CHAPTER 11

Scattered field formulation

As discussed before, sources are the necessary components of a finite-difference time-domain (FDTD) simulation, and their types vary depending on the type of the problem under consideration. Usually sources are of two types: (1) *near-zone sources*, such as the voltage and current sources described in Chapter 4; and (2) *far-zone sources*, such as the incident fields in scattering problems. In the previous chapters we demonstrated several examples utilizing near-zone sources. In this chapter we present *scattered field formulation*, one of the techniques that integrates far-zone sources into the FDTD method.

11.1 Scattered field basic equations

Far-zone sources are the fields generated somewhere outside the FDTD problem space that illuminate the objects in the problem space. Therefore, they are the *incident fields* exciting the FDTD problem space. The incident field exists in a problem space in which there are no scatterers. Therefore, they are the fields that can be described by analytical expressions and would satisfy Maxwell's curl equations where the problem space medium is free space, such that

$$\nabla \times \vec{H}_{inc} = \varepsilon_0 \frac{\partial \vec{E}_{inc}}{\partial t}, \tag{11.1a}$$

$$\nabla \times \vec{E}_{inc} = -\mu_0 \frac{\partial \vec{H}_{inc}}{\partial t}. \tag{11.1b}$$

The most common type of incident field is the plane wave. The expressions for the field components generated by plane waves are discussed in a subsequent section.

When the incident field illuminates the objects in an FDTD problem space, *scattered fields* are generated and thus need to be calculated. The scattered field formulation is one of the simplest techniques that can be used for calculating scattered fields.

The fields in a problem space are referred to in general as *total fields*. The total fields satisfy Maxwell's curl equations for a general medium, such that

$$\nabla \times \vec{H}_{tot} = \varepsilon \frac{\partial \vec{E}_{tot}}{\partial t} + \sigma^e \vec{E}_{tot}, \tag{11.2a}$$

$$\nabla \times \vec{E}_{tot} = -\mu \frac{\partial \vec{H}_{tot}}{\partial t} - \sigma^m \vec{H}_{tot}. \tag{11.2b}$$

Then the scattered fields are defined as the difference between the total fields and incident fields. Therefore, one can write

$$E_{tot} = E_{inc} + E_{scat}, \quad \text{and} \quad H_{tot} = H_{inc} + H_{scat}, \tag{11.3}$$

where E_{scat} and H_{scat} are the scattered electric and magnetic fields. In light of (11.3), (11.2) can be rewritten in terms of incident and scattered field terms as

$$\nabla \times \vec{H}_{scat} + \nabla \times \vec{H}_{inc} = \varepsilon \frac{\partial \vec{E}_{scat}}{\partial t} + \varepsilon \frac{\partial \vec{E}_{inc}}{\partial t} + \sigma^e \vec{E}_{scat} + \sigma^e \vec{E}_{inc}, \tag{11.4a}$$

$$\nabla \times \vec{E}_{scat} + \nabla \times \vec{E}_{inc} = -\mu \frac{\partial \vec{H}_{scat}}{\partial t} - \mu \frac{\partial \vec{H}_{inc}}{\partial t} - \sigma^m \vec{H}_{scat} - \sigma^m \vec{H}_{inc}. \tag{11.4b}$$

The curls of the incident fields in (11.4) can be replaced by the time-derivative terms from (11.1), which yields

$$\nabla \times \vec{H}_{scat} + \varepsilon_0 \frac{\partial \vec{E}_{inc}}{\partial t} = \varepsilon \frac{\partial \vec{E}_{scat}}{\partial t} + \varepsilon \frac{\partial \vec{E}_{inc}}{\partial t} + \sigma^e \vec{E}_{scat} + \sigma^e \vec{E}_{inc}, \tag{11.5a}$$

$$\nabla \times \vec{E}_{scat} - \mu_0 \frac{\partial \vec{H}_{inc}}{\partial t} = -\mu \frac{\partial \vec{H}_{scat}}{\partial t} - \mu \frac{\partial \vec{H}_{inc}}{\partial t} - \sigma^m \vec{H}_{scat} - \sigma^m \vec{H}_{inc}. \tag{11.5b}$$

After rearranging the terms in (11.5) one can obtain

$$\varepsilon \frac{\partial \vec{E}_{scat}}{\partial t} + \sigma^e \vec{E}_{scat} = \nabla \times \vec{H}_{scat} + (\varepsilon_0 - \varepsilon) \frac{\partial \vec{E}_{inc}}{\partial t} - \sigma^e \vec{E}_{inc}, \tag{11.6a}$$

$$\mu \frac{\partial \vec{H}_{scat}}{\partial t} + \sigma^m \vec{H}_{scat} = -\nabla \times \vec{E}_{scat} + (\mu_0 - \mu) \frac{\partial \vec{H}_{inc}}{\partial t} - \sigma^m \vec{H}_{inc}. \tag{11.6b}$$

At this point the derivatives in (11.6) can be represented by central finite difference approximations, and updating equations can be obtained for the scattered field formulation.

11.2 The scattered field updating equations

After applying the central finite difference approximation, (11.6a) can be expressed for the x component as

$$\begin{aligned}
&\varepsilon_x(i,j,k) \frac{E^{n+1}_{scat,x}(i,j,k) - E^{n}_{scat,x}(i,j,k)}{\Delta t} + \sigma^e_x(i,j,k) \frac{E^{n+1}_{scat,x}(i,j,k) + E^{n}_{scat,x}(i,j,k)}{2} \\
&= \frac{H^{n+\frac{1}{2}}_{scat,z}(i,j,k) - H^{n+\frac{1}{2}}_{scat,x}(i,j-1,k)}{\Delta y} - \frac{H^{n+\frac{1}{2}}_{scat,y}(i,j,k) - H^{n+\frac{1}{2}}_{scat,y}(i,j,k-1)}{\Delta z} \\
&\quad + (\varepsilon_0 - \varepsilon_x(i,j,k)) \frac{E^{n+1}_{inc,x}(i,j,k) + E^{n}_{inc,x}(i,j,k)}{\Delta t} \\
&\quad - \sigma^e_x(i,j,k) \frac{E^{n+1}_{inc,x}(i,j,k) + E^{n}_{inc,x}(i,j,k)}{2},
\end{aligned} \tag{11.7}$$

Equation (11.7) can be arranged such that the new value of the scattered electric field component $E^{n+1}_{scat,x}(i,j,k)$ is calculated using other terms as

$$\begin{aligned} E^{n+1}_{scat,x}(i,j,k) &= C_{exe}(i,j,k) \times E^{n}_{scat,x}(i,j,k) \\ &+ C_{exhz}(i,j,k) \times \left[H^{n+\frac{1}{2}}_{scat,z}(i,j,k) - H^{n+\frac{1}{2}}_{scat,z}(i,j-1,k)\right] \\ &+ C_{exhy}(i,j,k) \times \left[H^{n+\frac{1}{2}}_{scat,y}(i,j,k) - H^{n+\frac{1}{2}}_{scat,y}(i,j,k-1)\right] \\ &+ C_{exeic}(i,j,k) \times E^{n+1}_{inc,x}(i,j,k) + C_{exeip}(i,j,k) \times E^{n}_{inc,x}(i,j,k), \end{aligned} \tag{11.8}$$

where

$$\begin{aligned} C_{exe}(i,j,k) &= \frac{2\varepsilon_x(i,j,k) - \sigma^e_x(i,j,k)\Delta t}{2\varepsilon_x(i,j,k) + \sigma^e_x(i,j,k)\Delta t}, \\ C_{exhz}(i,j,k) &= \frac{2\Delta t}{\Delta y\left(2\varepsilon_x(i,j,k) + \sigma^e_x(i,j,k)\Delta t\right)}, \\ C_{exhy}(i,j,k) &= -\frac{2\Delta t}{\Delta z\left(2\varepsilon_x(i,j,k) + \sigma^e_x(i,j,k)\Delta t\right)}, \\ C_{exeic}(i,j,k) &= \frac{2(\varepsilon_0 - \varepsilon_x(i,j,k)) - \sigma^e_x(i,j,k)\Delta t}{2\varepsilon_x(i,j,k) + \sigma^e_x(i,j,k)\Delta t}, \\ C_{exeip}(i,j,k) &= -\frac{2(\varepsilon_0 - \varepsilon_x(i,j,k)) + \sigma^e_x(i,j,k)\Delta t}{2\varepsilon_x(i,j,k) + \sigma^e_x(i,j,k)\Delta t}. \end{aligned}$$

In this equation, the *ic* term in the subscript of the coefficient C_{exeic} indicates that this coefficient is multiplied with the *current* value of the *incident* field component, whereas the *ip* term in the subscript of the coefficient C_{exeip} indicates that this coefficient is multiplied with the *previous* value of the *incident* field component.

Similarly the updating equation can be written for $E^{n+1}_{scat,y}(i,j,k)$ as

$$\begin{aligned} E^{n+1}_{scat,y}(i,j,k) &= C_{eye}(i,j,k) \times E^{n}_{scat,y}(i,j,k) \\ &+ C_{eyhx}(i,j,k) \times \left[H^{n+\frac{1}{2}}_{scat,x}(i,j,k) - H^{n+\frac{1}{2}}_{scat,x}(i,j,k-1)\right] \\ &+ C_{eyhz}(i,j,k) \times \left[H^{n+\frac{1}{2}}_{scat,z}(i,j,k) - H^{n+\frac{1}{2}}_{scat,z}(i-1,j,k)\right] \\ &+ C_{eyeic}(i,j,k) \times E^{n+1}_{inc,y}(i,j,k) + C_{eyeip}(i,j,k) \times E^{n}_{inc,y}(i,j,k), \end{aligned} \tag{11.9}$$

where

$$\begin{aligned} C_{eye}(i,j,k) &= \frac{2\varepsilon_y(i,j,k) - \sigma^e_y(i,j,k)\Delta t}{2\varepsilon_y(i,j,k) + \sigma^e_y(i,j,k)\Delta t}, \\ C_{eyhx}(i,j,k) &= \frac{2\Delta t}{\Delta z\left(2\varepsilon_y(i,j,k) + \sigma^e_y(i,j,k)\Delta t\right)}, \end{aligned}$$

$$C_{eyhz}(i,j,k) = -\frac{2\Delta t}{\Delta x\left(2\varepsilon_y(i,j,k) + \sigma_y^e(i,j,k)\Delta t\right)},$$

$$C_{eyeic}(i,j,k) = \frac{2\left(\varepsilon_0 - \varepsilon_y(i,j,k)\right) - \sigma_y^e(i,j,k)\Delta t}{2\varepsilon_y(i,j,k) + \sigma_y^e(i,j,k)\Delta t},$$

$$C_{eyeip}(i,j,k) = -\frac{2\left(\varepsilon_0 - \varepsilon_y(i,j,k)\right) + \sigma_y^e(i,j,k)\Delta t}{2\varepsilon_y(i,j,k) + \sigma_y^e(i,j,k)\Delta t}.$$

The updating equation can be written for $E^{n+1}_{scat,z}(i,j,k)$ as

$$\begin{aligned} E^{n+1}_{scat,z}(i,j,k) = {} & C_{eze}(i,j,k) \times E^{n}_{scat,z}(i,j,k) \\ & + C_{ezhy}(i,j,k) \times \left[H^{n+\frac{1}{2}}_{scat,y}(i,j,k) - H^{n+\frac{1}{2}}_{scat,y}(i,j,k-1)\right] \\ & + C_{ezhx}(i,j,k) \times \left[H^{n+\frac{1}{2}}_{scat,x}(i,j,k) - H^{n+\frac{1}{2}}_{scat,x}(i-1,j,k)\right] \\ & + C_{ezeic}(i,j,k) \times E^{n+1}_{inc,z}(i,j,k) + C_{ezeip}(i,j,k) \times E^{n}_{inc,z}(i,j,k), \end{aligned} \tag{11.10}$$

where

$$C_{eze}(i,j,k) = \frac{2\varepsilon_z(i,j,k) - \sigma_z^e(i,j,k)\Delta t}{2\varepsilon_z(i,j,k) + \sigma_z^e(i,j,k)\Delta t},$$

$$C_{ezhy}(i,j,k) = \frac{2\Delta t}{\Delta x\left(2\varepsilon_z(i,j,k) + \sigma_z^e(i,j,k)\Delta t\right)},$$

$$C_{ezhx}(i,j,k) = -\frac{2\Delta t}{\Delta y\left(2\varepsilon_z(i,j,k) + \sigma_z^e(i,j,k)\Delta t\right)},$$

$$C_{ezeic}(i,j,k) = \frac{2(\varepsilon_0 - \varepsilon_z(i,j,k)) - \sigma_z^e(i,j,k)\Delta t}{2\varepsilon_z(i,j,k) + \sigma_z^e(i,j,k)\Delta t},$$

$$C_{eyeip}(i,j,k) = -\frac{2(\varepsilon_0 - \varepsilon_z(i,j,k)) + \sigma_z^e(i,j,k)\Delta t}{2\varepsilon_z(i,j,k) + \sigma_z^e(i,j,k)\Delta t}.$$

