CHAPTER 9

Near-field to far-field transformation

In previous chapters, the finite-difference time-domain (FDTD) method is used to compute electric and magnetic fields within a finite space around an electromagnetic object (i.e., the near-zone electromagnetic fields). In many applications, such as antennas and radar cross-section, it is necessary to find the radiation or scattered fields in the region that is far away from an antenna or scatterer. With the FDTD technique, the direct evaluation of the far field calls for an excessively large computational domain, which is not practical in applications. Instead, the far-zone electromagnetic fields are computed from the near-field FDTD data through a near-field to far-field (NF–FF) transformation technique [1].

A simple condition for the far field is defined as follows:

$$kR \gg 1 \Rightarrow \frac{2\pi R}{\lambda} \gg 1, \tag{9.1}$$

where R is the distance from radiator to the observation point, k is the wavenumber in free space, and λ is the wavelength. For an electrically large antenna such as a parabolic reflector, the aperture size D is often used to determine the far-field condition [29]:

$$r > \frac{2D^2}{\lambda}, \tag{9.2}$$

where r is the distance from the center of the antenna aperture to the observation point. In the far-field region, the electromagnetic field at an observation point (r, θ, ϕ) can be expressed as

$$\vec{E}(r, \theta, \phi) = \frac{e^{-jkr}}{4\pi r}\vec{F}(\theta, \phi), \tag{9.3a}$$

$$\vec{H} = \hat{r} \times \frac{\vec{E}}{\eta_0}, \tag{9.3b}$$

where η_0 is the wave impedance of free space, and $\vec{F}(\theta, \phi)$ is a term determining the angular variations of the far-field pattern of the electric field. Thus, the radiation pattern of the antenna is only a function of the angular position (θ, ϕ) and is independent of the distance r.

In general, the near-field to far-field transformation technique is implemented in a two-step procedure. First, an imaginary surface is selected to enclose the antenna, as shown in Figure 9.1. The currents $\vec{J}$ and $\vec{M}$ on the surface are determined by the computed $\vec{E}$ and $\vec{H}$ fields inside the computational domain. According to the equivalence theorem, the radiation field from the currents is equivalent to the radiation field from the antenna. Next, the vector potentials $\vec{A}$ and $\vec{F}$ are used to compute the radiation fields from the equivalent currents $\vec{J}$ and $\vec{M}$. The far-field conditions are used in the derivations to obtain the appropriate analytical formulas.

Compared with the direct FDTD simulation that requires a mesh extending many wavelengths from the object, a much smaller FDTD mesh is needed to evaluate the equivalent currents $\vec{J}$ and $\vec{M}$. Thus, it is much more computationally efficient to use this near-field to far-field transformation technique.

According to different computation objectives, the transformation technique can be applied in both the time and frequency domains, as shown in Figure 9.2. When transient or broadband frequency-domain results are required at a limited number of observation angles, the left path in Figure 9.2 is adopted. For these situations, the time-domain transformation is used and the transient far-zone fields at each angle of interest are stored while updating the field components [30,31].

In contrast, when the far fields at all observation angles are required for a limited number of frequencies, the right path in Figure 9.2 is adopted. For each frequency of interest a running discrete Fourier transform (DFT) of the tangential fields (surface currents) on a closed surface is updated at each time step. The complex frequency-domain currents obtained from the DFT are then used to compute the far-zone fields at all observation angles through the frequency-domain transformation.

This chapter is organized as follows. First, the surface equivalence theorem is discussed, and the equivalent currents $\vec{J}$ and $\vec{M}$ are obtained from the near-field FDTD data. Then the important radiation formulas are presented to calculate the far fields from the equivalent currents. The implementation procedure is illustrated with full details. Finally, two antenna examples are provided to demonstrate the validity of the near- to far-field technique.

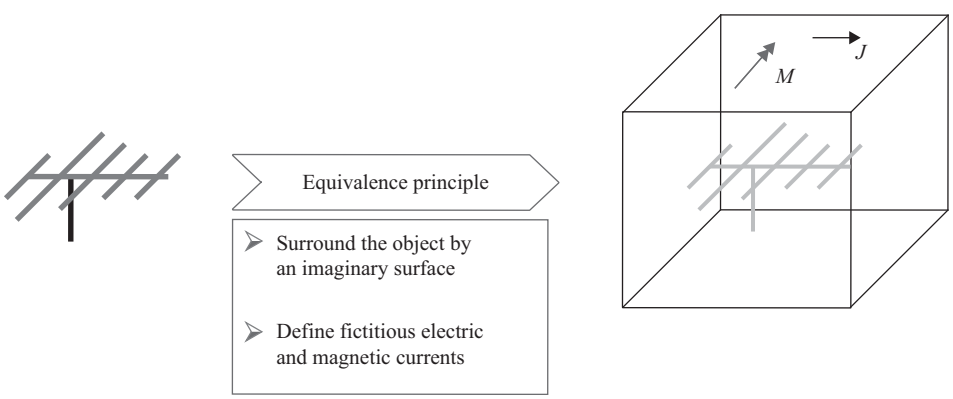

Figure 9.1 Near-field to far-field transformation technique: equivalent currents on an imaginary surface.

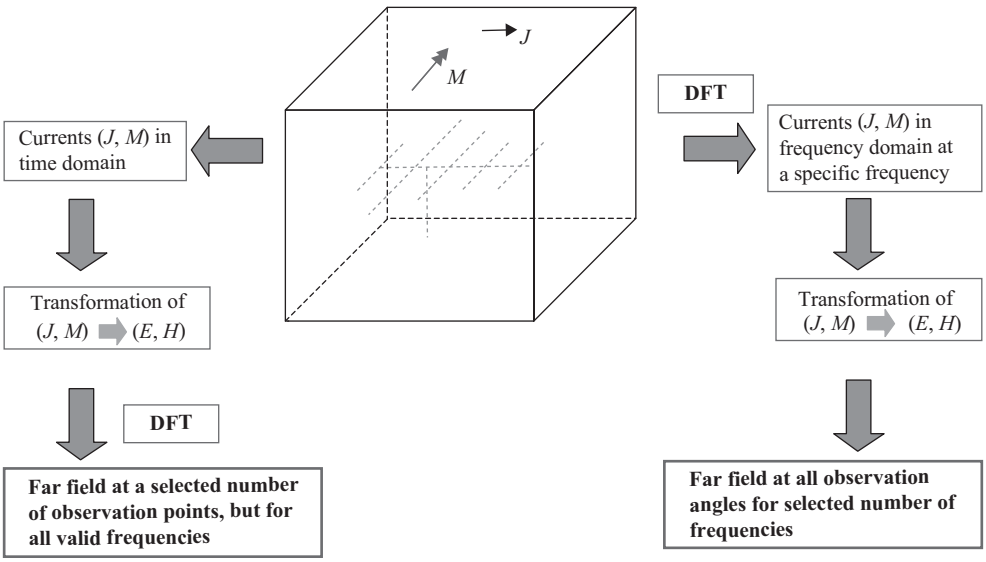

Figure 9.2 Two paths of the near-field to far-field transformation technique are implemented to achieve different computation objectives.

9.1 Implementation of the surface equivalence theorem

9.1.1 Surface equivalence theorem

The surface equivalence theorem was introduced in 1936 by Sckelkunoff [32] and is now widely used in electromagnetic and antenna problems [33]. The basic idea is to replace the actual sources such as antennas or scatterers with fictitious surface currents on a surrounding closed surface.

Within a specific region, the fields generated by the fictitious currents represent the original fields.

Figure 9.3 illustrates a typical implementation of the surface equivalent theorem. Assume that fields generated by an arbitrary source are $(\vec{E}, \vec{H})$. An imaginary surface S is selected to enclose *all* the sources and scattering objects, as shown in Figure 9.3(a). Outside the surface S is only free space. An equivalent problem is set up in Figure 9.3(b), where the fields outside the surface S remain the same but inside the surface S are set to zero. It is obvious that this setup is feasible because the fields satisfy Maxwell's equations both inside and outside the surface S. To comply with the boundary conditions on the surface, equivalent surface currents must be introduced on S:

$$\vec{J}_S = \hat{n} \times \left(\vec{H}^{out} - \vec{H}^{in}\right) = \hat{n} \times \vec{H}, \tag{9.4a}$$

$$\vec{M}_S = -\hat{n} \times \left(\vec{E}^{out} - \vec{E}^{in}\right) = -\hat{n} \times \vec{E}. \tag{9.4b}$$

It is worthwhile to emphasize that Figures 9.3(a) and 9.3(b) have the same fields outside the surface S but different fields inside the surface S.

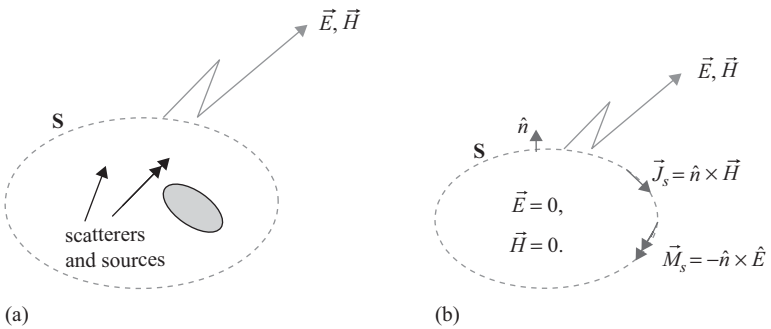

Figure 9.3 Surface equivalence theorem: (a) original problem and (b) equivalent problem for region outside *S*.

If the field values on the surface *S* in the original problem Figure 9.3(a) can be accurately obtained by some means, the surface currents in Figure 9.3(b) can be determined from (9.4). Then the fields at any arbitrary far observation point in Figure 9.3(b) can be readily calculated from the vector potential approach. Based on the uniqueness theorem, the computed fields are the only solution of the problem in Figure 9.3(b). According to the relations between Figures 9.3(a) and 9.3(b), the computed fields outside the surface *S* are also the solution for the original problem.

The implementation of the surface equivalence theorem simplifies the far-field calculation. In the original Figure 9.3(a) problem, materials with different permittivities and permeabilities may exist inside the surface *S*. Thus, a complex Green's function needs to be derived to calculate the radiating field. In the problem in Figure 9.3(b), the fields inside the surface are zero, and the permittivity and permeability can be set the same as the outside free space. Hence, the simple free space Green's function is used to compute the radiating field.

9.1.2 Equivalent surface currents in FDTD simulation

From the previous discussion, it is clear that the key point of the equivalence theorem implementation is to accurately obtain the equivalent currents on the imaginary surface *S*. In the FDTD simulation, the surface currents can be readily computed from the following procedure.

