}_{\text{pene}}(\mathbf{r},\omega) = \sum_{nm}\left[r_{nm}\alpha^{(0)}_n \underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi) + s_{nm}\beta^{(0)}_n \underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0 r,\theta,\phi)\right], \qquad r > r_1 \tag{215a}$$

$$\mathbf{E}^{(1)}_{\text{sour}}(\mathbf{r},\omega) = \sum_{nm}\left[r_{nm}\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_1 r,\theta,\phi) + s_{nm}\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_1 r,\theta,\phi)\right], \qquad r_1 > r > r_{\text{D}} \tag{215b}$$

$$\mathbf{E}^{(1)}_{\text{sour}}(\mathbf{r},\omega) = \sum_{nm}\left[p_{nm}\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) + q_{nm}\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi)\right], \qquad r_{\text{D}} > r > 0 \tag{215c}$$

$$\mathbf{E}^{(1)}_{\text{core}}(\mathbf{r},\omega) = \sum_{nm}\left[r_{nm}\delta^{(1)}_n \underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi) + s_{nm}\gamma^{(1)}_n \underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1 r,\theta,\phi)\right], \qquad r_1 > r > 0. \tag{215d}$$

Additionally, the magnetic fields are deduced by the relation: $i\omega\mu_0\mathbf{H}^{(i)}(\mathbf{r}_i,\omega) = \nabla\times\mathbf{E}^{(i)}(\mathbf{r}_i,\omega)$. By considering Eqs. (73a) - (73d) at $r_i = r_1$, we get the four equations,

$$\begin{array}{rcrcl}
\underline{n}_0\xi'_n(\underline{n}_1\rho_1) & + & \underline{n}_0\psi'_n(\underline{n}_1\rho_1)\delta^{(1)}_n & = & \underline{n}_1\xi'_n(\underline{n}_0\rho_1)\alpha^{(0)}_n \\
\xi_n(\underline{n}_1\rho_1) & + & \psi_n(\underline{n}_1\rho_1)\delta^{(1)}_n & = & \xi_n(\underline{n}_0\rho_1)\alpha^{(0)}_n \\
\underline{n}_0\xi_n(\underline{n}_1\rho_1) & + & \underline{n}_0\psi_n(\underline{n}_1\rho_1)\gamma^{(1)}_n & = & \underline{n}_1\xi_n(\underline{n}_0\rho_1)\beta^{(0)}_n \\
\xi'_n(\underline{n}_1\rho_1) & + & \psi'_n(\underline{n}_1\rho_1)\gamma^{(1)}_n & = & \xi'_n(\underline{n}_0\rho_1)\beta^{(0)}_n
\end{array}. \tag{216}$$

Solving the linear equations, we get the analytical expression of Mie coefficients,

$$\alpha_n^{(0)} = \frac{\underline{n}_0\xi_n'(\underline{n}_1\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\xi_n(\underline{n}_1\rho_1)\psi_n'(\underline{n}_1\rho_1)}{\underline{n}_1\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)} \tag{217a}$$

$$\beta_n^{(0)} = \frac{\underline{n}_0\xi_n'(\underline{n}_1\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\xi_n(\underline{n}_1\rho_1)\psi_n'(\underline{n}_1\rho_1)}{\underline{n}_0\xi_n'(\underline{n}_0\rho_1)\psi_n(\underline{n}_1\rho_1) - \underline{n}_1\xi_n(\underline{n}_0\rho_1)\psi_n'(\underline{n}_1\rho_1)}, \tag{217b}$$

$$\gamma_n^{(1)} = -\frac{\underline{n}_1\xi_n'(\underline{n}_1\rho_1)\xi_n(\underline{n}_0\rho_1) - \underline{n}_0\xi_n(\underline{n}_1\rho_1)\xi_n'(\underline{n}_0\rho_1)}{\underline{n}_1\psi_n'(\underline{n}_1\rho_1)\xi_n(\underline{n}_0\rho_1) - \underline{n}_0\psi_n(\underline{n}_1\rho_1)\xi_n'(\underline{n}_0\rho_1)}, \tag{217c}$$