Following a similar procedure, the updating equations for the magnetic field components can be obtained as follows. The updating equation for $H^{n+\frac{1}{2}}_{scat,x}(i,j,k)$ is

$$\begin{aligned} H^{n+\frac{1}{2}}_{scat,x}(i,j,k) = {} & C_{hxh}(i,j,k) \times H^{n-\frac{1}{2}}_{scat,x}(i,j,k) \\ & + C_{hxez}(i,j,k) \times \left[E^{n}_{scat,z}(i,j+1,k) - E^{n}_{scat,z}(i,j,k)\right] \\ & + C_{hxey}(i,j,k) \times \left[E^{n}_{scat,y}(i,j,k+1) - E^{n}_{scat,y}(i,j,k)\right] \\ & + C_{hxhic}(i,j,k) \times H^{n+\frac{1}{2}}_{inc,x}(i,j,k) + C_{hxhip}(i,j,k) \times H^{n-\frac{1}{2}}_{inc,x}(i,j,k) \end{aligned} \tag{11.11}$$

where

$$\begin{aligned}
C_{hxh}(i,j,k) &= \frac{2\mu_x(i,j,k) - \sigma_x^m(i,j,k)\Delta t}{2\mu_x(i,j,k) + \sigma_x^m(i,j,k)\Delta t},\\
C_{hxez}(i,j,k) &= -\frac{2\Delta t}{\Delta y\big(2\mu_x(i,j,k) + \sigma_x^m(i,j,k)\Delta t\big)},\\
C_{hxey}(i,j,k) &= \frac{2\Delta t}{\Delta z\big(2\mu_x(i,j,k) + \sigma_x^m(i,j,k)\Delta t\big)},\\
C_{hxhic}(i,j,k) &= \frac{2(\mu_0 - \mu_x(i,j,k)) - \sigma_x^m(i,j,k)\Delta t}{2\mu_x(i,j,k) + \sigma_x^m(i,j,k)\Delta t},\\
C_{hxhip}(i,j,k) &= -\frac{2(\mu_0 - \mu_x(i,j,k)) + \sigma_x^m(i,j,k)\Delta t}{2\mu_x(i,j,k) + \sigma_x^m(i,j,k)\Delta t}.
\end{aligned}$$

The updating equation for $H_{scat,y}^{n+\frac{1}{2}}(i,j,k)$ is

$$\begin{aligned}
H_{scat,y}^{n+\frac{1}{2}}(i,j,k) = C_{hyh}(i,j,k) \times H_{scat,y}^{n-\frac{1}{2}}(i,j,k)&\\
+ C_{hyex}(i,j,k) \times \Big[E_{scat,x}^{n}(i,j+1,k) - E_{scat,x}^{n}(i,j,k)\Big]&\\
+ C_{hyez}(i,j,k) \times \Big[E_{scat,z}^{n}(i,j,k+1) - E_{scat,z}^{n}(i,j,k)\Big]&\\
+ C_{hyhic}(i,j,k) \times H_{inc,y}^{n+\frac{1}{2}}(i,j,k) + C_{hyhip}(i,j,k) \times H_{inc,y}^{n-\frac{1}{2}}(i,j,k),&
\end{aligned} \tag{11.12}$$

where

$$\begin{aligned}
C_{hyh}(i,j,k) &= \frac{2\mu_y(i,j,k) - \sigma_y^m(i,j,k)\Delta t}{2\mu_y(i,j,k) + \sigma_y^m(i,j,k)\Delta t},\\
C_{hyex}(i,j,k) &= -\frac{2\Delta t}{\Delta z\Big(2\mu_y(i,j,k) + \sigma_y^m(i,j,k)\Delta t\Big)},\\
C_{hyez}(i,j,k) &= \frac{2\Delta t}{\Delta x\Big(2\mu_y(i,j,k) + \sigma_y^m(i,j,k)\Delta t\Big)},\\
C_{hyhic}(i,j,k) &= \frac{2\Big(\mu_0 - \mu_y(i,j,k)\Big) - \sigma_y^m(i,j,k)\Delta t}{2\mu_y(i,j,k) + \sigma_y^m(i,j,k)\Delta t}.\\
C_{hyhip}(i,j,k) &= -\frac{2\Big(\mu_0 - \mu_y(i,j,k)\Big) + \sigma_y^m(i,j,k)\Delta t}{2\mu_y(i,j,k) + \sigma_y^m(i,j,k)\Delta t},
\end{aligned}$$

Finally, the updating equation for $H^{n+\frac{1}{2}}_{scat,z}(i,j,k)$ is

$$\begin{aligned} H^{n+\frac{1}{2}}_{scat,z}(i,j,k) = {} & C_{hzh}(i,j,k) \times H^{n-\frac{1}{2}}_{scat,z}(i,j,k) \\ & + C_{hzey}(i,j,k) \times \left[E^{n}_{scat,y}(i,j+1,k) - E^{n}_{scat,y}(i,j,k)\right] \\ & + C_{hzex}(i,j,k) \times \left[E^{n}_{scat,x}(i,j,k+1), -E^{n}_{scat,x}(i,j,k)\right] \\ & + C_{hzhic}(i,j,k) \times H^{n+\frac{1}{2}}_{inc,z}(i,j,k) + C_{hzhip}(i,j,k) \times H^{n-\frac{1}{2}}_{inc,z}(i,j,k) \end{aligned} \tag{11.13}$$

where

$$\begin{aligned} C_{hzh}(i,j,k) &= \frac{2\mu_z(i,j,k) - \sigma^m_z(i,j,k)\Delta t}{2\mu_z(i,j,k) + \sigma^m_z(i,j,k)\Delta t}, \\ C_{hzey}(i,j,k) &= -\frac{2\Delta t}{\Delta x\left(2\mu_z(i,j,k) + \sigma^m_z(i,j,k)\Delta t\right)}, \\ C_{hzex}(i,j,k) &= \frac{2\Delta t}{\Delta y\left(2\mu_z(i,j,k) + \sigma^m_z(i,j,k)\Delta t\right)}, \\ C_{hzhic}(i,j,k) &= \frac{2(\mu_0 - \mu_z(i,j,k)) - \sigma^m_z(i,j,k)\Delta t}{2\mu_z(i,j,k) + \sigma^m_z(i,j,k)\Delta t}, \\ C_{hzhip}(i,j,k) &= -\frac{2(\mu_0 - \mu_z(i,j,k)) + \sigma^m_z(i,j,k)\Delta t}{2\mu_z(i,j,k) + \sigma^m_z(i,j,k)\Delta t}. \end{aligned}$$

Equations (11.8)–(11.13) are the updating equations with corresponding definitions for associated coefficients for the scattered field formulation. Comparing these equations with the set of general updating (1.26)–(1.31) one can realize that the forms of the equations and expressions of the coefficients are the same except that (11.8)–(11.13) include additional incident field terms.

11.3 Expressions for the incident plane waves

The aforementioned scattered field updating equations are derived to calculate scattered electric fields in a problem space in response to an incident field. The incident field in general can be any far-zone source that can be expressed by analytical expressions. In this context, incident plane waves are the most commonly used type of incident field. This section discusses the incident plane wave field expressions.

Figure 11.1 illustrates an incident plane wave traveling in the direction denoted by a unit vector $\hat{k}$. The incident electric field in general may have a θ component and a ϕ component when expressed in a spherical coordinate system denoting its polarization. The incident electric field can be expressed for a given point denoted by a position vector $\vec{r}$ as

$$\vec{E}_{inc} = \left(E_\theta\hat{\theta} + E_\phi\hat{\phi}\right) f\left(t - \frac{1}{c}\hat{k}\cdot\vec{r}\right), \tag{11.14}$$

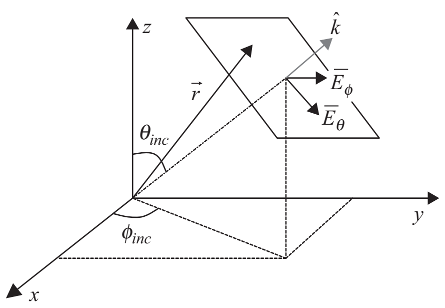

Figure 11.1 An incident plane wave.

where f is a function determining the waveform of the incident electric field. This equation can be rewritten by allowing a time delay t_0 and spatial shift l_0 as

$$\vec{E}_{inc} = \left(E_\theta \hat{\theta} + E_\phi \hat{\phi}\right) f\left((t - t_0) - \frac{1}{c}\left(\hat{k} \cdot \vec{r} - l_0\right)\right). \tag{11.15}$$

The vectors $\vec{r}$ and $\hat{k}$ can be written in Cartesian coordinates as

$$\vec{r} = \hat{x}x + \hat{y}y + \hat{z}z, \quad \hat{k} = \hat{x}\sin\theta_{inc}\cos\phi_{inc} + \hat{y}\sin\theta_{inc}\sin\phi_{inc} + \hat{z}\cos\theta_{inc}, \tag{11.16}$$

where θ_{inc} and ϕ_{inc} are the angles indicating the direction of propagation of the incident plane wave as shown in Figure 11.1. The components of the incident electric field E_θ and E_ϕ can also be expressed in Cartesian coordinates. Therefore, after using (11.16) in (11.15) and applying the spherical to Cartesian coordinates transformation, one can obtain the components of the incident electric field for any point (x, y, z) in space as

$$E_{inc,x} = (E_\theta \cos\theta_{inc}\cos\phi_{inc} - E_\phi \sin\phi_{inc}) f_{wf}, \tag{11.17a}$$

$$E_{inc,y} = (E_\theta \cos\theta_{inc}\sin\phi_{inc} + E_\phi \cos\phi_{inc}) f_{wf}, \tag{11.17b}$$

$$E_{inc,z} = (-E_\theta \sin\theta_{inc}) f_{wf}, \tag{11.17c}$$

where f_{wf} is given by

$$f_{wf} = f\left((t - t_0) - \frac{1}{c}(x\sin\theta_{inc}\, cos\phi_{inc} + y\sin\theta_{inc}\sin\phi_{inc} + z\cos\theta_{inc} - l_0)\right). \tag{11.18}$$

Once the expressions for the incident electric field components are obtained, the expressions for magnetic field components can be obtained using

$$\vec{H}_{inc} = \frac{1}{\eta_0}\hat{k} \times \vec{E}_{inc},$$

where n_0 is the intrinsic impedance of free space, and the $\times$ denotes the cross-product operator. Then the magnetic field components are

$$H_{inc,x} = \frac{-1}{\eta_0}(E_\phi \cos\theta_{inc}\cos\phi_{inc} + E_\theta \sin\phi_{inc}) f_{wf}, \tag{11.19a}$$

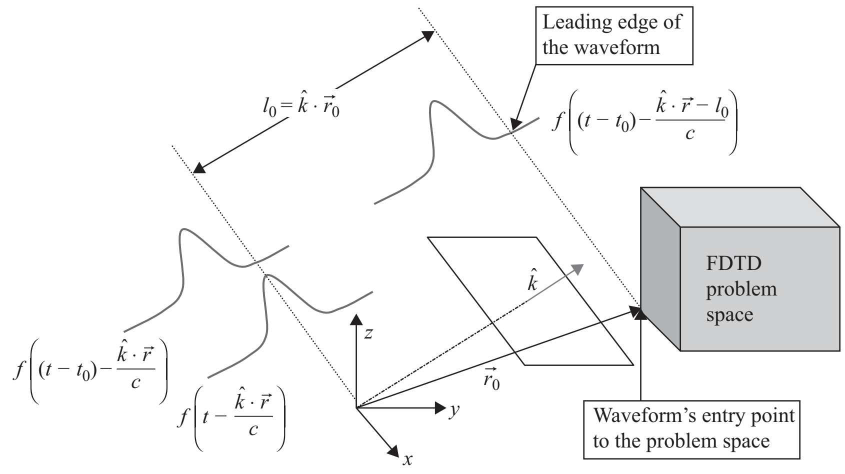
Leading edge of the waveform
$l_0 = \hat{k} \cdot \vec{r}_0$
$f\left((t - t_0) - \frac{\hat{k} \cdot \vec{r} - l_0}{c}\right)$
FDTD problem space
$\hat{k}$
$f\left((t - t_0) - \frac{\hat{k} \cdot \vec{r}}{c}\right)$
z
$\vec{r}_0$
$f\left(t - \frac{\hat{k} \cdot \vec{r}}{c}\right)$
y
x
Waveform's entry point to the problem space

Figure 11.2 An incident plane wave delayed in time and shifted in space.

$$H_{inc,y} = \frac{-1}{\eta_0}(E_\phi \cos\theta_{inc} \sin\phi_{inc} - E_\theta \cos\phi_{inc})f_{wf}, \tag{11.19b}$$

$$H_{inc,z} = \frac{1}{\eta_0}(E_\phi \sin\theta_{inc})f_{wf}. \tag{11.19c}$$

Equations (11.17)–(11.19) express an incident field with a polarization determined by $(E_\theta\hat{\theta} + E_\phi\hat{\phi})$, propagating in a direction determined by the angles θ_{inc} and ϕ_{inc} and having a waveform described by f_{wf}. In general, the waveform can be a function that has the desired bandwidth characteristics as discussed in Chapter 5. However, there are two additional parameters in (11.18) we have not discussed yet: t_0 and l_0. These parameters are used to shift the given waveform in time and space such that when the FDTD iterations start the incident field in the problem space is zero, and as time proceeds the incident field propagates into the FDTD problem space.