First, a closed surface is selected around the antennas or scatterers, as shown in Figure 9.4. The selected surface is usually a rectangular box that fits the FDTD grid. It is set between the analyzed objects and the outside absorbing boundary. The location of the box can be defined by two corners: lowest coordinate (*li*, *lj*, *lk*) corner and upper coordinate (*ui*, *uj*, *uk*) corner. It is critical that all the antennas or scatterers must be enclosed by this rectangular box so that the equivalent theorem can be implemented. It is also important to have this box in the air buffer area between all objects and the first interface of the absorbing boundary.

Once the imaginary closed surface is selected, the equivalent surface currents are computed next. There are six surfaces of the rectangular box, and each surface has four

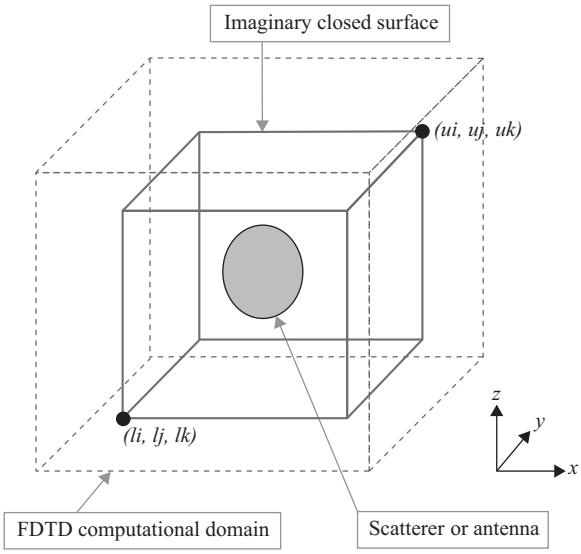

Figure 9.4 An imaginary surface is selected to enclose the antennas or scatterers.

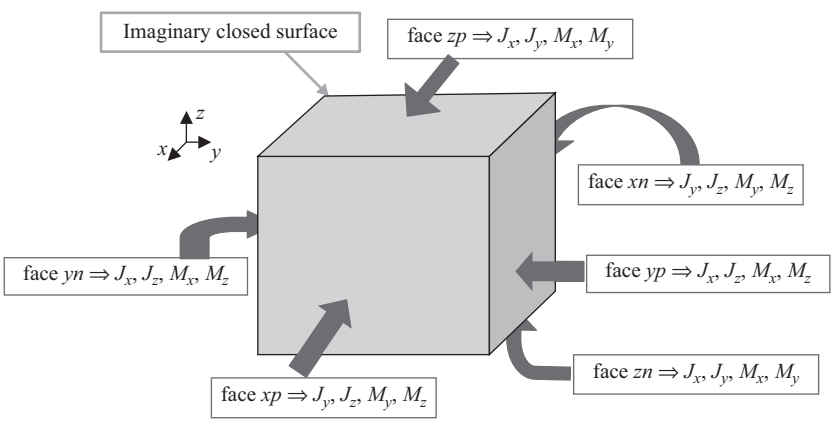

Figure 9.5 Equivalent surface currents on the imaginary closed surface.

scalar electric and magnetic currents, as shown in Figure 9.5. For the top surface, the normal direction is $\hat{z}$. From (9.4), the equivalent surface currents are calculated as

$$\vec{J}_S = \hat{z} \times \vec{H} = \hat{z} \times \left(\hat{x}H_x + \hat{y}H_y + \hat{z}H_z\right) = -\hat{x}H_y + \hat{y}H_x, \tag{9.5a}$$

$$\vec{M}_S = -\hat{z} \times \vec{E} = -\hat{z} \times \left(\hat{x}E_x + \hat{y}E_y + \hat{z}E_z\right) = \hat{x}E_y - \hat{y}E_x. \tag{9.5b}$$

Thus, the scalar surface currents can be obtained:

$$\vec{J}_S = \hat{x}J_x + \hat{y}J_y \Rightarrow J_x = -H_y, \quad J_y = H_x, \tag{9.6a}$$

$$\vec{M}_S = \hat{x}M_x + \hat{y}M_y \Rightarrow M_x = E_y, \quad M_y = -E_x. \tag{9.6b}$$

Note that the E and H fields used in (9.6) are computed from the FDTD simulation. For a time-domain far-field calculation, the time-domain data are used directly. For a frequency-domain far-field calculation, a DFT needs to be carried out to obtain the desired frequency components of the fields.

Similar methodology is used to obtain the surface currents on the other five surfaces. On the bottom surface,

$$\vec{J}_S = \hat{x}J_x + \hat{y}J_y \Rightarrow J_x = H_y, \quad J_y = -H_x, \tag{9.7a}$$

$$\vec{M}_S = \hat{x}M_x + \hat{y}M_y \Rightarrow M_x = -E_y, \quad M_y = E_x. \tag{9.7b}$$

On the left surface,

$$\vec{J}_S = \hat{x}J_x + \hat{z}J_z \Rightarrow J_x = -H_z, \quad J_z = H_x, \tag{9.8a}$$

$$\vec{M}_S = \hat{x}M_x + \hat{z}M_z \Rightarrow M_x = E_z, \quad M_z = -E_x. \tag{9.8b}$$

On the right surface,

$$\vec{J}_S = \hat{x}J_x + \hat{z}J_z \Rightarrow J_x = H_z, \quad J_z = -H_x, \tag{9.9a}$$

$$\vec{M}_S = \hat{x}M_x + \hat{z}M_z \Rightarrow M_x = -E_z, \quad M_z = E_x. \tag{9.9b}$$

On the front surface,

$$\vec{J}_S = \hat{y}J_y + \hat{z}J_z \Rightarrow J_y = -H_z, \quad J_z = H_y, \tag{9.10a}$$

$$\vec{M}_S = \hat{y}M_y + \hat{z}M_z \Rightarrow M_y = E_z, \quad M_z = -E_y. \tag{9.10b}$$

On the back surface,

$$\vec{J}_S = \hat{y}J_y + \hat{z}J_z \Rightarrow J_y = H_z, \quad J_z = -H_y, \tag{9.11a}$$

$$\vec{M}_S = \hat{y}M_y + \hat{z}M_z \Rightarrow M_y = -E_z, \quad M_z = E_y. \tag{9.11b}$$

To obtain complete source currents for the far-field calculation, (9.6)–(9.11) must be calculated at every FDTD cell on the equivalent closed surface. It is preferable that the magnetic and electric currents should be located at the same position, namely, the center of each Yee's cell surface that touches the equivalent surface S. Because of the spatial offset between the components of the E and H field locations on Yee's cell, averaging of the field components may be performed to obtain the value of the current components at the center location. The obtained surface currents are then used to compute the far-field pattern in the next section.

9.1.3 Antenna on infinite ground plane

It was mentioned earlier that the imaginary surface must surround all the scatterers or antennas so that the equivalent currents radiate in a homogeneous medium, usually free space. For many antenna applications, the radiator is mounted on a large or infinite ground plane. In this situation, it is impractical to select a large surface to enclose the ground plane. Instead, the selected surface that encloses the radiator lies on top of the ground plane, and the ground plane effect is considered using image theory, as shown in Figure 9.6. The image currents have the same direction for horizontal magnetic current and vertical electric current. In contrast, the image currents flow along the opposite direction for horizontal electric current and vertical magnetic currents. This arrangement is valid for ground planes simulating infinite perfect electric conductor (PEC).

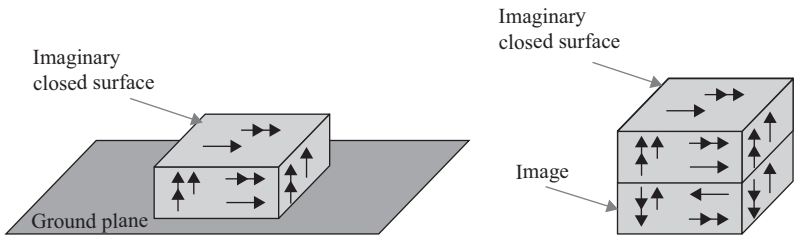

Figure 9.6 An imaginary closed surface on an infinite PEC ground plane using the image theory.

9.2 Frequency domain near-field to far-field transformation

In this section, the obtained equivalent surface currents are used to calculate the far-field radiation patterns. Antenna polarization and radiation efficiency are also discussed.

9.2.1 Time-domain to frequency-domain transformation

This section focuses on the frequency-domain far-field calculation. For the time-domain analysis, readers are suggested to study [30, 31]. The first thing in the frequency-domain calculation is to convert the time-domain FDTD data into frequency-domain data using the DFT. For example, the surface current J_y in (9.6) can be calculated as follows:

$$J_y(u, v, w; f_1) = H_x(u, v, w; f_1) = \sum_{n=1}^{N_{steps}} H_x(u, v, w; n) e^{-j2\pi f_1 n\Delta t} \Delta t. \tag{9.12}$$

Here (u, v, w) is the index for the space location, and n is the time step index. N_{steps} is the maximum number of the time steps used in the time-domain simulation. Similar formulas can be applied in calculating other surface currents in (9.6)–(9.11). Therefore, the frequency-domain patterns are calculated after all the time steps of the FDTD computation is finished. For a cubic imaginary box with $N \times N$ cells on each surface, the total required storage size for the surface currents is $4 \times 6 \times N^2$. Note that the frequency-domain data in (9.12) have complex values.

If radiation patterns at multiple frequencies are required, a pulse excitation is used in the FDTD simulation. For each frequency of interest, the DFT in (9.12) is performed with the corresponding frequency value. One round of FDTD simulation is capable to provide surface currents and radiation patterns at multiple frequencies.