$$\delta_n^{(1)} = -\frac{\underline{n}_0\xi_n'(\underline{n}_1\rho_1)\xi_n(\underline{n}_0\rho_1) - \underline{n}_1\xi_n(\underline{n}_1\rho_1)\xi_n'(\underline{n}_0\rho_1)}{\underline{n}_0\psi_n'(\underline{n}_1\rho_1)\xi_n(\underline{n}_0\rho_1) - \underline{n}_1\psi_n(\underline{n}_1\rho_1)\xi_n'(\underline{n}_0\rho_1)}. \tag{217d}$$

Recall that

$$\mathscr{D}\psi_n(z) = \frac{\mathrm{d}}{\mathrm{d}z}\ln\psi_n(z) = \frac{\psi_n'(z)}{\psi_n(z)}, \tag{218a}$$

$$\mathscr{D}\xi_n(z) = \frac{\mathrm{d}}{\mathrm{d}z}\ln\xi_n(z) = \frac{\xi_n'(z)}{\xi_n(z)}. \tag{218b}$$

$$\alpha_n^{(0)} = \frac{\underline{n}_0\mathscr{D}\xi_n(\underline{n}_1\rho_1) - \underline{n}_0\mathscr{D}\psi_n(\underline{n}_1\rho_1)}{\underline{n}_1\mathscr{D}\xi_n(\underline{n}_0\rho_1) - \underline{n}_0\mathscr{D}\psi_n(\underline{n}_1\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\xi_n(\underline{n}_0\rho_1)} \tag{219a}$$

$$\beta_n^{(0)} = \frac{\underline{n}_0\mathscr{D}\xi_n(\underline{n}_1\rho_1) - \underline{n}_0\mathscr{D}\psi_n(\underline{n}_1\rho_1)}{\underline{n}_0\mathscr{D}\xi_n(\underline{n}_0\rho_1) - \underline{n}_1\mathscr{D}\psi_n(\underline{n}_1\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\xi_n(\underline{n}_0\rho_1)}, \tag{219b}$$

$$\gamma_n^{(1)} = -\frac{\underline{n}_1\mathscr{D}\xi_n(\underline{n}_1\rho_1) - \underline{n}_0\mathscr{D}\xi_n(\underline{n}_0\rho_1)}{\underline{n}_1\mathscr{D}\psi_n(\underline{n}_1\rho_1) - \underline{n}_0\mathscr{D}\xi_n(\underline{n}_0\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\psi_n(\underline{n}_1\rho_1)}, \tag{219c}$$

$$\delta_n^{(1)} = -\frac{\underline{n}_0\mathscr{D}\xi_n(\underline{n}_1\rho_1) - \underline{n}_1\mathscr{D}\xi_n(\underline{n}_0\rho_1)}{\underline{n}_0\mathscr{D}\psi_n(\underline{n}_1\rho_1) - \underline{n}_1\mathscr{D}\xi_n(\underline{n}_0\rho_1)} \cdot \frac{\xi_n(\underline{n}_1\rho_1)}{\psi_n(\underline{n}_1\rho_1)}. \tag{219d}$$

Applying Eq. (217) to Eq. (215), we are able to calculate the electric fields for each region now. Furthermore, because the electric fields are equivalent to the Green's functions, we can

also obtain the explicit expression of the dyadic Green's functions by the Mie coefficients,

$$\overline{\overline{\mathbf{G}}}^{(11)}(\mathbf{r},\mathbf{r}',\omega)=\overline{\overline{\mathbf{G}}}^{(11)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega)+\overline{\overline{\mathbf{G}}}^{(11)}_{\text{refl}}(\mathbf{r},\mathbf{r}',\omega) \tag{220}$$