Consider Figure 11.2, which illustrates a plane wave with a Gaussian waveform propagating in a direction $\hat{k}$ toward an FDTD problem space. This waveform is expressed by a function

$$f\left(t - \frac{1}{c}\left(\hat{k} \cdot \vec{r}\right)\right),$$

where the maximum of the waveform appears on a plane including the origin and normal to $\hat{k}$ at $t = 0$. Clearly the leading edge of this waveform is closer to the FDTD problem space. A time delay t_0 can be applied to this function such that

$$f\left((t - t_0) - \frac{1}{c}\left(\hat{k} \cdot \vec{r}\right)\right),$$

and the leading edge of the waveform coincides with the origin. The value of the waveform can be calculated as explained in Chapter 5. After applying the time delay, there is a distance between the leading edge of the waveform and the problem space that needs to be accounted for; otherwise, it may take considerable time after the simulation is started until the waveform enters to the problem space. The distance between the waveform and the problem space can be found as $l_0 = \hat{k} \cdot \vec{r}_0$, where $\vec{r}_0$ is the position vector indicating the closest point of the problem space to the waveform. A problem space has eight corners, and one of these corners will be closest to the waveform approaching from an arbitrary direction. If $\vec{r}_0$ indicates the corner points of the problem space, then the minimum distance for l_0 can be calculated by $l_0 = min\ [\hat{k} \cdot \vec{r}_0]$. After l_0 is determined, the waveform equation can be modified as

$$f\left((t - t_0) - \frac{1}{c}\left(\hat{k} \cdot \vec{r} - l_0\right)\right),$$

which leads to (11.18).

In some scattering applications, a scatterer is placed at the origin and the scattered field due to an incident plane wave is calculated. The direction of propagation of the plane wave must be defined appropriately in such applications. Consider Figure 11.3, where two plane waves are illustrated. Although these two plane waves are traveling in the same direction, one of them is illustrated as traveling toward the origin while the other one is traveling away from the origin. The equations given in this section are derived based on the wave traveling away from the origin; the angles determining the propagation, θ_{inc} and ϕ_{inc}, and amplitudes of the incident field components E_θ and E_ϕ are as shown in this figure. If it is desired to define the angles and field components based on the wave traveling toward the origin such as $\theta'_{inc}, \phi'_{inc}, E'_\theta$, and E'_ϕ, then the following conversions need to be employed to use the aforementioned equations as is:

$$\theta_{inc} = \pi - \theta'_{inc}, \tag{11.20a}$$

$$\phi_{inc} = \pi + \phi'_{inc}, \tag{11.20b}$$

$$E_\theta = E'_\theta, \tag{11.20c}$$

$$E_\phi = -E'_\phi. \tag{11.20d}$$

One of the types of results that can be obtained from the scattered field due to an incident plane wave is the radar cross-section (RCS) of a scatterer. The near-field to far-field NF-FF transformation algorithm discussed in Chapter 9 is used for this purpose. In this case the scattered near fields calculated in a problem space are captured on an imaginary surface

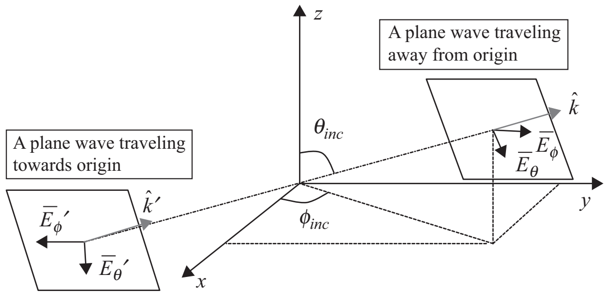

Figure 11.3 Two incident plane waves: one traveling toward the origin and the other traveling away from the origin.

enclosing the scatterer to determine the equivalent fictitious surface currents. These currents are transformed to the frequency domain while being captured. After the time-marching loop is completed, the far-field terms L_θ, L_ϕ, N_θ, and N_ϕ are calculated following the procedure described in Chapter 9.

At this point the components of the bistatic RCS can be calculated using

$$RCS_\theta = \frac{k^2}{8\pi\eta_0 P_{inc}} |L_\phi + \eta_0 N_\theta|^2 \tag{11.21a}$$

$$RCS_\phi = \frac{k^2}{8\pi\eta_0 P_{inc}} |L_\theta - \eta_0 N_\phi|^2, \tag{11.21b}$$

which is very similar to (9.39). The only difference is that P_{rad} is replaced by P_{inc}, which is the power carried by the incident plane wave. The P_{inc} can be calculated as

$$P_{inc} = \frac{1}{2\eta_0} |E_{inc}(\omega)|^2, \tag{11.22}$$

where $E_{inc}(\omega)$ is the discrete Fourier transform of the incident electric field waveform at the frequency for which the RCS calculation is sought.

11.4 MATLAB® implementation of the scattered field formulation

The scattered field formulation is presented in the previous sections. In this section the implementation of the scattered field formulation is discussed.

11.4.1 Definition of the incident plane wave

For a scattering problem it is necessary to define an incident field as the source. In Section 11.3 the equations necessary to construct a plane wave are presented, and the plane wave is used as the incident field in this discussion. In the FDTD program an incident plane wave is defined in the subroutine ***define_sources_and_lumped_elements*** as shown in Listing 11.1. In the given code first a new empty structure, **incident_plane_wave**, is defined. Then the parameters of the incident plane wave are defined as fields of this structure. Here **E_theta** denotes the maximum amplitude of the θ component, whereas **E_phi** denotes the maximum amplitude of the ϕ component of the electric field waveform. The fields **theta_incident** and **phi_incident** denote the angles indicating the direction of propagation, θ_{inc} and ϕ_{inc}, of the incident plane wave. It should be noted here that these parameters are based on the convention that the plane wave is propagating in a direction away from the origin rather than toward the origin. If the parameters are readily available for the convention wave traveling toward the origin, then the necessary transformation needs to be performed based on (11.20). Finally, the waveform of the incident plane wave need to be defined. The definition of the waveform for the incident plane wave follows the same procedure as the voltage source or the current source waveform. The types of waveforms are defined as structure **waveforms**, and the fields **waveform_type** and **waveform_index** are defined under **incident_plane_wave**, which point to a specific waveform defined under **waveforms**.

Listing 11.1 define_sources_and_lumped_elements

```
disp('defining␣sources␣and␣lumped␣element␣components');

voltage_sources = [];
current_sources = [];
diodes = [];
resistors = [];
inductors = [];
capacitors = [];
incident_plane_wave = [];

% define source waveform types and parameters
waveforms.gaussian(1).number_of_cells_per_wavelength = 0;
waveforms.gaussian(2).number_of_cells_per_wavelength = 15;

% Define incident plane wave, angles are in degrees
incident_plane_wave.E_theta = 1;
incident_plane_wave.E_phi = 0;
incident_plane_wave.theta_incident = 0;
incident_plane_wave.phi_incident = 0;
incident_plane_wave.waveform_type = 'gaussian';
incident_plane_wave.waveform_index = 1;
```

11.4.2 Initialization of the incident fields

The next step is the initialization of the incident fields and the updating coefficients. The initialization of the incident fields is performed in the subroutine ***initialize_sources_and_lumped_elements***. The section of the code listed in Listing 11.2 is added to this subroutine.

Listing 11.2 initialize_sources_and_lumped_elements

```
% initialize incident plane wave
if isfield(incident_plane_wave,'E_theta')
    incident_plane_wave.enabled = true;
else
    incident_plane_wave.enabled = false;
end

if incident_plane_wave.enabled
    % create incident field arrays for current time step
    Hxic = zeros(nxp1,ny,nz);
    Hyic = zeros(nx,nyp1,nz);
    Hzic = zeros(nx,ny,nzp1);
    Exic = zeros(nx,nyp1,nzp1);
    Eyic = zeros(nxp1,ny,nzp1);
    Ezic = zeros(nxp1,nyp1,nz);
    % create incident field arrays for previous time step
    Hxip = zeros(nxp1,ny,nz);
    Hyip = zeros(nx,nyp1,nz);
    Hzip = zeros(nx,ny,nzp1);
```

```
Exip = zeros(nx,nyp1,nzp1);
Eyip = zeros(nxp1,ny,nzp1);
Ezip = zeros(nxp1,nyp1,nz);

% calculate the amplitude factors for field components
theta_incident = incident_plane_wave.theta_incident*pi/180;
phi_incident = incident_plane_wave.phi_incident*pi/180;
E_theta = incident_plane_wave.E_theta;
E_phi = incident_plane_wave.E_phi;
eta_0 = sqrt(mu_0/eps_0);
Exi0 = E_theta * cos(theta_incident) * cos(phi_incident) ...
    - E_phi * sin(phi_incident);
Eyi0 = E_theta * cos(theta_incident) * sin(phi_incident) ...
    + E_phi * cos(phi_incident);
Ezi0 = -E_theta * sin(theta_incident);
Hxi0 = (-1/eta_0)*(E_phi * cos(theta_incident) ...
    * cos(phi_incident) + E_theta * sin(phi_incident));
Hyi0 = (-1/eta_0)*(E_phi * cos(theta_incident) ...
    * sin(phi_incident) - E_theta * cos(phi_incident));
Hzi0 = (1/eta_0)*(E_phi * sin(theta_incident));

% Create position arrays indicating the coordinates of the nodes
x_pos = zeros(nxp1,nyp1,nzp1);
y_pos = zeros(nxp1,nyp1,nzp1);
z_pos = zeros(nxp1,nyp1,nzp1);
for ind = 1:nxp1
    x_pos(ind,:,:) = (ind - 1) * dx + fdtd_domain.min_x;
end
for ind = 1:nyp1
    y_pos(:,ind,:) = (ind - 1) * dy + fdtd_domain.min_y;
end
for ind = 1:nzp1
    z_pos(:,:,ind) = (ind - 1) * dz + fdtd_domain.min_z;
end

% calculate spatial shift, l_0, required for incident plane wave
r0 =[fdtd_domain.min_x fdtd_domain.min_y fdtd_domain.min_z;
fdtd_domain.min_x fdtd_domain.min_y fdtd_domain.max_z;
fdtd_domain.min_x fdtd_domain.max_y fdtd_domain.min_z;
fdtd_domain.min_x fdtd_domain.max_y fdtd_domain.max_z;
fdtd_domain.max_x fdtd_domain.min_y fdtd_domain.min_z;
fdtd_domain.max_x fdtd_domain.min_y fdtd_domain.max_z;
fdtd_domain.max_x fdtd_domain.max_y fdtd_domain.min_z;
fdtd_domain.max_x fdtd_domain.max_y fdtd_domain.max_z;];

k_vec_x =  sin(theta_incident)*cos(phi_incident);
k_vec_y =  sin(theta_incident)*sin(phi_incident);
k_vec_z =  cos(theta_incident);

k_dot_r0 = k_vec_x * r0(:,1) ...
    + k_vec_y * r0(:,2) ...
    + k_vec_z * r0(:,3);
    l_0 = min(k_dot_r0)/c;

    % calculate k.r for every field component
    k_dot_r_ex = ((x_pos(1:nx,1:nyp1,1:nzp1)+dx/2) * k_vec_x ...
        + y_pos(1:nx,1:nyp1,1:nzp1) * k_vec_y ...
        + z_pos(1:nx,1:nyp1,1:nzp1) * k_vec_z)/c;

    k_dot_r_ey = (x_pos(1:nxp1,1:ny,1:nzp1) * k_vec_x ...
        + (y_pos(1:nxp1,1:ny,1:nzp1)+dy/2) * k_vec_y ...
        + z_pos(1:nxp1,1:ny,1:nzp1) * k_vec_z)/c;
    k_dot_r_ez = (x_pos(1:nxp1,1:nyp1,1:nz) * k_vec_x ...
        + y_pos(1:nxp1,1:nyp1,1:nz) * k_vec_y ...
        + (z_pos(1:nxp1,1:nyp1,1:nz)+dz/2) * k_vec_z)/c;

    k_dot_r_hx = (x_pos(1:nxp1,1:ny,1:nz) * k_vec_x ...
        + (y_pos(1:nxp1,1:ny,1:nz)+dy/2) * k_vec_y ...
        + (z_pos(1:nxp1,1:ny,1:nz)+dz/2) * k_vec_z)/c;

    k_dot_r_hy = ((x_pos(1:nx,1:nyp1,1:nz)+dx/2) * k_vec_x ...
        + y_pos(1:nx,1:nyp1,1:nz) * k_vec_y ...
        + (z_pos(1:nx,1:nyp1,1:nz)+dz/2) * k_vec_z)/c;

    k_dot_r_hz = ((x_pos(1:nx,1:ny,1:nzp1)+dx/2) * k_vec_x ...
        + (y_pos(1:nx,1:ny,1:nzp1)+dy/2) * k_vec_y ...
        + z_pos(1:nx,1:ny,1:nzp1) * k_vec_z)/c;

    % embed spatial shift in k.r
    k_dot_r_ex = k_dot_r_ex - l_0;
    k_dot_r_ey = k_dot_r_ey - l_0;
    k_dot_r_ez = k_dot_r_ez - l_0;
    k_dot_r_hx = k_dot_r_hx - l_0;
    k_dot_r_hy = k_dot_r_hy - l_0;
    k_dot_r_hz = k_dot_r_hz - l_0;

    % store the waveform
    wt_str = incident_plane_wave.waveform_type;
    wi_str = num2str(incident_plane_wave.waveform_index);
    eval_str = ['a_waveform␣=␣waveforms.' ...
        wt_str '(' wi_str ').waveform;'];
    eval(eval_str);
    incident_plane_wave.waveform = a_waveform;

    clear x_pos y_pos z_pos;
end
```

