CHAPTER 15

Nonuniform grid

15.1 Introduction

Choice of the FDTD grid cell size is a trade-off between reduced solution time and improved accuracy. For problems in which there are no fine geometrical details or any fine details that affect the solution accuracy, this trade-off can be well balanced by choosing a cell size which is not too small to keep reasonable solution time and accuracy. If fine details are expected to affect the accuracy, some advanced techniques can be used. For instance, subgridding techniques can be employed within the uniform FDTD grid, or a nonuniform grid can be used instead of the uniform grid.

One of the methods to model fine geometrical details in an FDTD problem, while keeping the solution time reasonable, is nonuniform grids: Instead of using the same cell size all over the computational domain, different cell sizes can be used at different regions. For instance consider the problem shown in Figure 15.1(a), where a coarse uniform grid is used. In this geometry there are two fine strips which are narrower than a cells size. Moreover, some parts of the geometry do not conform to the underlying grid. An approximate geometry that would conform to this grid will be used in the conventional FDTD, hence the loss of modeling accuracy will reflect into the results of the calculations.

To model the geometry more accurately, one can use a fine uniform grid, as shown in Figure 15.1(b), which will increase the problem size – in number of cells – significantly, and increase the calculation time much more. For instance, if the cell size is halved for a three-dimensional domain, the calculations will take about 16 times longer.

Figure 15.1(c) illustrates a nonuniform grid where coarse cells are used in the background of the problem space and fine cells are used only at the fine geometrical regions. Transitions between fine cell regions and coarse cell regions are made smooth by gradually changing the cell sizes between these regions, thus numerical reflections that might occur due to transitions can be minimized. Moreover, by appropriately shifting the mesh lines, the grid is made to conform to the objects completely. Compared with the uniform fine grid, the total number of cells in this grid is much less, thus the total simulation time will be less while maintaining a level of accuracy comparable to the one obtained with the fine uniform grid.

15.2 Transition between fine and coarse grid subregions

It is essential to set a smooth transition between the fine grid regions and coarse grid regions to minimize numerical reflections that would be generated due to the transition.

In this chapter, a transition scheme will be discussed in which the cell sizes in the transition region are gradually increased starting with the fine cell size on one end and ending with the coarse cell size at the other end. While this scheme will be illustrated for a one-dimensional grid, it applies to all directions of two and three-dimensional grids as well.