9.2.2 Vector potential approach

For radiation problems, a vector potential approach is well developed to compute the unknown far fields from the known electric and magnetic currents [33]. A pair of vector potential functions is defined as

$$\vec{A} = \frac{\mu_0 e^{-jkR}}{4\pi R}\vec{N}, \tag{9.13a}$$

$$\vec{F} = \frac{\varepsilon_0 e^{-jkR}}{4\pi R}\vec{L}, \tag{9.13b}$$

where

$$\vec{N} = \int_S \vec{J}_S e^{-jkr'\cos(\psi)} dS', \tag{9.14a}$$

$$\vec{L} = \int_S \vec{M}_S e^{-jkr'\cos(\psi)} dS'. \tag{9.14b}$$

As illustrated in Figure 9.7, the vector $\vec{r} = r\hat{r}$ denotes the position of the observation point (x, y, z), whereas the vector $\vec{r}' = r'\hat{r}'$ denotes the position of source point (x', y', z') on the surface S. The vector $\vec{R} = R\hat{R}$ is between the source point and the observation point, and

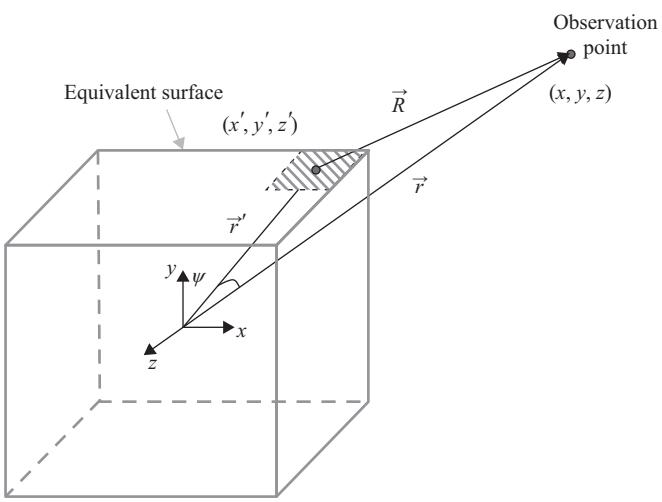

Figure 9.7 The equivalent surface current source and far field.

the angle ψ represents the angle between $\vec{r}$ and $\vec{r}'$. In the far-field calculation, the distance R is approximated by

$$R = \sqrt{r^2 + (r')^2 - 2rr'\cos(\psi)} = \begin{cases} r - r'\cos(\psi) & \text{for the phase term} \\ r & \text{for the amplitude term} \end{cases} \tag{9.15}$$

The computation of the components of E and H in the far fields can then be obtained using the vector potentials, which are expressed as

$$E_r = 0, \tag{9.16a}$$

$$E_\theta = -\frac{jke^{-jkr}}{4\pi r}\left(L_\phi + \eta_0 N_\theta\right), \tag{9.16b}$$

$$E_\phi = +\frac{jke^{-jkr}}{4\pi r}\left(L_\theta - \eta_0 N_\phi\right), \tag{9.16c}$$

$$H_r = 0, \tag{9.16d}$$

$$H_\theta = +\frac{jke^{-jkr}}{4\pi r}\left(N_\phi - \frac{L_\theta}{\eta_0}\right), \tag{9.16e}$$

$$H_\phi = -\frac{jke^{-jkr}}{4\pi r}\left(N_\theta + \frac{L_\phi}{\eta_0}\right). \tag{9.16f}$$

When the closed surface S is chosen as in Figure 9.4, the equivalent surface currents are computed based on (9.6)–(9.11), and the DFT in (9.12) is performed to obtain frequency-domain components; the auxiliary functions N and L are calculated as

$$N_\theta = \int_S \left(J_x\cos(\theta)\cos(\phi) + J_y\cos(\theta)\sin(\phi) - J_z\sin(\theta)\right)e^{jkr'\cos(\psi)}dS', \tag{9.17a}$$

$$N_\phi = \int_S \left(-J_x\sin(\phi) + J_y\cos(\phi)\right)e^{jkr'\cos(\psi)}dS', \tag{9.17b}$$

$$L_\theta = \int_S \left(M_x\cos(\theta)\cos(\phi) + M_y\cos(\theta)\sin(\phi) - M_z\sin(\theta)\right)e^{jkr'\cos(\psi)}dS', \tag{9.17c}$$

$$L_\phi = \int_S \left(-M_x\sin(\phi) + M_y\cos(\phi)\right)e^{jkr'\cos(\psi)}dS'. \tag{9.17d}$$

Substituting (9.17) into (9.16), the far-field pattern can be obtained at any observation point (r, θ, ϕ).

9.2.3 Polarization of radiation field

The E and H fields calculated in (9.16) are linearly polarized (LP) components. In some antenna applications such as satellite communications, it is desired to obtain circularly

polarized (CP) field components. This can be done through unit vector transformations between LP and CP components, such that

$$\hat{\theta} = \hat{\theta} - j\frac{\hat{\phi}}{2} + \hat{\theta} + j\frac{\hat{\phi}}{2} = \frac{\hat{E}_R}{\sqrt{2}} + \frac{\hat{E}_L}{\sqrt{2}}, \tag{9.18a}$$

$$\hat{\phi} = \hat{\theta} + j\frac{\hat{\phi}}{2j} - \hat{\theta} - j\frac{\hat{\phi}}{2j} = \frac{\hat{E}_L}{j\sqrt{2}} - \frac{\hat{E}_R}{j\sqrt{2}}, \tag{9.18b}$$

where $\hat{E}_R$ and $\hat{E}_L$ are unit vectors for the right-hand circularly polarized (RHCP) field and left-hand circularly polarized (LHCP) field. Substituting (9.18) into (9.16), we obtain

$$\begin{aligned}\vec{E} = \hat{\theta}E_\theta + \hat{\phi}E_\phi &= \left(\frac{\hat{E}_R}{\sqrt{2}} + \frac{\hat{E}_L}{\sqrt{2}}\right)E_\theta + \left(\frac{\hat{E}_L}{j\sqrt{2}} - \frac{\hat{E}_R}{j\sqrt{2}}\right)E_\phi \\ &= \hat{E}_R\left(\frac{E_\theta}{\sqrt{2}} - \frac{E_\phi}{j\sqrt{2}}\right) + \hat{E}_L\left(\frac{E_\theta}{\sqrt{2}} + \frac{E_\phi}{j\sqrt{2}}\right) = \hat{E}_R E_R + \hat{E}_L E_L,\end{aligned}$$

$$E_R = \frac{E_\theta}{\sqrt{2}} - \frac{E_\phi}{j\sqrt{2}}, \tag{9.19}$$

$$E_L = \frac{E_\theta}{\sqrt{2}} + \frac{E_\phi}{j\sqrt{2}}. \tag{9.20}$$

The magnitudes of the RHCP component (E_R) and the LHCP component (E_L) are then obtained. The axial ratio is defined to describe the polarization purity of the propagating waves and is calculated as follows [33]:

$$AR = -\frac{|E_R| + |E_L|}{|E_R| - |E_L|}. \tag{9.21}$$

For an LP wave, AR goes to infinity. For an RHCP wave $AR = -1$ and for an LHCP wave $AR = 1$. For a general elliptically polarized wave, $1 \leq |AR| \leq \infty$. Other expressions for direct computation of the AR from the far-field components E_θ and E_ϕ are given in [34]:

$$\mathrm{AR} = 20\log_{10}\left(\frac{\left[\frac{1}{2}\left(E_\phi^2 + E_\theta^2 + \left[E_\theta^4 + E_\phi^4 + 2E_\theta^2E_\phi^2\cos(2\delta)\right]^{\frac{1}{2}}\right)\right]^{\frac{1}{2}}}{\left[\frac{1}{2}\left(E_\phi^2 + E_\theta^2 + \left[E_\theta^4 + E_\phi^4 - 2E_\theta^2E_\phi^2\cos(2\delta)\right]^{\frac{1}{2}}\right)\right]^{\frac{1}{2}}}\right), \tag{9.22}$$

and in [35]

$$\mathrm{AR} = 20\log_{10}\left(\frac{|E_\phi|^2\sin^2(\tau) + |E_\theta|^2\cos^2(\tau) + |E_\phi||E_\theta|cos(\delta)\sin(2\tau)}{|E_\phi|^2\sin^2(\tau) + |E_\theta|^2\cos^2(\tau) - |E_\phi||E_\theta|cos(\delta)\sin(2\tau)}\right), \tag{9.23}$$

where

$$2\tau = \tan^{-1}\left(\frac{2|E_\phi||E_\theta|\cos(\delta)}{|E_\theta|^2 - |E_\phi|^2}\right),$$

and δ is the phase difference between E_θ and E_ϕ.

9.2.4 Radiation efficiency

The radiation efficiency is a very important indication for the effectiveness of an antenna, which can also be obtained using the FDTD technique. First of all, the radiation power of an antenna is obtained by applying the surface equivalence theorem to obtain

$$P_{rad} = \frac{1}{2} Re\left\{ \int_S \vec{E} \times \vec{H}^* \cdot \hat{n} dS' \right\} = \frac{1}{2} Re\left\{ \int_S \vec{J} * \times \vec{M} \cdot \hat{n} dS' \right\}. \tag{9.24}$$

The delivered power to an antenna is determined by the product of the voltage and current provided from the voltage source and can be expressed as

$$P_{del} = \frac{1}{2} Re\left\{ V_s(\omega) I_s^*(\omega) \right\}, \tag{9.25}$$

where $V_s(\omega)$ and $I_s(\omega)$ represent the Fourier transformed values of the source voltage and current. The antenna's radiation efficiency η_a is then defined as [36]:

$$\eta_a = \frac{P_{rad}}{P_{del}}. \tag{9.26}$$

9.3 MATLAB® implementation of near-field to far-field transformation

In this section we demonstrate the implementation of near-field to far-field transformation in the three-dimensional FDTD MATLAB code.

9.3.1 Definition of NF–FF parameters

The implementation of the CPML boundary in the three-dimensional code is demonstrated in Chapter 8. The availability of CPML makes it possible to simulate open boundary problems; the electric and magnetic fields in a problem space can be calculated as if the boundaries are free space extending to infinity. Moreover, the near-field to far-field transformation technique described in the previous section can be used to compute the far-field patterns for a number of frequencies. The desired frequencies as well as some other parameters for far-field radiation are defined in the definition section of the FDTD program. The definition of certain NF–FF transformation parameters is implemented in the subroutine ***define_output_parameters,*** partial implementation of which is shown in Listing 9.1.

In Listing 9.1, a structure named **farfield** is defined with a field **frequencies**, which is initialized as an empty array. Then the frequencies for which the far-field patterns are sought are assigned to the array **farfield.frequencies**. Another variable for **farfield** that needs to be initialized is **number_of_cells_from_outer_boundary**, which indicates the position of the imaginary NF–FF transformation surface enclosing the radiators or scatterers existing in the problem space. This imaginary surface must not coincide with any objects in the problem space or with the CPML boundaries. Therefore, **number_of_cells_from_outer_boundary** should be chosen such that it is larger than the thickness of the CPML medium and less than

Listing 9.1 define_output_parameters

```
disp('defining output parameters');

sampled_electric_fields = [];
sampled_magnetic_fields = [];
sampled_voltages = [];
sampled_currents = [];
ports = [];
farfield.frequencies = [];

% figure refresh rate
plotting_step = 10;

% mode of operation
run_simulation = true;
show_material_mesh = true;
show_problem_space = true;

% far field calculation parameters
farfield.frequencies(1) = 3e9;
farfield.frequencies(2) = 5e9;
farfield.frequencies(3) = 6e9;
farfield.frequencies(4) = 9e9;
farfield.number_of_cells_from_outer_boundary = 10;
```

the sum of the thicknesses of the CPML medium and the air gap surrounding the objects as illustrated in Figure 9.4. This parameter is used to determine the nodes (*li*, *lj*, *lk*) and (*ui*, *uj*, *uk*) indicating the boundaries of the imaginary surface in Figure 9.4.