In this code the first step is to determine whether an incident plane wave is defined. If an incident plane wave is defined, then it indicates that the FDTD simulation is running a scattering problem and, therefore, the algorithm of the scattered field formulation will be employed. Here a logical parameter **incident_plane_wave.enabled** is assigned a value "***true***" or "***false***" indicating the mode of the FDTD simulation.

The calculation of the scattered field components using the scattered field updating equation is the same as the calculation of the *total* field components using the general updating equation, except for the additional incident field terms. Therefore, the parameters corresponding to the field arrays **Ex**, **Ey**, **Ez**, **Hx**, **Hy**, and **Hz** can be assumed as scattered fields if the mode of the simulation is scattering. Then the field arrays corresponding to incident fields must be defined. The new field arrays are thus defined as **Exi**, **Eyi**, **Ezi**, **Hxi**, **Hyi**, and **Hzi** and are initialized as zeros. One should notice that the sizes of these three-dimensional arrays are the same as their corresponding scattered field arrays.

In the next step the amplitudes of the electric and magnetic field waveforms are transformed from the spherical coordinates to Cartesian coordinates based on (11.17) and (11.19), yielding **Exi0**, **Eyi0**, **Ezi0**, **Hxi0**, **Hyi0**, and **Hzi0**.

The incident field components are computed at different positions in a three-dimensional space. The coordinates of these field components can be determined with respect to positions of the nodes of the FDTD grid. Three three-dimensional arrays are constructed, each of which stores the x, y, and z coordinates of the nodes.

As discussed in Section 11.3 the waveform of the incident field should be shifted in time and space to ensure that the leading edge of the waveform enters into the problem space after the simulation has started. These shifts are indicated as t_0 and l_0, respectively. The temporal shift t_0 is determined in the subroutine ***initialize_waveforms***, which has been called in ***initialize_sources_and_lumped_elements*** before. Thus, t_0 for a specific waveform is readily available. The spatial shift, however, needs to be calculated. Eight corners determine the boundaries of the problem space, and an incident plane wave will enter to the problem space from one of these corers. The coordinates of these corners are stored in an array denoted as **r0**. The value of l_0 is calculated by $l_0 = min[\hat{k} \cdot \vec{r}_0]$, where $\hat{k}$ is the propagation vector indicating the direction of propagation. The components of the vector $\hat{k}$ are calculated based on (11.16) and are stored in the parameters **k_vec_x**, **k_vec_y**, and **k_vec_z**. Then these parameters are used together with **r0** to calculate l_0 and to store it in the parameter **1_0**.

Another term that determines the three-dimensional distribution of the waveform is $\hat{k} \cdot \vec{r}$ as shown in (11.18). Here the vector $\vec{r}$ is the position vector. In the three-dimensional case this dot product would yield three-dimensional arrays. Since all field components are located at different positions, this dot product would yield a different three-dimensional array for each component. Therefore, the dot product is performed separately for each field and is stored in the arrays **k_dot_r_ex**, **k_dot_r_ey**, **k_dot_r_ez**, **k_dot_r_hx**, **k_dot_r_hy**, and **k_dot_r_hz**. Then the spatial shift l_0 appearing in (11.18) is embedded in these arrays.

Finally, the waveform of the incident plane wave is acquired from the structure **waveforms** and is stored in the structure **incident_plane_wave**. This waveform is used only for plotting purposes after the FDTD calculation is completed. The distribution of the waveform and incident plane wave has to be recalculated at every time step and stored in the three-dimensional incident field arrays for use in the scattered field updating equations.

11.4.3 Initialization of the updating coefficients

The scattered field formulation algorithm requires a new set of updating coefficients. As discussed in Section 11.2 the updating equations for the scattered field formulation are the same as the general updating equations except that the scattered field updating equations have additional terms. These terms consist of incident fields multiplied by incident field coefficients.

Listing 11.3 initialize_incident_field_updating_coefficients

```
% initialize incident field updating coefficients

 if incident_plane_wave.enabled == false
     return;
end

% Coeffiecients updating Ex
Cexeic= (2*(1-eps_r_x)*eps_0-dt*sigma_e_x) ...
     ./(2*eps_r_x*eps_0+dt*sigma_e_x);
Cexeip=-(2*(1-eps_r_x)*eps_0+dt*sigma_e_x) ...
     ./(2*eps_r_x*eps_0+dt*sigma_e_x);

% Coeffiecients updating Ey
Ceyeic= (2*(1-eps_r_y)*eps_0-dt*sigma_e_y) ...
     ./(2*eps_r_y*eps_0+dt*sigma_e_y);
Ceyeip=-(2*(1-eps_r_y)*eps_0+dt*sigma_e_y) ...
     ./(2*eps_r_y*eps_0+dt*sigma_e_y);

% Coeffiecients updating Ez
Cezeic= (2*(1-eps_r_z)*eps_0-dt*sigma_e_z) ...
     ./(2*eps_r_z*eps_0+dt*sigma_e_z);
Cezeip=-(2*(1-eps_r_z)*eps_0+dt*sigma_e_z) ...
     ./(2*eps_r_z*eps_0+dt*sigma_e_z);

% Coeffiecients updating Hx
Chxhic= (2*(1-mu_r_x)*mu_0-dt*sigma_m_x)./(2*mu_r_x*mu_0+dt*sigma_m_x);
Chxhip=-(2*(1-mu_r_x)*mu_0+dt*sigma_m_x)./(2*mu_r_x*mu_0+dt*sigma_m_x);

% Coeffiecients updating Hy
Chyhic= (2*(1-mu_r_y)*mu_0-dt*sigma_m_y)./(2*mu_r_y*mu_0+dt*sigma_m_y);
Chyhip=-(2*(1-mu_r_y)*mu_0+dt*sigma_m_y)./(2*mu_r_y*mu_0+dt*sigma_m_y);

% Coeffiecients updating Hz
Chzhic=(2*(1-mu_r_z)*mu_0-dt*sigma_m_z)./(2*mu_r_z*mu_0+dt*sigma_m_z);
Chzhip=-(2*(1-mu_r_z)*mu_0+dt*sigma_m_z)./(2*mu_r_z*mu_0+dt*sigma_m_z);
```

The incident field coefficients are initialized in a subroutine ***initialize_incident_field_updating_coefficients***, the implementation of which is shown in Listing 11.3. This subroutine is called in ***initialize_updating_coefficients*** after all other updating coefficient subroutines are called. Here a set of new three-dimensional arrays is constructed representing the updating coefficients related to the incident field components based on (11.8)–(11.13).

11.4.4 Calculation of the scattered fields

As the iterations proceed in the FDTD time-marching algorithm, the incident plane wave enters the FDTD problem space and propagates such that at every time instant the values of the incident field components change. Therefore, the values of the incident field components need to be recalculated at every time step. A new subroutine named ***update_incident_fields*** is implemented for this purpose and is provided in Listing 11.4.

Listing 11.4 update_incident_fields

```
% update incident fields for the current time step
if incident_plane_wave.enabled == false
    return;
end

tm = current_time + dt/2;
te = current_time + dt;

% update incident fields for previous time step
Hxip = Hxic; Hyip = Hyic; Hzip = Hzic;
Exip = Exic; Eyip = Eyic; Ezip = Ezic;

wt_str = incident_plane_wave.waveform_type;
wi = incident_plane_wave.waveform_index;

% if waveform is Gaussian waveforms
if strcmp(incident_plane_wave.waveform_type,'gaussian')
    tau = waveforms.gaussian(wi).tau;
    t_0 = waveforms.gaussian(wi).t_0;
    Exic = Exi0 * exp(-((te - t_0 - k_dot_r_ex )/tau).^2);
    Eyic = Eyi0 * exp(-((te - t_0 - k_dot_r_ey )/tau).^2);
    Ezic = Ezi0 * exp(-((te - t_0 - k_dot_r_ez )/tau).^2);
    Hxic = Hxi0 * exp(-((tm - t_0 - k_dot_r_hx )/tau).^2);
    Hyic = Hyi0 * exp(-((tm - t_0 - k_dot_r_hy )/tau).^2);
    Hzic = Hzi0 * exp(-((tm - t_0 - k_dot_r_hz )/tau).^2);
end

% if waveform is derivative of Gaussian
if strcmp(incident_plane_wave.waveform_type,'derivative_gaussian')
    tau = waveforms.derivative_gaussian(wi).tau;
    t_0 = waveforms.derivative_gaussian(wi).t_0;
    Exic = Exi0 * (-sqrt(2*exp(1))/tau)*(te - t_0 - k_dot_r_ex) ...
           .*exp(-((te - t_0 - k_dot_r_ex)/tau).^2);
    Eyic = Eyi0 * (-sqrt(2*exp(1))/tau)*(te - t_0 - k_dot_r_ey) ...
           .*exp(-((te - t_0 - k_dot_r_ey)/tau).^2);
    Ezic = Ezi0 * (-sqrt(2*exp(1))/tau)*(te - t_0 - k_dot_r_ez) ...
           .*exp(-((te - t_0 - k_dot_r_ez)/tau).^2);
    Hxic = Hxi0 * (-sqrt(2*exp(1))/tau)*(tm - t_0 - k_dot_r_hx) ...
           .*exp(-((tm - t_0 - k_dot_r_hx)/tau).^2);
    Hyic = Hyi0 * (-sqrt(2*exp(1))/tau)*(tm - t_0 - k_dot_r_hy) ...
           .*exp(-((tm - t_0 - k_dot_r_hy)/tau).^2);
    Hzic = Hzi0 * (-sqrt(2*exp(1))/tau)*(tm - t_0 - k_dot_r_hz) ...
           .*exp(-((tm - t_0 - k_dot_r_hz)/tau).^2);
end

% if waveform is cosine modulated Gaussian
if strcmp(incident_plane_wave.waveform_type, ...
    'cosine_modulated_gaussian')
    f = waveforms.cosine_modulated_gaussian(wi).modulation_frequency;
    tau = waveforms.cosine_modulated_gaussian(wi).tau;
    t_0 = waveforms.cosine_modulated_gaussian(wi).t_0;
    Exic = Exi0 * cos(2*pi*f*(te - t_0 - k_dot_r_ex)) ...
           .*exp(-((te - t_0 - k_dot_r_ex)/tau).^2);
    Eyic = Eyi0 * cos(2*pi*f*(te - t_0 - k_dot_r_ey)) ...
           .*exp(-((te - t_0 - k_dot_r_ey)/tau).^2);
    Ezic = Ezi0 * cos(2*pi*f*(te - t_0 - k_dot_r_ez)) ...
           .*exp(-((te - t_0 - k_dot_r_ez)/tau).^2);
    Hxic = Hxi0 * cos(2*pi*f*(tm - t_0 - k_dot_r_hx)) ...
           .*exp(-((tm - t_0 - k_dot_r_hx)/tau).^2);
    Hyic = Hyi0 * cos(2*pi*f*(tm - t_0 - k_dot_r_hy)) ...
           .*exp(-((tm - t_0 - k_dot_r_hy)/tau).^2);
    Hzic = Hyi0 * cos(2*pi*f*(tm - t_0 - k_dot_r_hz)) ...
           .*exp(-((tm - t_0 - k_dot_r_hz)/tau).^2);
end
```