$$\begin{aligned}\overline{\overline{\mathbf{G}}}^{(01)}(\mathbf{r},\mathbf{r}',\omega)=ik_1\sum_{nm}(-1)^m\Big[&\beta_n^{(0)}\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_0r,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\text{I})}_{n(-m)}(k_1r',\theta',\phi')\\&+\alpha_n^{(0)}\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_0r,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\text{I})}_{n(-m)}(k_1r',\theta',\phi')\Big],\end{aligned} \tag{221}$$

where $\overline{\overline{\mathbf{G}}}^{(11)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega)$ and $\overline{\overline{\mathbf{G}}}^{(11)}_{\text{refl}}(\mathbf{r},\mathbf{r}',\omega)$ are

$$\overline{\overline{\mathbf{G}}}^{(11)}_{\text{sour}}(\mathbf{r},\mathbf{r}',\omega)=\begin{cases}ik_1\sum_{nm}(-1)^m\Big[\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1r,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\text{III})}_{n(-m)}(k_1r',\theta',\phi')\\\qquad+\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1r,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\text{III})}_{n(-m)}(k_1r',\theta',\phi')\Big], & r<r'\\ ik_1\sum_{nm}(-1)^m\Big[\underline{\mathbf{M}}^{(\text{III})}_{nm}(k_1r,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\text{I})}_{n(-m)}(k_1r',\theta',\phi')\\\qquad+\underline{\mathbf{N}}^{(\text{III})}_{nm}(k_1r,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\text{I})}_{n(-m)}(k_1r',\theta',\phi')\Big], & r>r'\end{cases}, \tag{222}$$

and

$$\begin{aligned}\overline{\overline{\mathbf{G}}}^{(11)}_{\text{refl}}(\mathbf{r},\mathbf{r}',\omega)=ik_1\sum_{nm}(-1)^m\Big[&\gamma_n^{(1)}\underline{\mathbf{M}}^{(\text{I})}_{nm}(k_1r,\theta,\phi)\otimes\underline{\mathbf{M}}^{(\text{I})}_{n(-m)}(k_1r',\theta',\phi')\\&+\delta_n^{(1)}\underline{\mathbf{N}}^{(\text{I})}_{nm}(k_1r,\theta,\phi)\otimes\underline{\mathbf{N}}^{(\text{I})}_{n(-m)}(k_1r',\theta',\phi')\Big],\end{aligned} \tag{223}$$

respectively.

References

(1) Chew, W. C. *Waves and Fields in Inhomogeneous Media*; IEEE: New York, 1995.

(2) Bohren, C. F.; Huffman, D. R. *Absorption and scattering of light by small particles*; John Wiley & Sons, 2008; pp 89–92.

(3) Jia, X. Calculation of auxiliary functions related to Riccati–Bessel functions in Mie scattering. *Journal of Modern Optics* **2016**, *63*, 2348–2355.

(4) Crandall, R. Computation of Special Functions by Shanjie Zhang and Jianming Jin. *AMERICAN JOURNAL OF PHYSICS* **1997**, *65*, 355–355.

(5) Feng, X.; Wang, P.; Yang, W.; Jin, G. High-precision evaluation of Wigner's d matrix by exact diagonalization. *Physical Review E* **2015**, *92*, 043307.

(6) Yang, W. Improved recursive algorithm for light scattering by a multilayered sphere. *Applied optics* **2003**, *42*, 1710–1720.

(7) Mackowski, D.; Altenkirch, R.; Menguc, M. Internal absorption cross sections in a stratified sphere. *Applied Optics* **1990**, *29*, 1551–1559.

(8) Thompson, I. NIST Handbook of Mathematical Functions, edited by Frank WJ Olver, Daniel W. Lozier, Ronald F. Boisvert, Charles W. Clark. 2011.

(9) Tasdemir, K. Overwritable message outputs to commandline window. 2022; `https://www.mathworks.com/matlabcentral/fileexchange/44673-overwritable-message-outputs-to-commandline-window`.