9.3.2 Initialization of NF–FF parameters

A new subroutine, **initialize_farfield_arrays,** is added to the main FDTD program **fdtd_solve** as shown in Listing 9.2. Implementation of **initialize_farfield_arrays** is given in Listing 9.3.

The first step in the NF–FF initialization process is to determine the nodes indicating the boundaries of the imaginary NF–FF surface: (*li*, *lj*, *lk*) and (*ui*, *uj*, *uk*). Once the start and end nodes are determined they are used to construct the NF–FF auxiliary arrays. Two sets of arrays are needed for the NF–FF transformation: The first set is used to capture and store the fictitious electric and magnetic currents on the faces of the imaginary surface at every time step of the FDTD time-marching loop, and the second set is used to store the on-the-fly DFTs of these currents. The naming conventions of these arrays are elaborated as follows.

There are six faces on the imaginary surface, and the fictitious electric and magnetic currents are calculated for each face. Therefore, 12 two-dimensional arrays are needed. The names of these arrays start with the letter "t," which is followed by a letter "j" or "m." Here "j" indicates the electric current whereas "m" indicates the magnetic current being stored in the array.

Listing 9.2 fdtd_solve

```
% initialize the matlab workspace
clear all; close all; clc;

% define the problem
define_problem_space_parameters;
define_geometry;
define_sources_and_lumped_elements;
define_output_parameters;

% initialize the problem space and parameters
initialize_fdtd_material_grid;
display_problem_space;
display_material_mesh;
if run_simulation
    initialize_fdtd_parameters_and_arrays;
    initialize_sources_and_lumped_elements;
    initialize_updating_coefficients;
    initialize_boundary_conditions;
    initialize_output_parameters;
    initialize_farfield_arrays;
    initialize_display_parameters;

    % FDTD time marching loop
    run_fdtd_time_marching_loop;

    % display simulation results
    post_process_and_display_results;
end
```

Listing 9.3 initialize_farfield_arrays

```
% initialize farfield arrays
number_of_farfield_frequencies = size(farfield.frequencies,2);

if number_of_farfield_frequencies == 0
    return;
end
nc_farbuffer = farfield.number_of_cells_from_outer_boundary;
li = nc_farbuffer + 1;
lj = nc_farbuffer + 1;
lk = nc_farbuffer + 1;
ui = nx - nc_farbuffer+1;
uj = ny - nc_farbuffer+1;
uk = nz - nc_farbuffer+1;

farfield_w = 2*pi*farfield.frequencies;
```

```
16
   tjxyp = zeros(1,ui-li,1,uk-lk);
18 tjxzp = zeros(1,ui-li,uj-lj,1);
   tjyxp = zeros(1,1,uj-lj,uk-lk);
20 tjyzp = zeros(1,ui-li,uj-lj,1);
   tjzxp = zeros(1,1,uj-lj,uk-lk);
22 tjzyp = zeros(1,ui-li,1,uk-lk);
   tjxyn = zeros(1,ui-li,1,uk-lk);
24 tjxzn = zeros(1,ui-li,uj-lj,1);
   tjyxn = zeros(1,1,uj-lj,uk-lk);
26 tjyzn = zeros(1,ui-li,uj-lj,1);
   tjzxn = zeros(1,1,uj-lj,uk-lk);
28 tjzyn = zeros(1,ui-li,1,uk-lk);
   tmxyp = zeros(1,ui-li,1,uk-lk);
30 tmxzp = zeros(1,ui-li,uj-lj,1);
   tmyxp = zeros(1,1,uj-lj,uk-lk);
32 tmyzp = zeros(1,ui-li,uj-lj,1);
   tmzxp = zeros(1,1,uj-lj,uk-lk);
34 tmzyp = zeros(1,ui-li,1,uk-lk);
   tmxyn = zeros(1,ui-li,1,uk-lk);
36 tmxzn = zeros(1,ui-li,uj-lj,1);
   tmyxn = zeros(1,1,uj-lj,uk-lk);
38 tmyzn = zeros(1,ui-li,uj-lj,1);
   tmzxn = zeros(1,1,uj-lj,uk-lk);
40 tmzyn = zeros(1,ui-li,1,uk-lk);

42 cjxyp = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
   cjxzp = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
44 cjyxp = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cjyzp = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
46 cjzxp = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cjzyp = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
48 cjxyn = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
   cjxzn = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
50 cjyxn = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cjyzn = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
52 cjzxn = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cjzyn = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
54 cmxyp = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
   cmxzp = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
56 cmyxp = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cmyzp = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
58 cmzxp = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cmzyp = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
60 cmxyn = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
   cmxzn = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
62 cmyxn = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cmyzn = zeros(number_of_farfield_frequencies,ui-li,uj-lj,1);
64 cmzxn = zeros(number_of_farfield_frequencies,1,uj-lj,uk-lk);
   cmzyn = zeros(number_of_farfield_frequencies,ui-li,1,uk-lk);
```

The third character in the array name is a letter "x," "y," or "z," which indicates the direction of the current in consideration. The last two characters are "xn," "yn," "zn," "xp," "yp," or "zp" indicating the face of the imaginary surface with which the array is associated. These arrays are four-dimensional, though they are effectively two-dimensional. The size of the first dimension is 1; this dimension is added to facilitate the on-the-fly DFT calculations, as is illustrated later. The fields are calculated in three-dimensional space and are stored in three-dimensional arrays. However, it is required to capture the fields coinciding with the faces of the imaginary surface. Therefore, the sizes of the other three dimensions of the fictitious current arrays are determined by the number of field components coinciding with imaginary surface faces.

The fictitious current arrays are used to calculate DFTs of these currents. Then the currents obtained in the frequency domain are stored in their respective arrays. Thus, each fictitious current array is associated with another array in the frequency domain. Therefore, the naming conventions of the frequency-domain arrays are the same as for the ones in time domain except that they are preceded by a letter "c." Furthermore, the sizes of the first dimension of the frequency domain arrays are equal to the number of frequencies for which the far-field calculation is sought.

One additional parameter defined in Listing 9.3 is **farfield_w**. Since the far-field frequencies are used as angular frequencies, with the unit *radians/second*, during the calculations they are calculated once and stored in **farfield_w** for future use.

9.3.3 NF–FF DFT during time-marching loop

While the FDTD time-marching loop is running, the electric and magnetic fields are captured and used to calculate fictitious magnetic and electric currents using (9.6)–(9.11) on the NF–FF imaginary surface. A new subroutine called ***calculate_JandM*** is added to ***run_fdtd_time_marching_loop*** as shown in Listing 9.4. The implementation of ***calculate_JandM*** is given in Listing 9.5.

One can notice in Listing 9.5 that an average of two electric field components is used to calculate the value of a fictitious magnetic current component. Here the purpose is to obtain the values of electric field components at the centers of the faces of the cells coinciding with the NF–FF surface. For instance, Figure 9.8 illustrates the electric field components E_x on the *yn* face of the imaginary NF–FF surface. The E_x components are located on the edges of the faces of cells coinciding with the NF–FF surface. To obtain an equivalent E_x value at the centers of the faces of the cells, the average of two E_x components are calculated for each cell. Thus, the fictitious magnetic current components M_z generated by the averaged E_x components using (9.10b) are located at the centers of the faces of the cells.

Similarly, every fictitious electric current component is calculated at the centers of the faces of the cells by averaging four magnetic field components. For example, Figure 9.9 illustrates the magnetic field components H_x around the *yn* face of the imaginary NF–FF surface. The H_x components are located at the centers of the faces of cells, which are not coinciding with the NF–FF surface. To obtain equivalent H_x values at the centers of the cell faces, the average of four H_x components are calculated for each cell. Thus, the fictitious electric current components J_z generated by the averaged H_x components using (9.10a) are located at the centers of the cell faces.

Listing 9.4 run_fdtd_time_marching_loop

```
disp ('Starting the time marching loop');
disp(['Total number of time steps : ' ...
    num2str(number_of_time_steps)]);

start_time = cputime;
current_time = 0;

for time_step = 1:number_of_time_steps
    update_magnetic_fields;
    update_magnetic_field_CPML_ABC;
    capture_sampled_magnetic_fields;
    capture_sampled_currents;
    update_electric_fields;
    update_electric_field_CPML_ABC;
    update_voltage_sources;
    update_current_sources;
    update_inductors;
    update_diodes;
    capture_sampled_electric_fields;
    capture_sampled_voltages;
    calculate_JandM;
    display_sampled_parameters;
end

end_time = cputime;
total_time_in_minutes = (end_time - start_time)/60;
disp(['Total simulation time is ' ...
    num2str(total_time_in_minutes) ' minutes.']);
```