This subroutine is called in ***run_fdtd_time_marching_loop***, in the time-marching loop, before calling ***update_magnetic_fields*** and therefore before updating the magnetic field components. Since the waveform needs to be recalculated at every time step, this subroutine is composed of several sections, each of which pertains to a specific type of waveform. For instance, this implementation supports the Gaussian, derivative of Gaussian, and cosine-modulated Gaussian waveforms. Thus, at every iteration only the section pertaining to the type of the waveform of the incident plane wave is executed. At every time step all the six incident electric and magnetic field arrays are recalculated. One should notice that the electric and magnetic field components are calculated at time instants a half time step off from each other since the magnetic field components are evaluated at half-integer time steps while the electric field components are evaluated at integer time steps.

After ***update_incident_fields*** is called in ***run_fdtd_time_marching_loop***, the subroutine ***update_magnetic_fields*** is called in which the magnetic field components are updated for the current time step. The implementation of ***update_magnetic_fields*** is modified as shown in Listing 11.5 to accommodate the scattered field formulation. Here the general updating equation is kept as is; however, an additional section is added that will be executed if the mode of the simulation is scattering. In this additional section the incident field arrays are multiplied by the corresponding coefficients and are added to the magnetic field components, which in this case are assumed to be scattered fields. Similarly, the implementation of ***update_electric_fields*** is modified as shown in Listing 11.6 to accommodate the scattered field formulation for updating the scattered electric field components.

11.4.5 Postprocessing and simulation results

The scattered electric and magnetic field components at desired locations can be captured as the FDTD simulation is running by defining **sampled_electric_fields** and **sampled_magnetic_fields** in the subroutine ***define_output_parameters*** before the FDTD simulation is started. Another type of output from a scattering simulation is the RCS as discussed in Section 11.3. The calculation of RCS follows the same procedure as the NF–FF transformation. Only a slight modification of the code is required at the last stage of the NF–FF transformation.

Listing 11.5 update_magnetic_fields

```
% update magnetic fields

current_time = current_time + dt/2;

Hx = Chxh.*Hx+Chxey.*(Ey(1:nxp1,1:ny,2:nzp1)-Ey(1:nxp1,1:ny,1:nz)) ...
    + Chxez.*(Ez(1:nxp1,2:nyp1,1:nz)-Ez(1:nxp1,1:ny,1:nz));

Hy = Chyh.*Hy+Chyez.*(Ez(2:nxp1,1:nyp1,1:nz)-Ez(1:nx,1:nyp1,1:nz)) ...
    + Chyex.*(Ex(1:nx,1:nyp1,2:nzp1)-Ex(1:nx,1:nyp1,1:nz));

Hz = Chzh.*Hz+Chzex.*(Ex(1:nx,2:nyp1,1:nzp1)-Ex(1:nx,1:ny,1:nzp1))  ...
    + Chzey.*(Ey(2:nxp1,1:ny,1:nzp1)-Ey(1:nx,1:ny,1:nzp1));

if incident_plane_wave.enabled
    Hx = Hx + Chxhic .* Hxic + Chxhip .* Hxip;
    Hy = Hy + Chyhic .* Hyic + Chyhip .* Hyip;
    Hz = Hz + Chzhic .* Hzic + Chzhip .* Hzip;
end
```

Listing 11.6 update_electric_fields

```
% update electric fields except the tangential components
% on the boundaries

current_time = current_time + dt/2;

Ex(1:nx,2:ny,2:nz) = Cexe(1:nx,2:ny,2:nz).*Ex(1:nx,2:ny,2:nz) ...
                     + Cexhz(1:nx,2:ny,2:nz).*...
                     (Hz(1:nx,2:ny,2:nz)-Hz(1:nx,1:ny-1,2:nz)) ...
                     + Cexhy(1:nx,2:ny,2:nz).*...
                     (Hy(1:nx,2:ny,2:nz)-Hy(1:nx,2:ny,1:nz-1));

Ey(2:nx,1:ny,2:nz)=Ceye(2:nx,1:ny,2:nz).*Ey(2:nx,1:ny,2:nz) ...
                     + Ceyhx(2:nx,1:ny,2:nz).*   ...
                     (Hx(2:nx,1:ny,2:nz)-Hx(2:nx,1:ny,1:nz-1)) ...
                     + Ceyhz(2:nx,1:ny,2:nz).*   ...
                     (Hz(2:nx,1:ny,2:nz)-Hz(1:nx-1,1:ny,2:nz));

Ez(2:nx,2:ny,1:nz)=Ceze(2:nx,2:ny,1:nz).*Ez(2:nx,2:ny,1:nz) ...
                     + Cezhy(2:nx,2:ny,1:nz).*   ...
                     (Hy(2:nx,2:ny,1:nz)-Hy(1:nx-1,2:ny,1:nz)) ...
                     + Cezhx(2:nx,2:ny,1:nz).*...
                     (Hx(2:nx,2:ny,1:nz)-Hx(2:nx,1:ny-1,1:nz));

if incident_plane_wave.enabled
    Ex = Ex + Cexeic .* Exic + Cexeip .* Exip;
    Ey = Ey + Ceyeic .* Eyic + Ceyeip .* Eyip;
    Ez = Ez + Cezeic .* Ezic + Cezeip .* Ezip;
end
```

However, before discussing this modification, it should be noted that if the calculation of the RCS is desired, the frequencies for which the RCS calculation is sought must be defined in the subroutine ***define_output_parameters*** as the parameter **farfield.frequencies(i)**. Furthermore, the parameter **farfield.number_of_cells_from_outer_boundary** must be defined as well.

The calculation of RCS from the captured NF–FF fictitious currents and display of results are performed in the subroutine ***calculate_and_display_farfields***. The initial lines of this subroutine are modified as shown in Listing 11.7. In the last part of the given code section it can be seen that if the mode of the simulation is scattering, then the plotted radiation patterns are RCS; otherwise, plots show the directivity of the field patterns. Another modification is that a new subroutine named ***calculate_incident_plane_wave_power*** is called after the subroutine ***calculate_radiated_power***. The RCS calculation is based on (11.21), which requires the value of the incident field power. Hence, the subroutine ***calculate_incident_plane_wave_power***, which is shown in Listing 11.8, is implemented to calculate the power of the incident plane wave at the desired frequency or frequencies based on (11.22).

The actual calculation of RCS is performed in the subroutine ***calculate_farfields_per_plane***. The last part of this subroutine is modified as shown in Listing 11.9. If the mode of the simulation is scattering then RCS is calculated using (11.21); otherwise, the directivity is calculated.

Finally, a section of code is added to subroutine ***display_transient_parameters***, as shown in Listing 11.10, to display the waveform of the incident plane wave that would be observed at the origin. It should be noted that the displayed waveform includes the temporal shift t_0 but not the spatial shift l_0.

Listing 11.7 calculate_and_display_farfields

```
if number_of_farfield_frequencies == 0
    return;
end

calculate_radiated_power;
calculate_incident_plane_wave_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

if incident_plane_wave.enabled == false
    plot_type = 'D';
else
    plot_type = 'RCS';
end
```

Listing 11.8 calculate_incident_plane_wave_power

```
% Calculate total radiated power
if incident_plane_wave.enabled == false
    return;
end
x = incident_plane_wave.waveform;
time_shift = 0;
[X] = time_to_frequency_domain(x,dt,farfield.frequencies,time_shift);
incident_plane_wave.frequency_domain_value = X;
incident_plane_wave.frequencies = frequency_array;
E_t = incident_plane_wave.E_theta;
E_p = incident_plane_wave.E_phi;
eta_0 = sqrt(mu_0/eps_0);
incident_plane_wave.incident_power = ...
    (0.5/eta_0) * (E_t^2 + E_p^2) * abs(X)^2;
```

Listing 11.9 calculate_farfields_per_plane

```
  if incident_plane_wave.enabled == false
    % calculate directivity
     farfield_dataTheta(mi,:)=(k^2./(8*pi*eta_0*radiated_power(mi))) ...
        .* (abs(Lphi+eta_0*Ntheta).^2);
     farfield_dataPhi(mi,:)  =(k^2./(8*pi*eta_0*radiated_power(mi))) ...
        .* (abs(Ltheta-eta_0*Nphi).^2);
  else
    % calculate radar cross-section
     farfield_dataTheta(mi,:)  = ...
        (k^2./(8*pi*eta_0*incident_plane_wave.incident_power(mi))) ...
        .* (abs(Lphi+eta_0*Ntheta).^2);
     farfield_dataPhi(mi,:)    = ...
        (k^2./(8*pi*eta_0*incident_plane_wave.incident_power(mi))) ...
        .* (abs(Ltheta-eta_0*Nphi).^2);
  end
```

Listing 11.10 display_transient_parameters

```
% figure for incident plane wave
if incident_plane_wave.enabled == true
    figure;
    xlabel('time␣(ns)','fontsize',12);
    ylabel('(volt/meter)','fontsize',12);
    title('Incident␣electric␣field','fontsize',12);
    grid on; hold on;
    sampled_value_theta = incident_plane_wave.E_theta * ...
        incident_plane_wave.waveform;
    sampled_value_phi = incident_plane_wave.E_phi * ...
        incident_plane_wave.waveform;
    sampled_time = time(1:time_step)*1e9;
    plot(sampled_time, sampled_value_theta,'b-',...
        sampled_time, sampled_value_phi,'r:','linewidth',1.5);
    legend('E_{\theta,inc}','E_{\phi,inc}'); drawnow;
end
```

11.5 Simulation examples

In the previous sections we discussed scattered field algorithm and demonstrated its implementation using MATLAB® programs. In this section we provide examples of scattering problems.

11.5.1 Scattering from a dielectric sphere

In this section we present the calculation of the bistatic RCS of a dielectric sphere. Figure 11.4 shows an FDTD problem space including a dielectric sphere illuminated by an x polarized plane wave traveling in the positive z direction. The problem space is composed of cells with size $\Delta x = 0.75$ cm, $\Delta y = 0.75$ cm, and $\Delta z = 0.75$ cm. The dielectric sphere radius is 10 cm, relative permittivity is 3, and relative permeability is 2. The definition of the problem space is shown in Listing 11.11. The incident plane wave source is defined as shown in Listing 11.12. The waveform of the plane wave is Gaussian. The output of the FDTD simulation is defined as far field at 1 GHz, which indicates that the RCS will be calculated. Furthermore, the x component of the scattered electric field at the origin is defined as the output as shown in Listing 11.13.

The simulation is run for 2,000 time steps, and the sampled electric field at the origin and the RCS plots are obtained. Figure 11.5 shows the sampled scattered electric field captured at the origin and shows the waveform of the incident plane wave. Figures 11.6, 11.7, and 11.8 show the RCS of the dielectric sphere in the xy, xz, and yz planes, respectively.