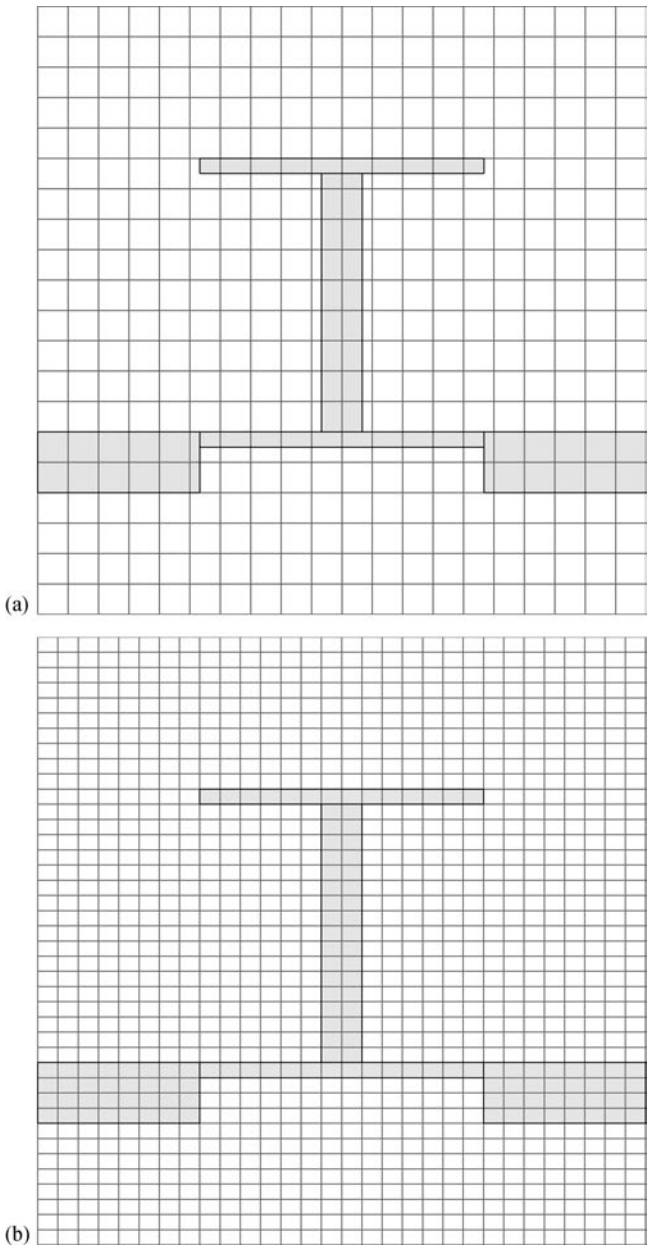

Figure 15.1 (a) An FDTD computational domain with coarse uniform grid; (b) an FDTD computational domain with fine uniform grid; and (c) an FDTD computational domain with nonuniform grid.

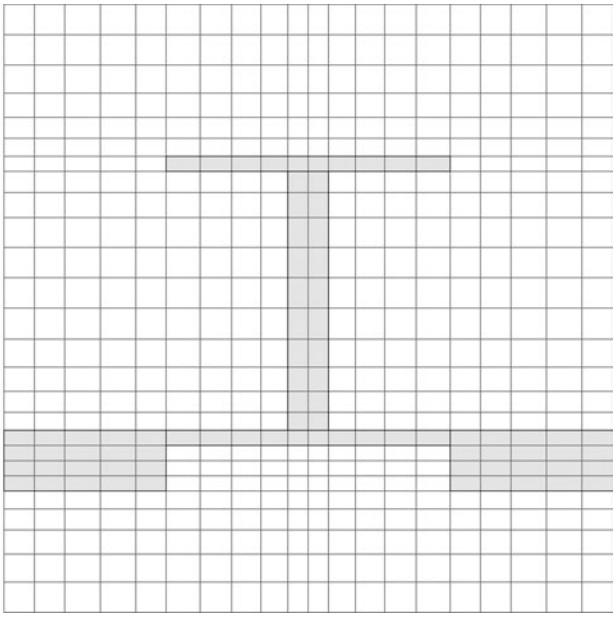

Figure 15.1 (*Continued*)

In what is called the uniform case all cells in the problem space have the same size in each coordinate direction Δx, Δy, or Δz. While each cell can have a different size in the nonuniform case, it is better to define subregions in the problem space, where needed, with uniform cells and establish nonuniform transition between the uniform subregions.

Figure 15.2 shows a nonuniform transition subregion (ΔT) between two uniform subregions. While performing the transition, the cell sizes, i.e., the distances between the subsequent electric field components, can be gradually changed. One can start with a cell size, shown as Δs length in Figure 15.2, just before the transition subregion. The length of first cell in the transition subregion can be set as

$$\Delta 1 = R\Delta s, \tag{15.1}$$

where R is rate of change between subsequent half cells. Thus the lengths of the cells in the transition subregion can be expressed as

$$\Delta M = R^M \Delta s, \tag{15.2}$$

where M is an index denoting a cell in the transition subregion. At the end of the transition the length of the first cell of the next uniform subregion is

$$\Delta e = R^{N_{nu}+1} \Delta s, \tag{15.3}$$

where N_{nu} is the number of cells in the transition subregion. If the transition is desired to happen on a given transition length ΔT, then it is needed to determine the rate R and the number of cells N_{nu}.

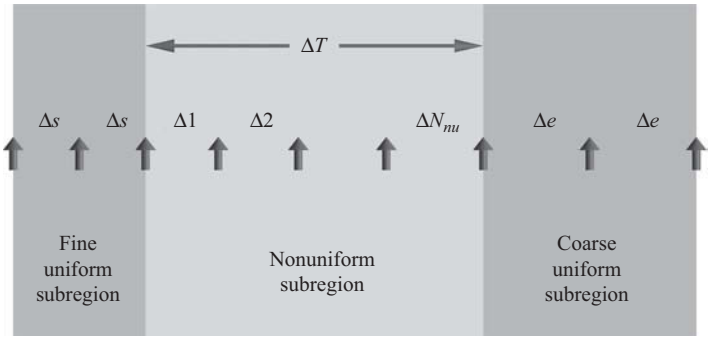

Figure 15.2 Uniform and nonuniform subregions.