Listing 9.5 calculate_JandM

```
% Calculate J and M on the imaginary farfiled surface
if number_of_farfield_frequencies == 0
    return;
end
j = sqrt(-1);

tmyxp(1,1,:,:) =  0.5*(Ez(ui,lj:uj-1,lk:uk-1)+Ez(ui,lj+1:uj,lk:uk-1));
tmzxp(1,1,:,:) = -0.5*(Ey(ui,lj:uj-1,lk:uk-1)+Ey(ui,lj:uj-1,lk+1:uk));
tmxyp(1,:,1,:) = -0.5*(Ez(li:ui-1,uj,lk:uk-1)+Ez(li+1:ui,uj,lk:uk-1));
tmzyp(1,:,1,:) =  0.5*(Ex(li:ui-1,uj,lk:uk-1)+Ex(li:ui-1,uj,lk+1:uk));
tmxzp(1,:,:,1) =  0.5*(Ey(li:ui-1,lj:uj-1,uk)+Ey(li+1:ui,lj:uj-1,uk));
tmyzp(1,:,:,1) = -0.5*(Ex(li:ui-1,lj:uj-1,uk)+Ex(li:ui-1,lj+1:uj,uk));

tjyxp(1,1,:,:) =-0.25*(Hz(ui,lj:uj-1,lk:uk-1)+Hz(ui,lj:uj-1,lk+1:uk) ...
 + Hz (ui-1,lj:uj-1,lk:uk-1) + Hz (ui-1,lj:uj-1,lk+1:uk));

tjzxp(1,1,:,:) =0.25*(Hy(ui,lj:uj-1,lk:uk-1)+Hy(ui,lj+1:uj,lk:uk-1) ...
 + Hy (ui-1,lj:uj-1,lk:uk-1) + Hy (ui-1,lj+1:uj,lk:uk-1));

tjzyp(1,:,1,:) =-0.25*(Hx(li:ui-1,uj,lk:uk-1)+Hx(li+1:ui,uj,lk:uk-1) ...
```

```
 + Hx (li:ui-1,uj-1,lk:uk-1) + Hx (li+1:ui,uj-1,lk:uk-1));

tjxyp(1,:,1,:) = 0.25*(Hz(li:ui-1,uj,lk:uk-1)+Hz(li:ui-1,uj,lk+1:uk) ...
 + Hz (li:ui-1,uj-1,lk:uk-1) + Hz (li:ui-1,uj-1,lk+1:uk));

tjyzp(1,:,:,1) = 0.25*(Hx(li:ui-1,lj:uj-1,uk)+Hx(li+1:ui,lj:uj-1,uk) ...
 + Hx (li:ui-1,lj:uj-1,uk-1) + Hx (li+1:ui,lj:uj-1,uk-1));

tjxzp(1,:,:,1) =-0.25*(Hy(li:ui-1,lj:uj-1,uk)+Hy(li:ui-1,lj+1:uj,uk) ...
 + Hy (li:ui-1,lj:uj-1,uk-1) + Hy (li:ui-1,lj+1:uj,uk-1));

tmyxn(1,1,:,:) = -0.5 * (Ez(li,lj:uj-1,lk:uk-1)+Ez(li,lj+1:uj,lk:uk-1));
tmzxn(1,1,:,:) =  0.5 * (Ey(li,lj:uj-1,lk:uk-1)+Ey(li,lj:uj-1,lk+1:uk));

tmxyn(1,:,1,:) =  0.5 * (Ez(li:ui-1,lj,lk:uk-1)+Ez(li+1:ui,lj,lk:uk-1));
tmzyn(1,:,1,:) = -0.5 * (Ex(li:ui-1,lj,lk:uk-1)+Ex(li:ui-1,lj,lk+1:uk));

tmxzn(1,:,:,1) = -0.5 * (Ey(li:ui-1,lj:uj-1,lk)+Ey(li+1:ui,lj:uj-1,lk));
tmyzn(1,:,:,1) =  0.5 * (Ex(li:ui-1,lj:uj-1,lk)+Ex(li:ui-1,lj+1:uj,lk));

tjyxn(1,1,:,:) = 0.25*(Hz(li,lj:uj-1,lk:uk-1)+Hz(li,lj:uj-1,lk+1:uk) ...
 + Hz (li-1,lj:uj-1,lk:uk-1) + Hz (li-1,lj:uj-1,lk+1:uk));

tjzxn(1,1,:,:) =-0.25*(Hy(li,lj:uj-1,lk:uk-1)+Hy(li,lj+1:uj,lk:uk-1) ...
 + Hy (li-1,lj:uj-1,lk:uk-1) + Hy (li-1,lj+1:uj,lk:uk-1));

tjzyn(1,:,1,:) = 0.25*(Hx(li:ui-1,lj,lk:uk-1)+Hx(li+1:ui,lj,lk:uk-1) ...
 + Hx (li:ui-1,lj-1,lk:uk-1) + Hx (li+1:ui,lj-1,lk:uk-1));

tjxyn(1,:,1,:) =-0.25*(Hz(li:ui-1,lj,lk:uk-1)+Hz(li:ui-1,lj,lk+1:uk) ...
 + Hz (li:ui-1,lj-1,lk:uk-1) + Hz (li:ui-1,lj-1,lk+1:uk));

tjyzn(1,:,:,1) =-0.25*(Hx(li:ui-1,lj:uj-1,lk)+Hx(li+1:ui,lj:uj-1,lk) ...
 + Hx (li:ui-1,lj:uj-1,lk-1)+Hx(li+1:ui,lj:uj-1,lk-1));

tjxzn(1,:,:,1) = 0.25*(Hy(li:ui-1,lj:uj-1,lk)+Hy(li:ui-1,lj+1:uj,lk) ...
 + Hy (li:ui-1,lj:uj-1,lk-1) + Hy (li:ui-1,lj+1:uj,lk-1));

% fourier transform
for mi=1:number_of_farfield_frequencies
    exp_h = dt * exp(-j*farfield_w(mi)*(time_step-0.5)*dt);
    cjxyp(mi,:,:,:) = cjxyp(mi,:,:,:) + exp_h * tjxyp(1,:,:,:);
    cjxzp(mi,:,:,:) = cjxzp(mi,:,:,:) + exp_h * tjxzp(1,:,:,:);
    cjyxp(mi,:,:,:) = cjyxp(mi,:,:,:) + exp_h * tjyxp(1,:,:,:);
    cjyzp(mi,:,:,:) = cjyzp(mi,:,:,:) + exp_h * tjyzp(1,:,:,:);
    cjzxp(mi,:,:,:) = cjzxp(mi,:,:,:) + exp_h * tjzxp(1,:,:,:);
    cjzyp(mi,:,:,:) = cjzyp(mi,:,:,:) + exp_h * tjzyp(1,:,:,:);

    cjxyn(mi,:,:,:) = cjxyn(mi,:,:,:) + exp_h * tjxyn(1,:,:,:);
    cjxzn(mi,:,:,:) = cjxzn(mi,:,:,:) + exp_h * tjxzn(1,:,:,:);
    cjyxn(mi,:,:,:) = cjyxn(mi,:,:,:) + exp_h * tjyxn(1,:,:,:);
    cjyzn(mi,:,:,:) = cjyzn(mi,:,:,:) + exp_h * tjyzn(1,:,:,:);
```

```
    cjzxn(mi,:,:,:) = cjzxn(mi,:,:,:) + exp_h * tjzxn(1,:,:,:);
    cjzyn(mi,:,:,:) = cjzyn(mi,:,:,:) + exp_h * tjzyn(1,:,:,:);

    exp_e = dt * exp(-j*farfield_w(mi)*time_step*dt);

    cmxyp(mi,:,:,:) = cmxyp(mi,:,:,:) + exp_e * tmxyp(1,:,:,:);
    cmxzp(mi,:,:,:) = cmxzp(mi,:,:,:) + exp_e * tmxzp(1,:,:,:);
    cmyxp(mi,:,:,:) = cmyxp(mi,:,:,:) + exp_e * tmyxp(1,:,:,:);
    cmyzp(mi,:,:,:) = cmyzp(mi,:,:,:) + exp_e * tmyzp(1,:,:,:);
    cmzxp(mi,:,:,:) = cmzxp(mi,:,:,:) + exp_e * tmzxp(1,:,:,:);
    cmzyp(mi,:,:,:) = cmzyp(mi,:,:,:) + exp_e * tmzyp(1,:,:,:);

    cmxyn(mi,:,:,:) = cmxyn(mi,:,:,:) + exp_e * tmxyn(1,:,:,:);
    cmxzn(mi,:,:,:) = cmxzn(mi,:,:,:) + exp_e * tmxzn(1,:,:,:);
    cmyxn(mi,:,:,:) = cmyxn(mi,:,:,:) + exp_e * tmyxn(1,:,:,:);
    cmyzn(mi,:,:,:) = cmyzn(mi,:,:,:) + exp_e * tmyzn(1,:,:,:);
    cmzxn(mi,:,:,:) = cmzxn(mi,:,:,:) + exp_e * tmzxn(1,:,:,:);
    cmzyn(mi,:,:,:) = cmzyn(mi,:,:,:) + exp_e * tmzyn(1,:,:,:);
end
```

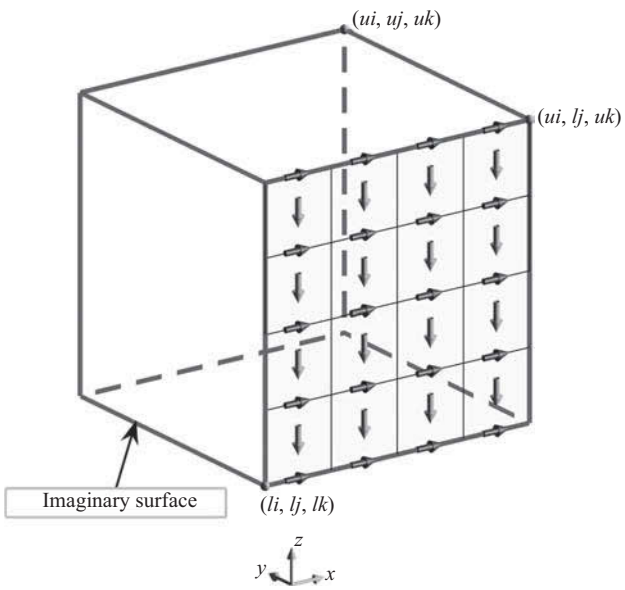

Figure 9.8 E_x field components on the *yn* face of the NF–FF imaginary surface and the magnetic currents generated by them.

After the fictitious electric and magnetic currents are sampled on the NF–FF surface at the current time step, they are used to update the on-the-fly DFTs for the far-field frequencies defined in the subroutine ***define_output_parameters***. For instance, one iteration of (9.12) is performed at each time step for calculating J_y in frequency domain. Therefore, when the FDTD time-marching loop is completed, the DFT of J_y is completed together.

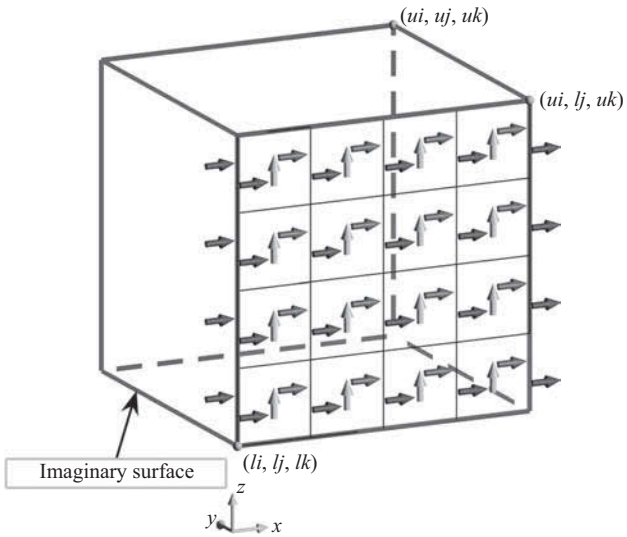

Figure 9.9 H_x field components around the *yn* face of the NF–FF imaginary surface and the electric currents generated by them.