To verify the accuracy of the calculated RCSs, the results are compared with the analytical solution presented in [44]. The θ component of the RCS in the xz plane calculated by the

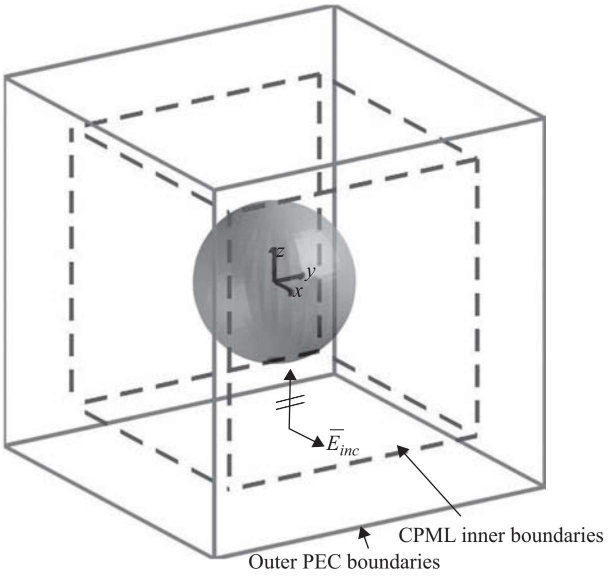

Figure 11.4 An FDTD problem space including a dielectric sphere.

Listing 11.11 define_geometry.m

```
disp('defining the problem geometry');

bricks  = [];
spheres = [];
thin_wires = [];

% define a sphere
spheres(1).radius = 100e-3;
spheres(1).center_x = 0;
spheres(1).center_y = 0;
spheres(1).center_z = 0;
spheres(1).material_type = 4;
```

Listing 11.12 define_sources_and_lumped_elements.m

```
disp('defining sources and lumped element components');

voltage_sources = [];
current_sources = [];
diodes = [];
resistors = [];
inductors = [];
capacitors = [];
incident_plane_wave = [];

% define source waveform types and parameters
waveforms.gaussian(1).number_of_cells_per_wavelength = 0;
waveforms.gaussian(2).number_of_cells_per_wavelength = 15;

% Define incident plane wave, angles are in degrees
incident_plane_wave.E_theta = 1;
incident_plane_wave.E_phi = 0;
incident_plane_wave.theta_incident = 0;
incident_plane_wave.phi_incident = 0;
incident_plane_wave.waveform_type = 'gaussian';
incident_plane_wave.waveform_index = 1;
```

FDTD simulation is compared with the analytical solution in the Cartesian plot in Figure 11.9, whereas the ϕ component of the RCS in the *yz* plane calculated by the FDTD simulation is compared with the analytical solution in the Cartesian plot in Figure 11.10. Both comparisons show very good agreement.

Listing 11.13 shows that an animation is also defined as an output. The scattered electric field is thus captured on the *xz* plane displayed during the simulation. Figure 11.11 shows the snapshots of the animation captured at time steps 120, 140, 160, and 180.

Listing 11.13 define_output_parameters.m

```
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) = 1.0e9;
farfield.number_of_cells_from_outer_boundary = 13;

% frequency domain parameters
frequency_domain.start = 20e6;
frequency_domain.end   = 4e9;
frequency_domain.step  = 20e6;


% define sampled electric fields
% component: vector component íx í,í y í,í z í, or magnitude ímí
% display plot = true, in order to plot field during simulation
sampled_electric_fields(1).x = 0;
sampled_electric_fields(1).y = 0;
sampled_electric_fields(1).z = 0;
sampled_electric_fields(1).component = 'x';
sampled_electric_fields(1).display_plot = false;

% define animation
% field_type shall be 'e' or 'h'
% plane cut shall be 'xy', yz, or zx
% component shall be 'x', 'y', 'z', or 'm;
animation(1).field_type = 'e';
animation(1).component = 'm';
animation(1).plane_cut(1).type = 'zx';
animation(1).plane_cut(1).position  = 0;
animation(1).enable = true;
animation(1).display_grid = false;
animation(1).display_objects = true;
```

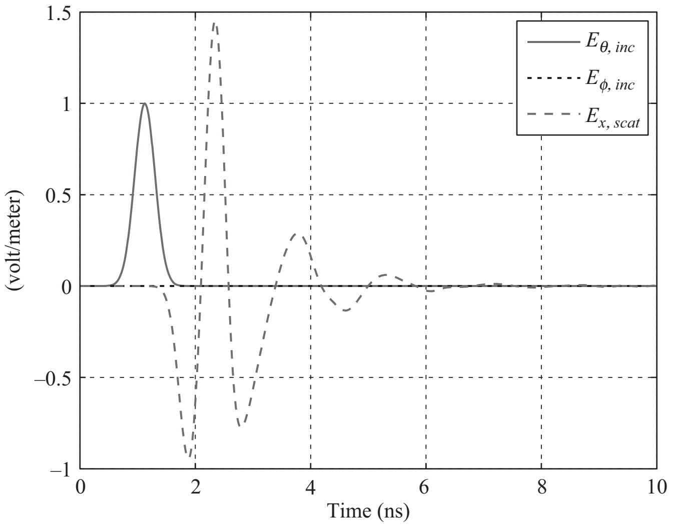

Figure 11.5 Scattered electric field captured at the origin and the incident field waveform.

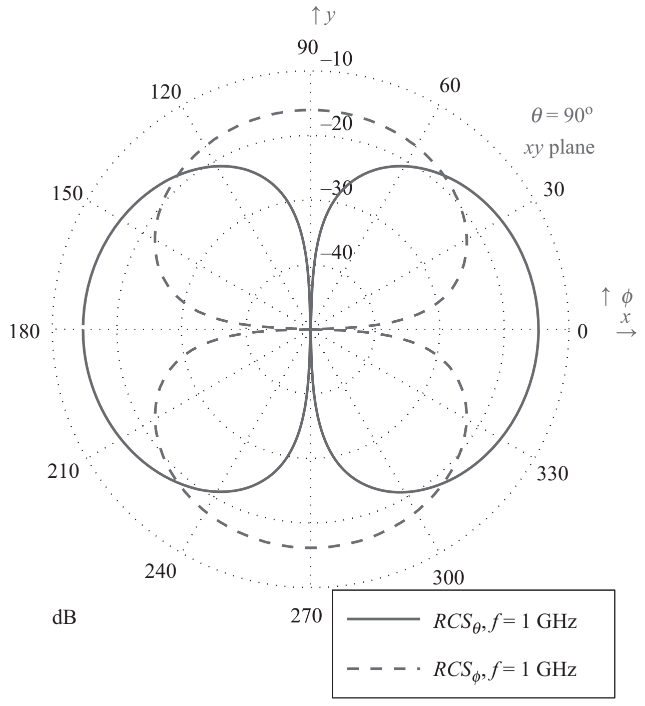

Figure 11.6 Bistatic RCS at 1 GHz in the *xy* plane.

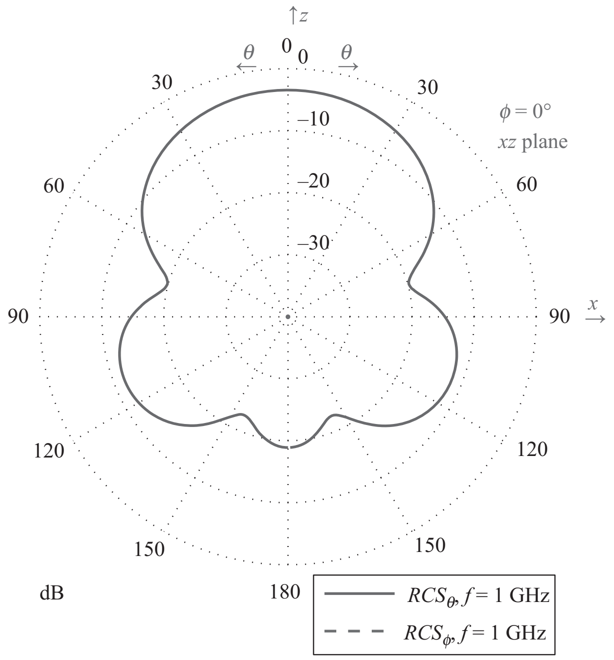

Figure 11.7 Bistatic RCS at 1 GHz in the *xz* plane.

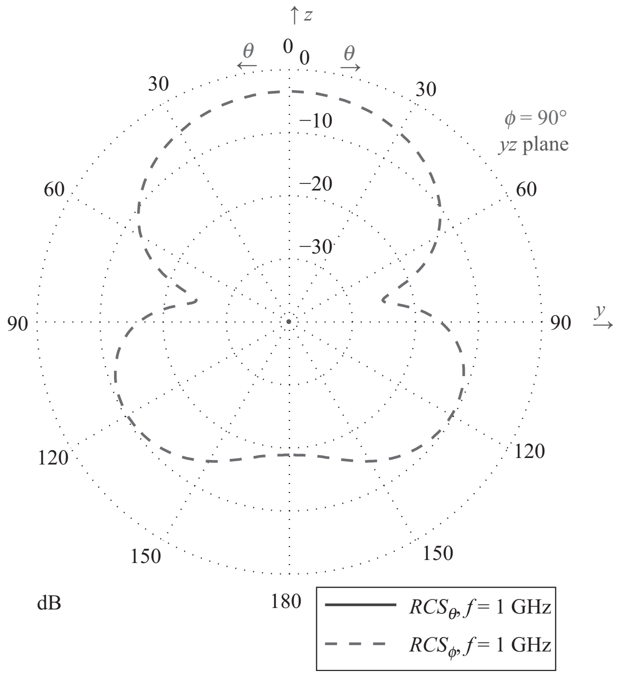

Figure 11.8 Bistatic RCS at 1 GHz in the *yz* plane.

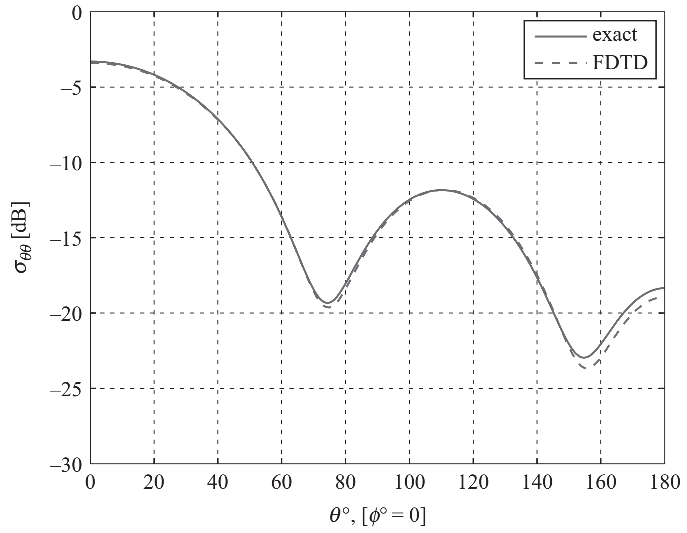

Figure 11.9 Calculated RCS_θ at 1 GHz in the *xz* plane compared with the analytical solution.

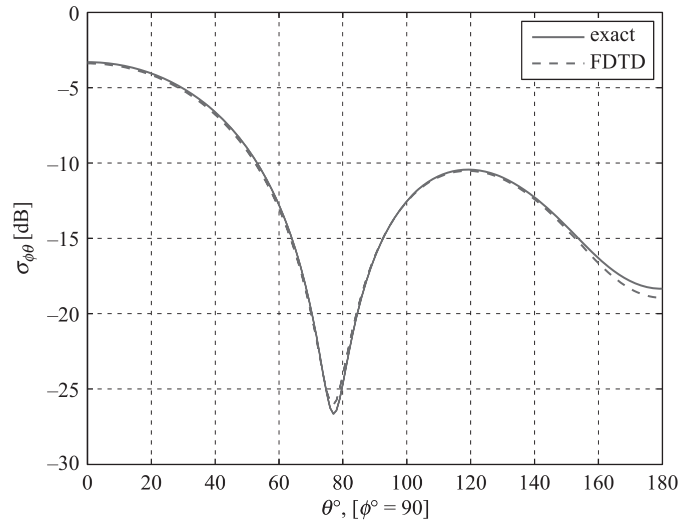

Figure 11.10 Calculated RCS_ϕ at 1 GHz in the *yz* plane compared with the analytical solution.

11.5.2 Scattering from a dielectric cube

In the previous section calculation of the bistatic RCS of a dielectric sphere is presented. In this section we present calculation of the RCS of a cube using the FDTD method and comparison with results obtained by a method of moments (MoM) solver [45]. In Figure 11.12 the FDTD problem space including a dielectric cube is illustrated. In the same figure the triangular meshing of the surface of the cube used by the MoM solver is shown as well.