The total length of the transition subregion, in addition to one cell at both ends from the uniform regions is

$$\Delta T + \Delta s + \Delta e = \sum_{k=0}^{N_{nu}+1} \Delta s R^k = \frac{\Delta s - \Delta s R^{N_{nu}+2}}{1-R} = \frac{\Delta s - \Delta e R}{1-R}. \tag{15.4}$$

Equation (15.4) can be used to find R as

$$R = \frac{\Delta T + \Delta e}{\Delta T + \Delta s}. \tag{15.5}$$

Then, the number of cells, N_{nu}, can be determined using (15.3) as

$$N_{nu} = \frac{\log(\Delta e/\Delta s)}{\log(R)} - 1. \tag{15.6}$$

One should notice that N_{nu} may not be found as an integer number by (15.6), however, it has to be an integer number. In this case, N_{nu} can be rounded to the closest appropriate integer. It should be noted that, if N_{nu} is rounded to a larger integer number, the transition cell sizes will be smaller than they are supposed to be and consequently a transition cell may become smaller than the fine uniform subregion cell size. Similarly, if N_{nu} is rounded to a smaller integer number, the cell sizes will be larger and a transition cell may get larger than the coarse uniform subregion cell size. In either case, there may be an abrupt transition between the uniform and nonuniform subregions, which may enhance the numerical error due to the transition. A smoother transition may be achieved if the transition region is selected to be appropriately long.

Since in a Yee cell the electric and magnetic field components are located at distinct positions, we can define a grid of electric field components as wells as a grid of magnetic field components. Figure 15.3 illustrates these two grids for a uniform one-dimensional space of N_x cells in the x direction. In the figure "nc_xe" denotes the coordinates of electric field grid points, whereas "nc_xh" denotes the coordinates of magnetic field points. The nonuniform electric field grid can be set up by determining the sizes of the cells (i.e., distances between the neighboring electric field components) as discussed above. Once the positions of the electric field components are determined, magnetic field components can be placed at the centers of the cells of the electric field grid and the magnetic field grid can be

constructed as shown in Figure 15.4. A representation of a fine grid with nonuniform surrounding regions is illustrated in Figure 15.5.

This kind of field positioning on a dual grid, where magnetic fields are located at the midpoints of electric field grid cells, has been discussed and analyzed in [77] in terms of its accuracy. Since magnetic fields are at the midpoints, as it is also the case in a uniform grid, the corresponding updating equations that update magnetic fields are second order accurate. The electric fields are off-centered between the magnetic field components, which leads to

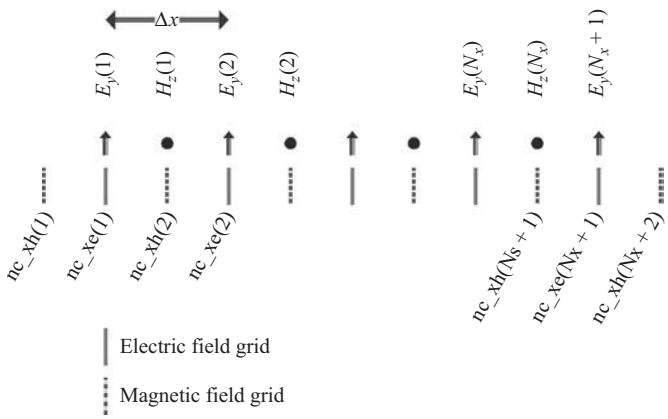

Figure 15.3 Uniform electric and magnetic field grids in a one-dimensional space.

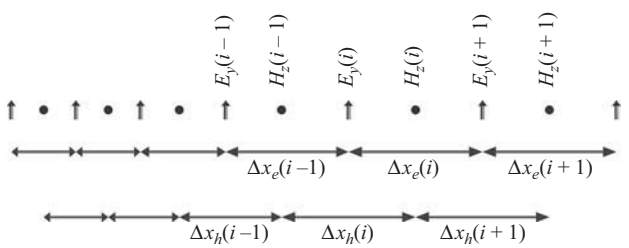

Figure 15.4 Electric and magnetic field positions in a nonuniform grid.

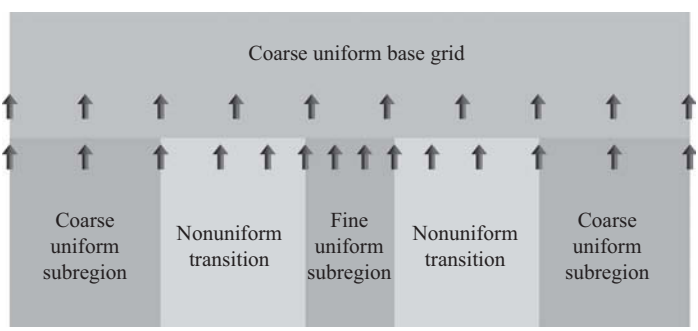

Figure 15.5 A fine grid subregion embedded in a coarse base grid.

first order accurate updating equations. However, it has been shown in [77] that, despite being first order accurate locally, Yee's scheme is second-order convergent regardless of mesh nonuniformity.