9.3.4 Postprocessing for far-field calculation

As the FDTD time-marching loop is completed, the frequency-domain values of fictitious currents for the far-field frequencies over the NF–FF surface are available. Then they can be used at the postprocessing stage of the FDTD simulation to calculate the far-field auxiliary fields based on (9.17). Then these fields can be used to calculate the desired far-field patterns and to plot them.

A new subroutine named ***calculate_and_display_farfields*** is added to the postprocessing routine ***post_process_and_display_results*** as shown in Listing 9.6, which performs the tasks of far-field calculation and display.

Implementation of ***calculate_and_display_farfields*** is given in Listing 9.7. This subroutine initializes necessary arrays and parameters for calculation and display of far-field patterns at three principal planes: namely, the *xy*, *xz*, and *yz* planes. For instance, to calculate

Listing 9.6 post_process_and_display_results

```
disp('postprocessing and displaying simulation results');

display_transient_parameters;
calculate_frequency_domain_outputs;
display_frequency_domain_outputs;
calculate_and_display_farfields;
```

Listing 9.7 calculate_and_display_farfields

```
% This file calls the routines necessary for calculating
% farfield patterns in xy, xz, and yz plane cuts, and displays them.
% The display can be modified as desired.
% You will find the instructions the formats for
% radiation pattern plots can be set by the user.

if number_of_farfield_frequencies == 0
    return;
end

calculate_radiated_power;

j = sqrt(-1);
number_of_angles = 360;

% parameters used by polar plotting functions
step_size = 10;       % increment between the rings in the polar grid
Nrings = 4;           % number of rings in the polar grid
line_style1 = 'b-';   % line style for theta component
line_style2 = 'r--';  % line style for phi component
scale_type = 'dB';    % linear or dB
plot_type = 'D';      % the calculated data is directivity

% xy plane
% ================================================
farfield_theta = zeros(number_of_angles, 1);
farfield_phi   = zeros(number_of_angles, 1);
farfield_theta = farfield_theta + pi/2;
farfield_phi = (pi/180)*[-180:1:179].';
const_theta = 90; % used for plot

% calculate farfields
calculate_farfields_per_plane;

% plotting the farfield data
for mi=1:number_of_farfield_frequencies
 f = figure;
 pat1 = farfield_dataTheta(mi,:).';
 pat2 = farfield_dataPhi(mi,:).';

 % if scale_type is db use these, otherwise comment these two lines
 pat1 = 10*log10(pat1);
 pat2 = 10*log10(pat2);

 max_val = max(max([pat1 pat2]));
 max_val = step_size * ceil(max_val/step_size);

 legend_str1 = [plot_type '_{\theta},␣f=' ...
     num2str(farfield.frequencies(mi)*1e-9) '␣GHz'];
 legend_str2 = [plot_type '_{\phi},␣f=' ...
```

```
    num2str(farfield.frequencies(mi)*1e-9) ' GHz'];

 polar_plot_constant_theta(farfield_phi,pat1,pat2,max_val, ...
        step_size, Nrings,line_style1,line_style2,const_theta, ...
        legend_str1,legend_str2,scale_type);
end

% xz plane
% ================================================
farfield_theta = zeros(number_of_angles, 1);
farfield_phi   = zeros(number_of_angles, 1);
farfield_theta = (pi/180)*[-180:1:179].';
const_phi = 0; % used for plot

% calculate farfields
calculate_farfields_per_plane;

% plotting the farfield data
for mi=1:number_of_farfield_frequencies
 f = figure;
 pat1 = farfield_dataTheta(mi,:).';
 pat2 = farfield_dataPhi(mi,:).';

% if scale_type is db use these, otherwise comment these two lines
 pat1 = 10*log10(pat1);
 pat2 = 10*log10(pat2);

 max_val = max(max([pat1 pat2]));
 max_val = step_size * ceil(max_val/step_size);

 legend_str1 = ...
 [plot_type '-{\theta}, f=' ...
 num2str(farfield.frequencies(mi)*1e-9) ' GHz'];
 legend_str2 = ...
 [plot_type '-{\phi}, f=' ...
 num2str(farfield.frequencies(mi)*1e-9) ' GHz'];

 polar_plot_constant_phi(farfield_theta,pat1,pat2,max_val, ...
        step_size, Nrings,line_style1,line_style2,const_phi, ...
        legend_str1,legend_str2,scale_type);
end

% yz plane
% ================================================
farfield_theta = zeros(number_of_angles, 1);
farfield_phi   = zeros(number_of_angles, 1);
farfield_phi = farfield_phi + pi/2;
farfield_theta = (pi/180)*[-180:1:179].';
const_phi = 90; % used for plot

% calculate farfields
calculate_farfields_per_plane;
```

```
% plotting the farfield data
for mi=1:number_of_farfield_frequencies
 f = figure;
 pat1 = farfield_dataTheta(mi,:).';
 pat2 = farfield_dataPhi(mi,:).';

% if scale_type is db use these, otherwise comment these two lines
 pat1 = 10*log10(pat1);
 pat2 = 10*log10(pat2);

 max_val = max(max([pat1 pat2]));
 max_val = step_size * ceil(max_val/step_size);

 legend_str1 = ...
 [plot_type '-{\theta},␣f=' ...
 num2str(farfield.frequencies(mi)*1e-9) '␣GHz'];
 legend_str2 = ...
 [plot_type '-{\phi},␣f=' ...
 num2str(farfield.frequencies(mi)*1e-9) '␣GHz'];

 polar_plot_constant_phi(farfield_theta,pat1,pat2,max_val, ...
        step_size, Nrings,line_style1,line_style2,const_phi, ...
        legend_str1,legend_str2,scale_type);
end
```

the far-fields in the *xy* plane, two arrays, **farfield_theta** and **farfield_phi**, are constructed to store the angles that represent the *xy* plane. For the *xy* plane E is 90° and ϕ sweeps from −180° to 180°. Then a function ***calculate_farfields_per_plane*** is called to calculate far-field directivity data at the given angles. These data are plotted using another function ***polar_plot_constant_theta***. For the *xy* plane E is a constant value of $\pi/2$ radians. For the *xz* and *yz* planes ϕ takes constant values of 0 and $\pi/2$ radians, respectively, hence another function, which is called for plotting patterns in the *xz* and *yz* planes is ***polar_plot_constant_phi***. The implementations of ***polar_plot_constant_theta*** and ***polar_plot_constant_phi*** are given in Appendix C.

The near-field to far-field calculations are performed mainly in the function ***calculate_farfields_per_plane***, implementation of which is given in Listing 9.9.

One of the initial tasks in ***calculate_farfields_per_plane*** is the calculation of total radiated power. A subroutine named ***calculate_radiated_power*** is called to perform this task. The total radiated power is stored in the parameter **radiated_power**, as can be seen in Listing 9.8. Calculation of radiated power is based on the discrete summation representation of (9.24).

Before calculating any far-field data, the auxiliary fields N_θ, N_ϕ, L_E, and L_ϕ need to be calculated based on (9.17). In (9.17) the parameters E and ϕ are angles indicating the position vector of the observation point $\vec{r}$ as illustrated in Figure 9.7. The parameters J_x, J_y, J_z, M_x, M_y, and M_z are the fictitious currents, which have already been calculated at the center

Listing 9.8 calculate_radiated_power

```
% Calculate total radiated power
radiated_power = zeros(number_of_farfield_frequencies,1);

for mi=1:number_of_farfield_frequencies
    powr = 0;
    powr = dx*dy* sum(sum(sum(cmyzp(mi,:,:,:).* ...
        conj(cjxzp(mi,:,:,:)) - cmxzp(mi,:,:,:) ...
        .* conj(cjyzp(mi,:,:,:)))));
    powr = powr - dx*dy* sum(sum(sum(cmyzn(mi,:,:,:) ...
        .* conj(cjxzn(mi,:,:,:)) - cmxzn(mi,:,:,:) ...
        .* conj(cjyzn(mi,:,:,:)))));
    powr = powr + dx*dz* sum(sum(sum(cmxyp(mi,:,:,:) ...
        .* conj(cjzyp(mi,:,:,:)) - cmzyp(mi,:,:,:) ...
        .* conj(cjxyp(mi,:,:,:)))));
    powr = powr - dx*dz* sum(sum(sum(cmxyn(mi,:,:,:) ...
        .* conj(cjzyn(mi,:,:,:)) - cmzyn(mi,:,:,:) ...
        .* conj(cjxyn(mi,:,:,:)))));
    powr = powr + dy*dz* sum(sum(sum(cmzxp(mi,:,:,:) ...
        .* conj(cjyxp(mi,:,:,:)) - cmyxp(mi,:,:,:) ...
        .* conj(cjzxp(mi,:,:,:)))));
    powr = powr - dy*dz* sum(sum(sum(cmzxn(mi,:,:,:) ...
        .* conj(cjyxn(mi,:,:,:)) - cmyxn(mi,:,:,:) ...
        .* conj(cjzxn(mi,:,:,:)))));
    radiated_power(mi) = 0.5 * real(powr);
end
```

points of the faces of the cells coinciding with the NF–FF surface for the given far-field frequencies. The parameter k is the wavenumber expressed by

$$k = 2\pi f\sqrt{\mu_0\varepsilon_0}, \tag{9.27}$$

where f is the far-field frequency under consideration. Yet another term in (9.17) is $r'\cos(\psi)$, which needs to be further defined explicitly. As shown in Figure 9.7, $\overline{r}$ is the position vector indicating the observation point while $\overline{r}'$ is the position vector indicating the source point. The term $r'\cos(\psi)$ is actually obtained from $\overline{r}'\cdot\hat{r}$ where $\hat{r}$ is a unit vector in the direction of $\overline{r}$. The unit vector $\hat{r}$ can be expressed in Cartesian coordinates as

$$\hat{r} = \sin(\theta)\cos(\phi)\hat{x} + \sin(\theta)\sin(\phi)\hat{y} + \cos(\theta)\hat{z}. \tag{9.28}$$