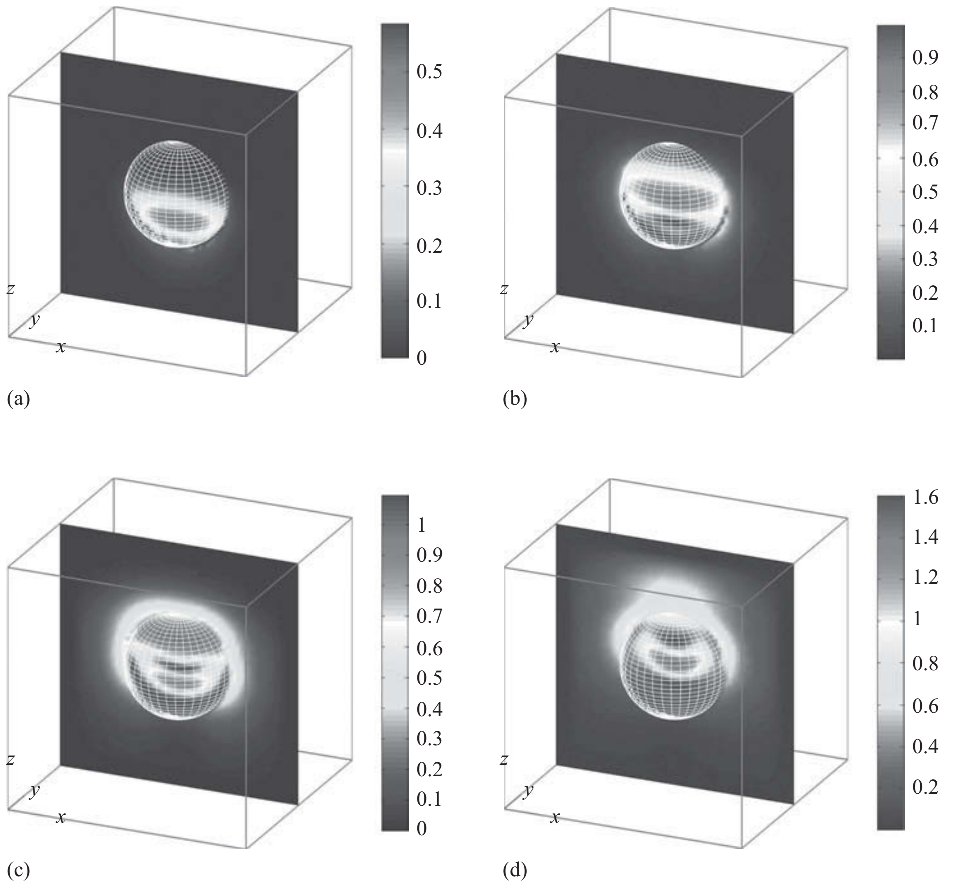

Figure 11.11 Scattered field due to a sphere captured on the *xz* plane cut: (a) at time step 120; (b) at time step 140; (c) at time step 160; and (d) at time step 180.

The FDTD problem space is composed of cubic cells 0.5 cm on a side. The definition of the problem geometry is shown in Listing 11.14. Each side of the cube is 16 cm. The relative permittivity of the cube is 5, and the relative permeability is 1. The incident plane wave is defined as shown in Listing 11.15. The incident plane wave illuminating the cube is θ polarized and propagates in a direction expressed by angles $\theta = 45°$ and $\phi = 30°$. The waveform of the plane wave is Gaussian. The far-field output is defined at 1 GHz as shown in Listing 11.16. Finally, a sampled electric field is defined at the origin to observe whether the fields in the problem space are sufficiently decayed.

After the FDTD time-marching loop is completed, the far-field patterns are obtained as shown in Figures 11.13, 11.14, and 11.15; for the RCS at 1 GHz in the *xy*, *xz*, and *yz* planes, respectively. The same problem is simulated at 1 GHz using the MoM solver. Figure 11.16 shows the compared RCS_θ in the *xz* plane, whereas Figure 11.17 shows the compared RCS_ϕ. The results show a good agreement.

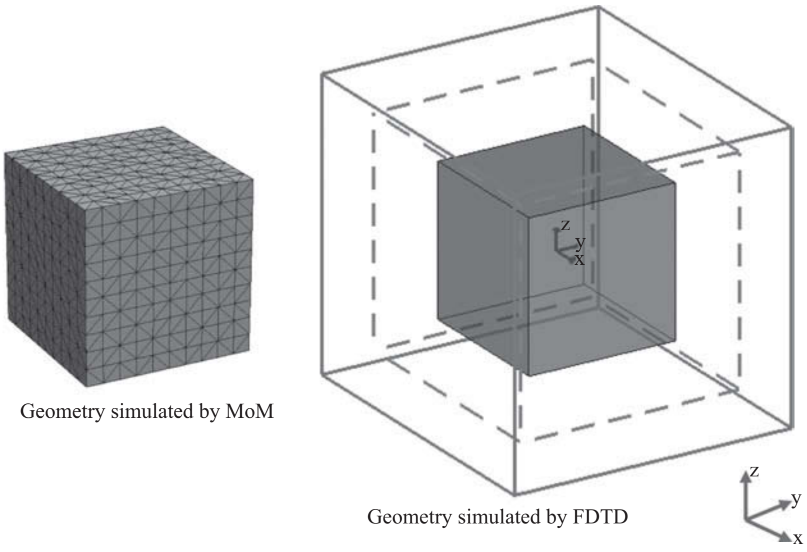

Figure 11.12 An FDTD problem space including a cube, and the surface mesh of the cube used by the MoM solver.

Listing 11.14 define_geometry.m

```
disp('defining the problem geometry');

bricks  = [];
spheres = [];
thin_wires = [];

% define dielectric
bricks(1).min_x = -80e-3;
bricks(1).min_y = -80e-3;
bricks(1).min_z = -80e-3;
bricks(1).max_x = 80e-3;
bricks(1).max_y = 80e-3;
bricks(1).max_z = 80e-3;
bricks(1).material_type = 4;
```

Listing 11.15 define_sources_and_lumped_elements.m

```
disp('defining sources and lumped element components');

voltage_sources = [];
current_sources = [];
diodes = [];
resistors = [];
inductors = [];
capacitors = [];
incident_plane_wave = [];

% define source waveform types and parameters
```

```
waveforms.gaussian(1).number_of_cells_per_wavelength = 0;
waveforms.gaussian(2).number_of_cells_per_wavelength = 15;

% Define incident plane wave, angles are in degrees
incident_plane_wave.E_theta = 1;
incident_plane_wave.E_phi = 0;
incident_plane_wave.theta_incident = 45;
incident_plane_wave.phi_incident = 30;
incident_plane_wave.waveform_type = 'gaussian';
incident_plane_wave.waveform_index = 1;
```

Listing 11.16 define_output_parameters.m

```
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) = 1.0e9;
farfield.number_of_cells_from_outer_boundary = 13;

% frequency domain parameters
frequency_domain.start = 20e6;
frequency_domain.end   = 4e9;
frequency_domain.step  = 20e6;


% define sampled electric fields
% component: vector component íx í,í y í,í z í, or magnitude ímí
% display plot = true, in order to plot field during simulation
sampled_electric_fields(1).x = 0;
sampled_electric_fields(1).y = 0;
sampled_electric_fields(1).z = 0;
sampled_electric_fields(1).component = 'x';
sampled_electric_fields(1).display_plot = false;
```

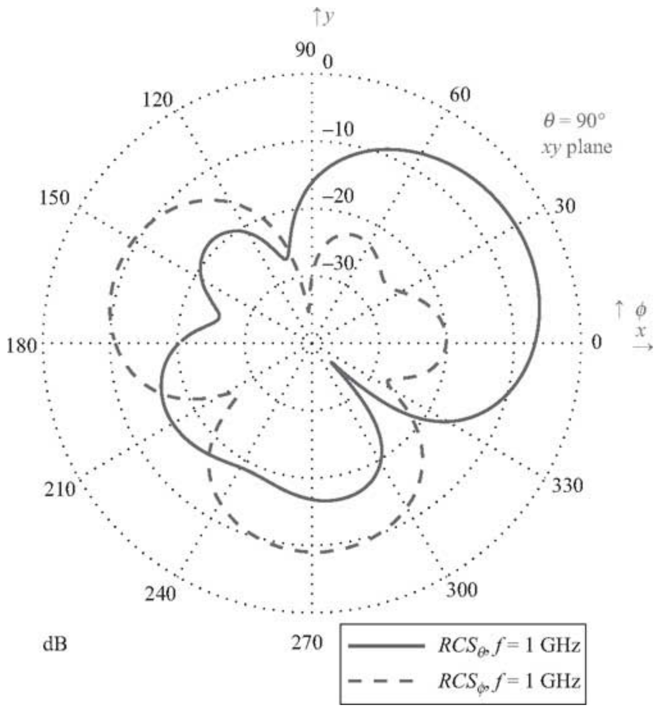

Figure 11.13 Bistatic RCS at 1 GHz in the *xy* plane.

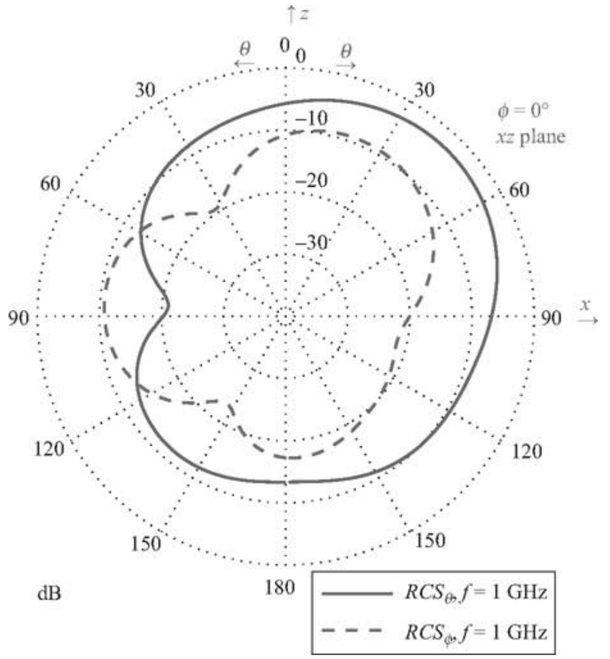

Figure 11.14 Bistatic RCS at 1 GHz in the *xz* plane.

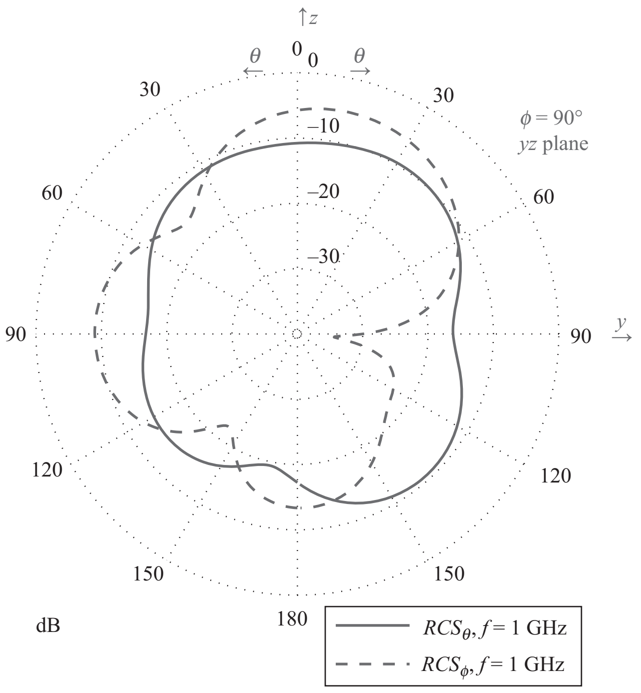

Figure 11.15 Bistatic RCS at 1 GHz in the *yz* plane.

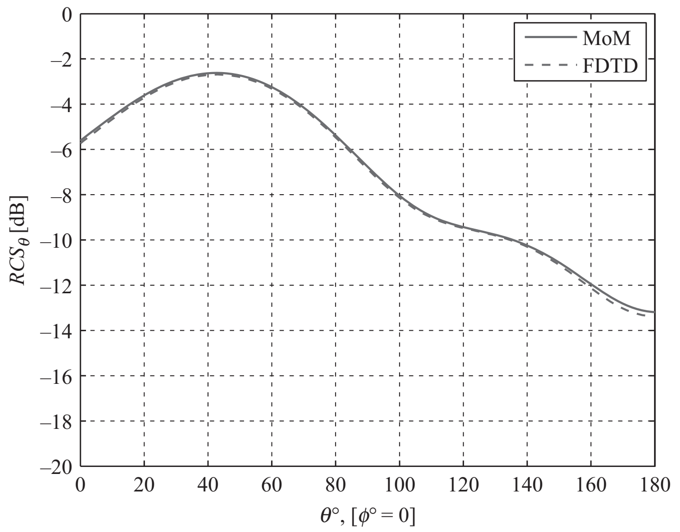

Figure 11.16 Calculated RCS_θ at 1 GHz in the *xz* plane compared with the MoM solution.