15.3 FDTD updating equations for the nonuniform grids

From Figure 15.4, one should notice that electric field components are located about the middle of the magnetic field grid cells, while the magnetic field components are located at the middle of the electric field grid cells. The cells in these two grids are different in size in the nonuniform regions, thus the cell sizes are denoted as Δx_e and Δx_h for electric and magnetic grids, respectively. Similarly, for a three-dimensional case we would have Δy_e, Δy_h, Δz_e, and Δz_h for the respective directions.

It is straightforward to construct general FDTD updating equations using the field and cell indexing scheme shown in Figure 15.4 for the one-dimensional case. Here, the E_y and H_z updating equations become

$$E_y^{n+1}(i) = C_{eye}(i) \times E_y^n(i) + C_{eyhz}(i) \times \left(H_z^{n+\frac{1}{2}}(i) - H_z^{n+\frac{1}{2}}(i-1) \right) + C_{eyj}(i) \times J_{iy}^{n+\frac{1}{2}}(i), \quad (15.7)$$

where

$$C_{eye}(i) = \frac{2\varepsilon_y(i) - \Delta t \sigma_y^e(i)}{2\varepsilon_y(i) + \Delta t \sigma_y^e(i)}$$

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

$$C_{eyj}(i) = -\frac{2\Delta t}{2\varepsilon_y(i) + \Delta t \sigma_y^e(i)}$$

and

$$H_z^{n+\frac{1}{2}}(i) = C_{hzh}(i) \times H_z^{n-\frac{1}{2}}(i) + C_{ezhy}(i) \times \left(E_y^n(i+1) - E_y^n(i) \right) + C_{hzm}(i) \times M_{iz}^n(i), \quad (15.8)$$

where

$$C_{hzh}(i) = \frac{2\mu_z(i) - \Delta t \sigma_z^m(i)}{2\mu_z(i) + \Delta t \sigma_z^m(i)}$$

$$C_{hzey}(i) = -\frac{2\Delta t}{\left(2\mu_z(i) + \Delta t \sigma_z^m(i)\right)\Delta x_e(i)}$$

$$C_{hzm}(i) = -\frac{2\Delta t}{2\mu_z(i) + \Delta t \sigma_z^m(i)}.$$

Comparing (15.7) and (15.8) with those of the uniform case in (1.40) and (1.41) one can notice that the only difference is the usage of $\Delta x_e(i)$ and $\Delta x_h(i)$ instead of a constant Δx; The spatial distance between $H_z^{n+\frac{1}{2}}(i)$ and $H_z^{n+\frac{1}{2}}(i-1)$ is $\Delta x_h(i)$, therefore $\Delta x_h(i)$ is used in (15.7), whereas the spatial distance between $E_y^n(i+1)$ and $\Delta x_e(i)$ is $\Delta x_e(i)$, therefore $\Delta x_e(i)$ is used in (15.8).

Updating equations for nonuniform two and three-dimensional cases can be obtained easily by modifying the equations of the uniform case. For instance, the updating equation for E_x in the three-dimensional case (see Chapter 1 (1.26)) can be written as

$$\begin{aligned} E_x^{n+1}(i,j,k) &= C_{exe}(i,j,k) \times E_x^n(i,j,k) \\ &+ C_{exhz}(i,j,k) \times \left(H_z^{n+\frac{1}{2}}(i,j,k) - H_z^{n+\frac{1}{2}}(i,j-1,k)\right) \\ &+ C_{exhy}(i,j,k) \times \left(H_y^{n+\frac{1}{2}}(i,j,k) - H_y^{n+\frac{1}{2}}(i,j,k-1)\right) \\ &+ C_{exj}(i,j,k) \times J_{ix}^{n+\frac{1}{2}}(i,j,k), \end{aligned} \tag{15.9}$$

where

$$C_{exe}(i,j,k) = \frac{2\varepsilon_x(i,j,k) - \Delta t\sigma_x^e(i,j,k)}{2\varepsilon_x(i,j,k) + \Delta t\sigma_x^e(i,j,k)}$$

$$C_{exhz}(i,j,k) = \frac{2\Delta t}{\left(2\varepsilon_x(i,j,k) + \Delta t\sigma_x^e(i,j,k)\right)\Delta y_h(j)}$$

$$C_{exhy}(i,j,k) = -\frac{2\Delta t}{\left(2\varepsilon_x(i) + \Delta t\sigma_x^e(i)\right)\Delta z_h(k)}$$

$$C_{exj}(i,j,k) = -\frac{2\Delta t}{2\varepsilon_x(i,j,k) + \Delta t\sigma_x^e(i,j,k)}.$$