As mentioned already, the vector $\overline{r}'$ is the position vector indicating the source point and is expressed in terms of source positions. The sources are located at the centers of the faces, and the source position vectors can be expressed as follows. A position vector for a source point located on the zn face of the NF–FF surface can be expressed as

$$\overline{r}' = (mi + 0.5 - ci)\Delta x\hat{x} + (mj + 0.5 - cj)\Delta y\hat{y} + (lk - ck)\Delta z\hat{z}, \tag{9.29}$$

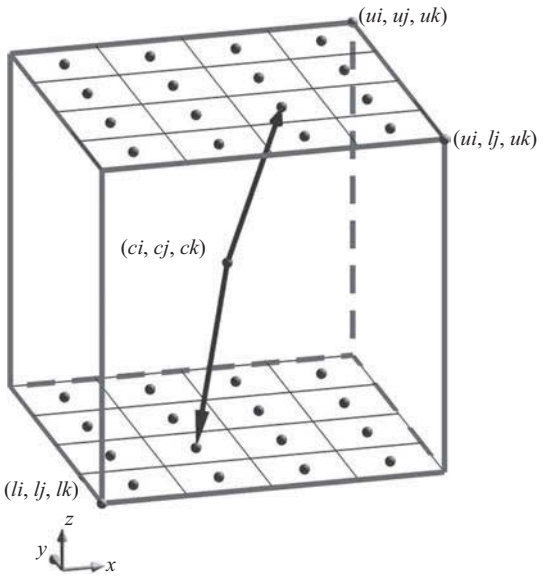

Figure 9.10 Position vectors for sources on the *zn* and *zp* faces.

as can be observed in Figure 9.10. Here (*ci*, *cj*, *ck*) is the center node of the problem space, which is assumed to be the origin for the far-field calculations. The parameters *mi* and *mj* are the indices of the nodes of the faces, which include the sources under consideration. Then the term $r'\cos(\psi)$ in (9.17) can be expressed explicitly using (9.28) and (9.29) as

$$\begin{aligned} r'\cos(\psi) = \overline{r}' \cdot \hat{r} = {} & (mi + 0.5 - ci)\Delta x \sin(\theta)\cos(\phi) \\ & + (mj + 0.5 - cj)\Delta y \sin(\theta)\sin(\phi) + (lk - ck)\Delta z \cos(\theta). \end{aligned} \tag{9.30}$$

Similarly, $r'\cos(\psi)$ can be obtained for the *zp* face as

$$\begin{aligned} r'\cos(\psi) = \overline{r}' \cdot \hat{r} = {} & (mi + 0.5 - ci)\Delta x \sin(\theta)\cos(\phi) \\ & + (mj + 0.5 - cj)\Delta y \sin(\theta)\sin(\phi) + (uk - ck)\Delta z \cos(\theta). \end{aligned} \tag{9.31}$$

The equations in (9.17) are integrations of continuous current distributions over the imaginary surface. However, we have obtained the currents at discrete points. Therefore, these integrations can be represented by discrete summations. For instance, for the *zn* and *zp* faces (9.17) can be rewritten as

$$N_\theta = \sum_{mi=li}^{ui-1} \sum_{mj=lj}^{uj-1} \Delta x \Delta y \big(J_x \cos(\theta)\cos(\phi) + J_y \cos(\theta)\sin(\phi)\big) e^{jkr'\cos(\psi)}, \tag{9.32a}$$

$$N_\phi = \sum_{mi=li}^{ui-1} \sum_{mj=lj}^{uj-1} \Delta x \Delta y \big(-J_x \sin(\phi) + J_y \cos(\phi)\big) e^{jkr'\cos(\psi)}, \tag{9.32b}$$

$$L_\theta = \sum_{mi=li}^{ui-1}\sum_{mj=lj}^{uj-1} \Delta x\Delta y\big(M_x\cos(\theta)\cos(\phi) + M_y\cos(\theta)\sin(\phi)\big)e^{jkr'\cos(\psi)}, \tag{9.32c}$$

$$L_\phi = \sum_{mi=li}^{ui-1}\sum_{mj=lj}^{uj-1} \Delta x\Delta y\big(-M_x\sin(\phi) + M_y\cos(\phi)\big)e^{jkr'\cos(\psi)}. \tag{9.32d}$$

Referring to Figure 9.11 one can obtain $r'\cos(\psi)$ for the *xn* and *xp* faces, respectively, as

$$\begin{aligned} r'\cos(\psi) = \bar{r}'\cdot\hat{r} &= (li - ci)\Delta x\sin(\theta)\cos(\phi) \\ &+ (mj + 0.5 - cj)\Delta y\sin(\theta)\sin(\phi) + (mk + 0.5 - ck)\Delta z\cos(\theta), \end{aligned} \tag{9.33}$$

$$\begin{aligned} r'\cos(\psi) = \bar{r}'\cdot\hat{r} &= (ui - ci)\Delta x\sin(\theta)\cos(\phi) \\ &+ (mj + 0.5 - cj)\Delta y\sin(\theta)\sin(\phi) + (mk + 0.5 - ck)\Delta z\cos(\theta), \end{aligned} \tag{9.34}$$

where *mj* and *mk* are the indices of the nodes of the faces that include the sources. Then the summations in 9.32 are expressed in terms of these indices *mj* and *mk* as

$$N_\theta = \sum_{mj=lj}^{uj-1}\sum_{mk=lk}^{uk-1} \Delta y\Delta z\big(J_y\cos(\theta)\sin(\phi) - J_z\sin(\theta)\big)e^{jkr'\cos(\psi)}, \tag{9.35a}$$

$$N_\phi = \sum_{mj=lj}^{uj-1}\sum_{mk=lk}^{uk-1} \Delta y\Delta z\big(J_y\cos(\phi)\big)e^{jkr'\cos(\psi)}, \tag{9.35b}$$

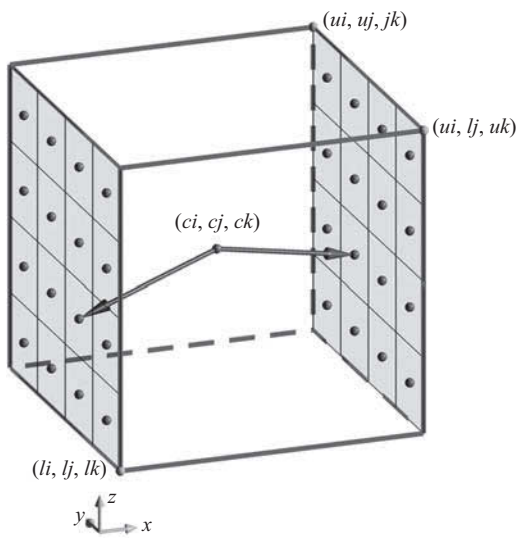

Figure 9.11 Position vectors for sources on the *xn* and *xp* faces.

$$L_\theta = \sum_{mj=lj}^{uj-1} \sum_{mk=lk}^{uk-1} \Delta y \Delta z \big(M_y \cos(\theta)\sin(\phi) - M_z \sin(\theta)\big) e^{jkr'\cos(\psi)}, \tag{9.35c}$$

$$L_\phi = \sum_{mj=lj}^{uj-1} \sum_{mk=lk}^{uk-1} \Delta y \Delta z \big(M_y \cos(\phi)\big) e^{jkr'\cos(\psi)}. \tag{9.35d}$$

Similarly, referring to Figure 9.12 $r'\cos(\psi)$ can be obtained for the *yn* and *yp* faces, respectively, as

$$\begin{aligned} r'\cos(\psi) &= (mi + 0.5 - ci)\bar{r}' \cdot \hat{r} = \Delta x \sin(\theta)\cos(\phi) \\ &\quad + (lj - cj)\Delta y \sin(\theta)\sin(\phi) + (mk + 0.5 - ck)\Delta z \cos(\theta), \end{aligned} \tag{9.36}$$

$$\begin{aligned} r'\cos(\psi) &= (mi + 0.5 - ci)\bar{r}' \cdot \hat{r} = \Delta x \sin(\theta)\cos(\phi) \\ &\quad + (uj - cj)\Delta y \sin(\theta)\sin(\phi) + (mk + 0.5 - ck)\Delta z \cos(\theta), \end{aligned} \tag{9.37}$$

Then the summations in (9.32) are expressed in terms of these indices *mi* and *mk* as

$$N_\theta = \sum_{mi=li}^{ui-1} \sum_{mk=lk}^{uk-1} \Delta x \Delta z (J_x \cos(\theta)\cos(\phi) - J_z \sin(\theta)) e^{jkr'\cos(\psi)}, \tag{9.38a}$$

$$N_\phi = \sum_{mi=li}^{ui-1} \sum_{mk=lk}^{uk-1} \Delta x \Delta z (-J_x \sin(\phi)) e^{jkr'\cos(\psi)}, \tag{9.38b}$$

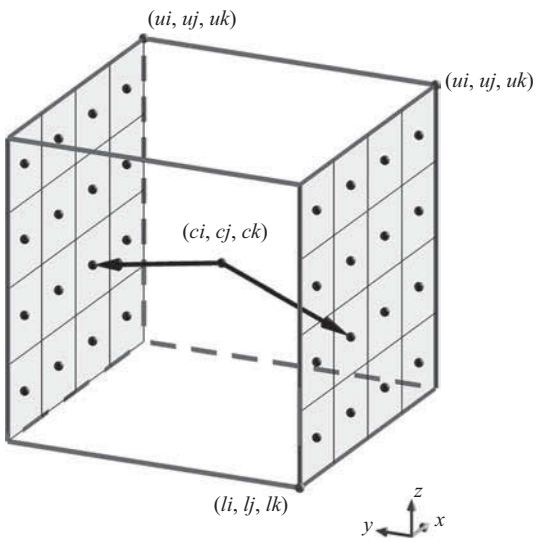

Figure 9.12 Position vectors for sources on the *yn* and *yp* faces.

$$L_\theta = \sum_{mi=li}^{ui-1} \sum_{mk=lk}^{uk-1} \Delta x \Delta z (M_x \cos(\theta)\cos(\phi) - M_z \sin(\theta)) e^{jkr'\cos(\psi)}, \tag{9.38c}$$

$$L_\phi = \sum_{mi=li}^{ui-1} \sum_{mk=lk}^{uk-1} \Delta x \Delta z (-M_x \sin(\phi)) e^{jkr'\cos(\psi)}. \tag{9.38d}$$

The implementation of the discrete summation just described can be followed in Listing 9.9.