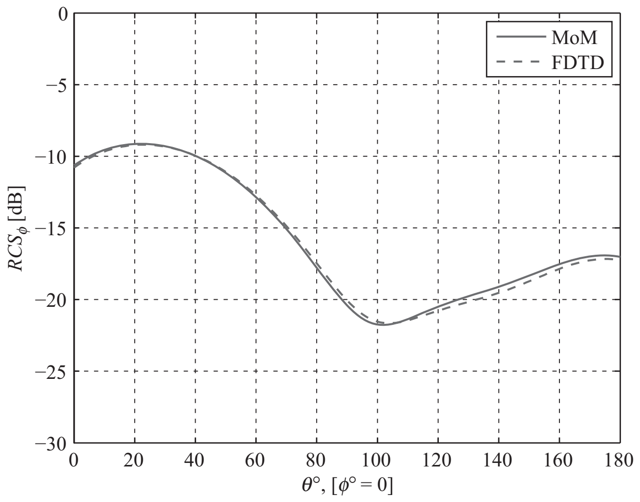

Figure 11.17 Calculated RCS_ϕ at 1 GHz in the xz plane compared with the MoM solution.

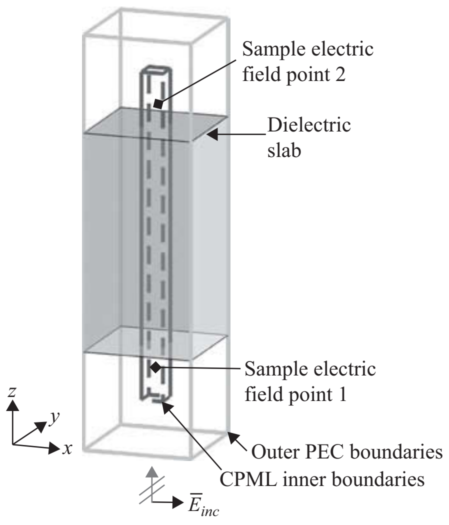

Figure 11.18 An FDTD problem space including a dielectric slab.

11.5.3 Reflection and transmission coefficients of a dielectric slab

Chapter 8 demonstrated that the objects with infinite length can be simulated by letting the objects penetrate into the convolutional perfectly matched layer (CPML) boundaries. In this example we show the simulation of a dielectric slab, which is essentially a one-dimensional problem. The availability of incident plane wave allows us to calculate the reflection and transmission coefficients of the slab.

The geometry of the problem in consideration is shown in Figure 11.18. A dielectric slab, having dielectric constant 4 and thickness 20 cm, penetrates into the CPML boundaries in *xn*, *xp*, *yn*, and *yp* directions. The problem space is composed of cells having 0.5 cm size on a side. The source of the simulation is an *x*-polarized incident field traveling in the positive *z* direction. Two sampled electric fields are defined to capture the *x* component of the electric field at two points, each 5 cm away from the slab – one below the slab and the other above the slab as illustrated in Figure 11.18. The definition of the problem space boundary conditions, incident field, the slab, and the sampled electric fields are shown in Listings 11.17–11.20, respectively. Notice that the CPML parameters have been modified for better performance.

In the previous sections, *scattered field formulation* has been demonstrated for calculating the *scattered field* when an *incident field* is used to excite the problem space. The sampled electric field therefore is the one being captured while the simulation is running.

Listing 11.17 define_problem_space_parameters.m

```
% ==< boundary conditions >========
% Here we define the boundary conditions parameters
% 'pec' : perfect electric conductor
% 'cpml' : conlvolutional PML
% if cpml_number_of_cells is less than zero
% CPML extends inside of the domain rather than outwards

boundary.type_xn = 'cpml';
boundary.air_buffer_number_of_cells_xn = 0;
boundary.cpml_number_of_cells_xn = -8;

boundary.type_xp = 'cpml';
boundary.air_buffer_number_of_cells_xp = 0;
boundary.cpml_number_of_cells_xp = -8;

boundary.type_yn = 'cpml';
boundary.air_buffer_number_of_cells_yn = 0;
boundary.cpml_number_of_cells_yn = -8;

boundary.type_yp = 'cpml';
boundary.air_buffer_number_of_cells_yp = 0;
boundary.cpml_number_of_cells_yp = -8;

boundary.type_zn = 'cpml';
boundary.air_buffer_number_of_cells_zn = 10;
boundary.cpml_number_of_cells_zn = 8;

boundary.type_zp = 'cpml';
boundary.air_buffer_number_of_cells_zp = 10;
boundary.cpml_number_of_cells_zp = 8;

boundary.cpml_order = 4;
boundary.cpml_sigma_factor = 1;
boundary.cpml_kappa_max = 1;
boundary.cpml_alpha_min = 0;
boundary.cpml_alpha_max = 0;
```

Listing 11.18 define_geometry.m

```
% define dielectric
bricks(1).min_x = -0.05;
bricks(1).min_y = -0.05;
bricks(1).min_z = 0;
bricks(1).max_x = 0.05;
bricks(1).max_y = 0.05;
bricks(1).max_z = 0.2;
bricks(1).material_type = 4;
```

Listing 11.19 define_sources_and_lumped_elements.m

```
% define source waveform types and parameters
waveforms.derivative_gaussian(1).number_of_cells_per_wavelength = 20;

% Define incident plane wave, angles are in degrees
incident_plane_wave.E_theta = 1;
incident_plane_wave.E_phi = 0;
incident_plane_wave.theta_incident = 0;
incident_plane_wave.phi_incident = 0;
incident_plane_wave.waveform_type = 'derivative_gaussian';
incident_plane_wave.waveform_index = 1;
```

Listing 11.20 define_output_parameters.m

```
% frequency domain parameters
frequency_domain.start = 2e6;
frequency_domain.end   = 2e9;
frequency_domain.step  = 1e6;

% define sampled electric fields
% component: vector component íx í,í y í,í z í, or magnitude ímí
% display plot = true, in order to plot field during simulation
sampled_electric_fields(1).x = 0;
sampled_electric_fields(1).y = 0;
sampled_electric_fields(1).z = -0.025;
sampled_electric_fields(1).component = 'x';
sampled_electric_fields(1).display_plot = false;

sampled_electric_fields(2).x = 0;
sampled_electric_fields(2).y = 0;
sampled_electric_fields(2).z = 0.225;
sampled_electric_fields(2).component = 'x';
sampled_electric_fields(2).display_plot = false;
```

The reflected field from the slab is the scattered field, and the ratio of the reflected field to incident field is required to calculate the reflection coefficient using

$$|\Gamma| = \frac{|\vec{E}_{scat}|}{|\vec{E}_{inc}|}, \tag{11.23}$$

Listing 11.21 capture_sampled_electric_fields.m

```
% Capturing the incident electric fields
if incident_plane_wave.enabled
    for ind=1:number_of_sampled_electric_fields
        is = sampled_electric_fields(ind).is;
        js = sampled_electric_fields(ind).js;
        ks = sampled_electric_fields(ind).ks;

        switch (sampled_electric_fields(ind).component)
            case 'x'
                sampled_value = 0.5 * sum(Exic(is-1:is,js,ks));
            case 'y'
                sampled_value = 0.5 * sum(Eyic(is,js-1:js,ks));
            case 'z'
                sampled_value = 0.5 * sum(Ezic(is,js,ks-1:ks));
            case 'm'
                svx = 0.5 * sum(Exic(is-1:is,js,ks));
                svy = 0.5 * sum(Eyic(is,js-1:js,ks));
                svz = 0.5 * sum(Ezic(is,js,ks-1:ks));
                sampled_value = sqrt(svx^2 + svy^2 + svz^2);
        end
        sampled_electric_fields(ind).incident_field_value(time_step) ...
                                                          = sampled_value;
    end
end
```

where Γ is the reflection coefficient. Here the scattered field is the one sampled at a point below the slab, and the incident field shall be sampled at the same point as well. Therefore, the section of code shown in Listing 11.21 is added to the subroutine ***capture_sampled_electric_fields*** to sample the incident electric field as the time-marching loop proceeds.

The ratio of the transmitted field to incident field is required to calculate the transmission coefficient, where the transmitted field is the *total field* that is the sum of the incident and scattered fields. Then the transmission coefficient can be calculated using

$$|T| = \frac{|\vec{E}_{tot}|}{|\vec{E}_{inc}|} = \frac{|\vec{E}_{scat} + \vec{E}_{inc}|}{|\vec{E}_{inc}|}, \tag{11.24}$$

where T is the transmission coefficient. Here the scattered field is the one sampled at a point above the slab, and the incident field is sampled at the same point as well.

The simulation for this problem is executed, the incident and scattered fields are captured at two points above and below the slab and transformed to frequency domain, and the reflection and transmission coefficients are calculated using (11.23) and (11.24), respectively. The calculation results are plotted in Figure 11.19 together with the exact solution of the same problem. The exact solution is obtained using the program published in [46]. A very good agreement can be observed between the simulated results and exact solution up to 1 GHz.

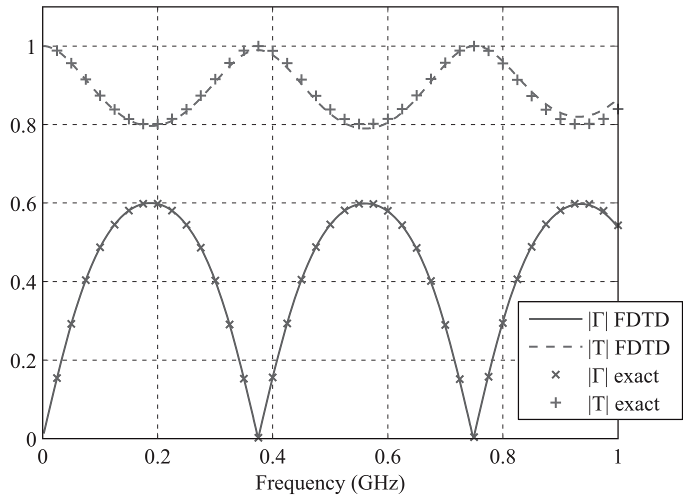

Figure 11.19 Reflection and transmission coefficients of the dielectric slab.

11.6 Exercises

11.1 Consider the problem demonstrated in Section 11.5.1, where the RCS of a dielectric sphere is calculated due to an incident field traveling in the positive z direction. In the given example the direction of incidence is defined as $\theta = 0°$ and $\phi = 0°$. Change the incident angle such that $\theta = 45°$ and $\phi = 0°$, run the simulation, and obtain the RCS at 1 GHz. Examine the RCS plots, and verify that the RCS in the xz plane is the same as the one shown in Figure 11.7, except that it is rotated by 45° about the y axis.

11.2 Consider the scattering problem constructed for Exercise 11.1, where the RCS of a dielectric sphere is calculated due to an incident field. The RCS calculations are based on the scattered fields; the fictitious electric and magnetic current densities used for NF–FF transformation are calculated using the scattered field electric and magnetic fields. The total field in the problem space is the sum of the incident field and the scattered field. Modify the program such that the RCS calculations are performed based on the total fields rather than on the scattered fields. It is necessary to modify the code of the subroutine ***calculate_JandM***. Run the simulation and obtain the RCS of the sphere. Verify that the RCS data are the same as the ones obtained from Exercise 11.1. Notice that the incident field does not contribute to far-field scattering at all.

11.3 Consider the dielectric slab problem demonstrated in Section 11.5.3. You can perform the reflection and transmission coefficient calculation for stacked slabs composed of multiple layers with different dielectric constants. For instance, create two slabs in the problem space, each 10 cm thick. Then set the dielectric constant of the bottom slab as 4 and the dielectric constant of the top slab as 2. Run the simulation, and then calculate the reflection and transmission coefficients. Compare your result with the exact solution of the same problem. For the exact solution you can refer to [46]. As an alternative, you can construct a one-dimensional FDTD code similar to the one presented in Chapter 1 based on the scattered field formulation to solve the same problem.