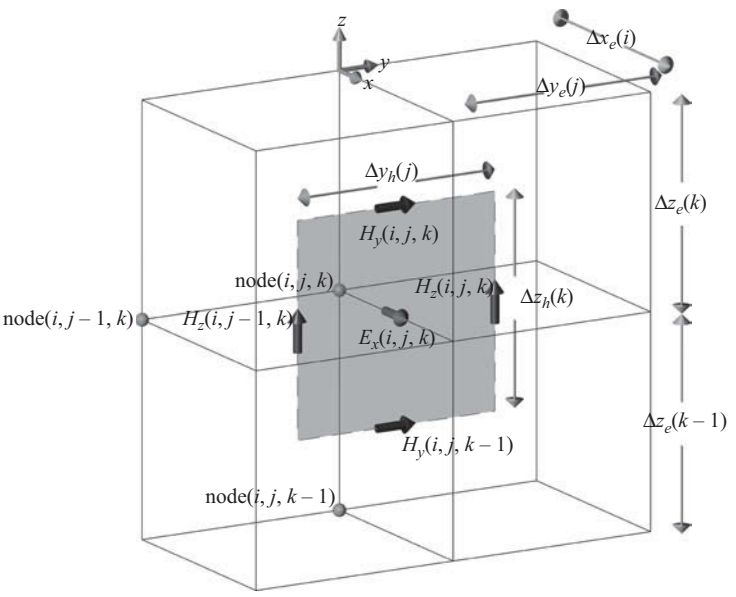

Figure 15.6 Electric field grid around $E_x(i,j,k)$, and magnetic grid cell sizes used to update $E_x(i,j,k)$.

Here the cell sizes $\Delta y_h(j)$ and $\Delta z_h(k)$ are used instead of Δy and Δz of the equation for the uniform grid. Figure 15.6 illustrates how the magnetic field components that surround the electric field component simulate Ampere's law and the path along which magnetic field curls conforms to the magnetic field grid.

Similarly, the updating equation for H_x in the three-dimensional case (Chapter 1, (1.29)) can be written as

$$\begin{aligned} H_x^{n+\frac{1}{2}}(i,j,k) = {} & C_{hxh}(i,j,k) \times H_x^{n-\frac{1}{2}}(i,j,k) \\ & + C_{hxey}(i,j,k) \times \left(E_y^n(i,j,k+1) - E_y^n(i,j,k)\right) \\ & + C_{hxez}(i,j,k) \times \left(E_z^n(i,j+1,k) - E_z^n(i,j,k)\right) \\ & + C_{hxm}(i,j,k) \times M_{ix}^n(i,j,k), \end{aligned} \tag{15.10}$$

where

$$\begin{aligned} C_{hxh}(i,j,k) &= \frac{2\mu_x(i,j,k) - \Delta t \sigma_x^m(i,j,k)}{2\mu_x(i,j,k) + \Delta t \sigma_x^m(i,j,k)} \\ C_{hxey}(i,j,k) &= \frac{2\Delta t}{\left(2\mu_x(i,j,k) + \Delta t \sigma_x^m(i,j,k)\right)\Delta z_e(k)} \\ C_{hxez}(i,j,k) &= -\frac{2\Delta t}{\left(2\mu_x(i) + \Delta t \sigma_x^m(i)\right)\Delta y_e(j)} \\ C_{hxm}(i,j,k) &= -\frac{2\Delta t}{2\mu_x(i,j,k) + \Delta t \sigma_x^e(i,j,k)}. \end{aligned}$$