Finally, having obtained the auxiliary fields N_θ, N_ϕ, L_E, and L_ϕ and the total radiated power, the far-field directivity data are calculated based on [1] as

$$D_\theta = \frac{k^2}{8\pi\eta_0 P_{rad}} |L_\phi + \eta_0 N_\theta|^2, \tag{9.39a}$$

$$D_\phi = \frac{k^2}{8\pi\eta_0 P_{rad}} |L_\theta - \eta_0 N_\phi|^2. \tag{9.39b}$$

Listing 9.9 calculate_farfields_per_plane

```
if number_of_farfield_frequencies == 0
    return;
end
c          = 2.99792458e+8;      % speed of light in free space
mu_0  = 4 * pi * 1e-7;      % permeability of free space
eps_0 = 1.0/(c*c*mu_0);          % permittivity of free space
eta_0 = sqrt(mu_0/eps_0);   % intrinsic impedance of free space

exp_jk_rpr = zeros(number_of_angles,1);
dx_sinth_cosphi = zeros(number_of_angles,1);
dy_sinth_sinphi = zeros(number_of_angles,1);
dz_costh = zeros(number_of_angles,1);
dy_dz_costh_sinphi = zeros(number_of_angles,1);
dy_dz_sinth = zeros(number_of_angles,1);
dy_dz_cosphi = zeros(number_of_angles,1);
dx_dz_costh_cosphi = zeros(number_of_angles,1);
dx_dz_sinth = zeros(number_of_angles,1);
dx_dz_sinphi = zeros(number_of_angles,1);
dx_dy_costh_cosphi = zeros(number_of_angles,1);
dx_dy_costh_sinphi = zeros(number_of_angles,1);
dx_dy_sinphi = zeros(number_of_angles,1);
dx_dy_cosphi = zeros(number_of_angles,1);
farfield_dataTheta = ...
    zeros(number_of_farfield_frequencies,number_of_angles);
farfield_dataPhi = ...
    zeros(number_of_farfield_frequencies,number_of_angles);

dx_sinth_cosphi = dx*sin(farfield_theta).*cos(farfield_phi);
dy_sinth_sinphi = dy*sin(farfield_theta).*sin(farfield_phi);
dz_costh = dz*cos(farfield_theta);
```

```
dy_dz_costh_sinphi = dy*dz*cos(farfield_theta).*sin(farfield_phi);
dy_dz_sinth = dy*dz*sin(farfield_theta);
dy_dz_cosphi = dy*dz*cos(farfield_phi);
dx_dz_costh_cosphi = dx*dz*cos(farfield_theta).*cos(farfield_phi);
dx_dz_sinth = dx*dz*sin(farfield_theta);
dx_dz_sinphi = dx*dz*sin(farfield_phi);
dx_dy_costh_cosphi = dx*dy*cos(farfield_theta).*cos(farfield_phi);
dx_dy_costh_sinphi = dx*dy*cos(farfield_theta).*sin(farfield_phi);
dx_dy_sinphi = dx*dy*sin(farfield_phi);
dx_dy_cosphi = dx*dy*cos(farfield_phi);
ci = 0.5*(ui+li);
cj = 0.5*(uj+lj);
ck = 0.5*(uk+lk);

% calculate directivity
for mi=1:number_of_farfield_frequencies
    disp(['Calculating directivity for ', ...
        num2str(farfield.frequencies(mi)) ' Hz']);
    k = 2*pi*farfield.frequencies(mi)*(mu_0*eps_0)^0.5;

    Ntheta = zeros(number_of_angles,1);
    Ltheta = zeros(number_of_angles,1);
    Nphi = zeros(number_of_angles,1);
    Lphi = zeros(number_of_angles,1);
    rpr = zeros(number_of_angles,1);

    for nj = lj:uj-1
        for nk =lk:uk-1
        % for +ax direction

        rpr = (ui - ci)*dx_sinth_cosphi ...
            + (nj-cj+0.5)*dy_sinth_sinphi ...
            + (nk-ck+0.5)*dz_costh;
        exp_jk_rpr = exp(j*k*rpr);
        Ntheta = Ntheta ...
            + (cjyxp(mi,1,nj-lj+1,nk-lk+1).*dy_dz_costh_sinphi ...
            - cjzxp(mi,1,nj-lj+1,nk-lk+1).*dy_dz_sinth).*exp_jk_rpr;
        Ltheta = Ltheta ...
            + (cmyxp(mi,1,nj-lj+1,nk-lk+1).*dy_dz_costh_sinphi ...
            - cmzxp(mi,1,nj-lj+1,nk-lk+1).*dy_dz_sinth).*exp_jk_rpr;
        Nphi = Nphi ...
            + (cjyxp(mi,1,nj-lj+1,nk-lk+1).*dy_dz_cosphi).*exp_jk_rpr;
        Lphi = Lphi ...
            + (cmyxp(mi,1,nj-lj+1,nk-lk+1).*dy_dz_cosphi).*exp_jk_rpr;

        % for -ax direction
        rpr = (li - ci)*dx_sinth_cosphi ...
            + (nj-cj+0.5)*dy_sinth_sinphi ...
            + (nk-ck+0.5)*dz_costh;
        exp_jk_rpr = exp(j*k*rpr);
        Ntheta = Ntheta ...
            + (cjyxn(mi,1,nj-lj+1,nk-lk+1).*dy_dz_costh_sinphi ...
```

```
            - cjzxn(mi,1,nj-lj+1,nk-lk+1).*dy_dz_sinth).*exp_jk_rpr;
        Ltheta = Ltheta ...
            + (cmyxn(mi,1,nj-lj+1,nk-lk+1).*dy_dz_costh_sinphi ...
            - cmzxn(mi,1,nj-lj+1,nk-lk+1).*dy_dz_sinth).*exp_jk_rpr;
        Nphi = Nphi ...
            + (cjyxn(mi,1,nj-lj+1,nk-lk+1).*dy_dz_cosphi).*exp_jk_rpr;
        Lphi = Lphi ...
            + (cmyxn(mi,1,nj-lj+1,nk-lk+1).*dy_dz_cosphi).*exp_jk_rpr;
        end
    end
    for ni =li:ui-1
        for nk =lk:uk-1
        % for +ay direction

        rpr = (ni - ci + 0.5)*dx_sinth_cosphi ...
            + (uj-cj)*dy_sinth_sinphi ...
            + (nk-ck+0.5)*dz_costh;
        exp_jk_rpr = exp(j*k*rpr);

        Ntheta = Ntheta ...
            + (cjxyp(mi,ni-li+1,1,nk-lk+1).*dx_dz_costh_cosphi ...
            - cjzyp(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinth).*exp_jk_rpr;
        Ltheta = Ltheta ...
            + (cmxyp(mi,ni-li+1,1,nk-lk+1).*dx_dz_costh_cosphi ...
            - cmzyp(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinth).*exp_jk_rpr;
        Nphi = Nphi ...
            + (-cjxyp(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinphi).*exp_jk_rpr;
        Lphi = Lphi ...
            + (-cmxyp(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinphi).*exp_jk_rpr;

        % for -ay direction
        rpr = (ni - ci + 0.5)*dx_sinth_cosphi ...
            + (lj-cj)*dy_sinth_sinphi ...
            + (nk-ck+0.5)*dz_costh;
        exp_jk_rpr = exp(j*k*rpr);

        Ntheta = Ntheta ...
            + (cjxyn(mi,ni-li+1,1,nk-lk+1).*dx_dz_costh_cosphi ...
            - cjzyn(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinth).*exp_jk_rpr;
        Ltheta = Ltheta ...
            + (cmxyn(mi,ni-li+1,1,nk-lk+1).*dx_dz_costh_cosphi ...
            - cmzyn(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinth).*exp_jk_rpr;
        Nphi = Nphi ...
            + (-cjxyn(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinphi).*exp_jk_rpr;
        Lphi = Lphi ...
            + (-cmxyn(mi,ni-li+1,1,nk-lk+1).*dx_dz_sinphi).*exp_jk_rpr;
        end
    end

    for ni =li:ui-1
        for nj =lj:uj-1
        % for +az direction
```

```
135
        rpr = (ni-ci+0.5)*dx_sinth_cosphi ...
137         + (nj - cj + 0.5)*dy_sinth_sinphi ...
            + (uk-ck)*dz_costh;
139     exp_jk_rpr = exp(j*k*rpr);

141     Ntheta = Ntheta ...
            + (cjxzp(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_cosphi ...
143         + cjyzp(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_sinphi) ...
            .*exp_jk_rpr;
145     Ltheta = Ltheta ...
            + (cmxzp(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_cosphi ...
147         + cmyzp(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_sinphi) ...
            .*exp_jk_rpr;
149     Nphi = Nphi + (-cjxzp(mi,ni-li+1,nj-lj+1,1) ...
            .*dx_dy_sinphi+cjyzp(mi,ni-li+1,nj-lj+1,1).*dx_dy_cosphi)...
151         .*exp_jk_rpr;
        Lphi = Lphi + (-cmxzp(mi,ni-li+1,nj-lj+1,1) ...
153         .*dx_dy_sinphi+cmyzp(mi,ni-li+1,nj-lj+1,1).*dx_dy_cosphi)...
            .*exp_jk_rpr;
155
        % for -az direction
157
        rpr = (ni-ci+0.5)*dx_sinth_cosphi ...
159         + (nj - cj + 0.5)*dy_sinth_sinphi ...
            + (lk-ck)*dz_costh;
161     exp_jk_rpr = exp(j*k*rpr);

163     Ntheta = Ntheta ...
            + (cjxzn(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_cosphi ...
165         + cjyzn(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_sinphi)...
            .*exp_jk_rpr;
167     Ltheta = Ltheta ...
            + (cmxzn(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_cosphi ...
169         + cmyzn(mi,ni-li+1,nj-lj+1,1).*dx_dy_costh_sinphi) ...
            .*exp_jk_rpr;
171     Nphi = Nphi + (-cjxzn(mi,ni-li+1,nj-lj+1,1) ...
            .*dx_dy_sinphi+cjyzn(mi,ni-li+1,nj-lj+1,1).*dx_dy_cosphi)...
173         .*exp_jk_rpr;
        Lphi = Lphi + (-cmxzn(mi,ni-li+1,nj-lj+1,1) ...
175         .*dx_dy_sinphi+cmyzn(mi,ni-li+1,nj-lj+1,1).*dx_dy_cosphi)...
            .*exp_jk_rpr;
177     end
    end
179 % calculate directivity
    farfield_dataTheta(mi,:)=(k^2./(8*pi*eta_0*radiated_power(mi))) ...
181     .* (abs(Lphi+eta_0*Ntheta).^2);
    farfield_dataPhi(mi,:)  =(k^2./(8*pi*eta_0*radiated_power(mi))) ...
183     .* (abs(Ltheta-eta_0*Nphi).^2);
end
```