15.4 Active and passive lumped elements

In the previous section, it is shown that updating equations for nonuniform grid can be obtained by slightly modifying the updating equations of the uniform grid by using the right cell sizes. Readily available updating equations that model lumped elements on a uniform grid also can be modified appropriately and equations for the nonuniform case can be obtained by replacing the constant cell sizes by the indexed cell sizes of electric or magnetic field grids. For instance, for a voltage source in the z-direction the equation that updates E_z becomes (see Chapter 4 (4.10))

$$\begin{aligned} E_z^{n+1}(i,j,k) = {} & C_{eze}(i,j,k) \times E_z^n(i,j,k) \\ & + C_{ezhy}(i,j,k) \times \left(H_y^{n+\frac{1}{2}}(i,j,k) - H_y^{n+\frac{1}{2}}(i-1,j,k)\right) \\ & + C_{ezhx}(i,j,k) \times \left(H_x^{n+\frac{1}{2}}(i,j,k) - H_x^{n+\frac{1}{2}}(i,j-1,k)\right) \\ & + C_{ezs}(i,j,k) \times V_s^{n+\frac{1}{2}}(i,j,k), \end{aligned} \tag{15.11}$$

where

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

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

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

$$C_{ezs}(i,j,k)\ =-\frac{2\Delta t}{\left(2\varepsilon_z(i,j,k)+\Delta t\sigma_z^e(i,j,k)+\dfrac{\Delta z_e(k)\Delta t}{R_s\Delta x_h(i)\Delta y_h(j)}\right)R_s\Delta x_h(i)\Delta y_h(j)}.$$

In (15.11), the cell size Δz_e in the electric field grid is used since the voltage source is along the z direction and conforming to the electric field grid. Other cell sizes Δx_h and Δy_h arise from the finite difference approximations of partial derivatives with respect to x and y, respectively, as discussed before.

Setting the source term in (15.11) to zero results in the updating equation for a resistor with resistance R as (see Chapter 4 (4.16))

$$\begin{aligned}E_z^{n+1}(i,j,k)=\ & C_{eze}(i,j,k)\times E_z^n(i,j,k)\\ &+C_{ezhy}(i,j,k)\times\left(H_y^{n+\frac{1}{2}}(i,j,k)-H_y^{n+\frac{1}{2}}(i-1,j,k)\right)\\ &+C_{ezhx}(i,j,k)\times\left(H_x^{n+\frac{1}{2}}(i,j,k)-H_x^{n+\frac{1}{2}}(i,j-1,k)\right),\end{aligned}\tag{15.12}$$

where

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

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

$$C_{ezhx}(i,j,k)=-\frac{2\Delta t}{\left(2\varepsilon_z(i)+\Delta t\sigma_z^e(i)+\dfrac{\Delta z_e(k)\Delta t}{R\Delta x_h(i)\Delta y_h(j)}\right)\Delta y_h(j)}.$$

The equation for a capacitor with capacitance C can be obtained as (see Chapter 4 (4.20))

$$\begin{aligned} E_z^{n+1}(i,j,k) &= C_{eze}(i,j,k) \times E_z^n(i,j,k) \\ &+ C_{ezhy}(i,j,k) \times \left(H_y^{n+\frac{1}{2}}(i,j,k) - H_y^{n+\frac{1}{2}}(i-1,j,k)\right) \\ &+ C_{ezhx}(i,j,k) \times \left(H_x^{n+\frac{1}{2}}(i,j,k) - H_x^{n+\frac{1}{2}}(i,j-1,k)\right), \end{aligned} \tag{15.13}$$

where

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

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

$$C_{ezhx}(i,j,k) = -\frac{2\Delta t}{\left(2\varepsilon_z(i) + \Delta t\sigma_z^e(i) + \dfrac{2C\Delta z_e(k)}{\Delta x_h(i)\Delta y_h(j)}\right)\Delta y_h(j)}.$$

The equation for an inductor with an inductance L becomes (see Chapter 4 (4.24))

$$\begin{aligned} E_z^{n+1}(i,j,k) &= C_{eze}(i,j,k) \times E_z^n(i,j,k) \\ &+ C_{ezhy}(i,j,k) \times \left(H_y^{n+\frac{1}{2}}(i,j,k) - H_y^{n+\frac{1}{2}}(i-1,j,k)\right) \\ &+ C_{ezhx}(i,j,k) \times \left(H_x^{n+\frac{1}{2}}(i,j,k) - H_x^{n+\frac{1}{2}}(i,j-1,k)\right) \\ &+ C_{ezj}(i,j,k) \times J_{iz}^{n+\frac{1}{2}}(i,j,k), \end{aligned} \tag{15.14}$$

where

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

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

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

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

It should be noted that during the FDTD time-marching iteration, at every time step the new value of $J_{iz}^{n+\frac{1}{2}}(i, j, k)$ should be calculated using

$$J_{iz}^{n+\frac{1}{2}}(i, j, k) = J_{iz}^{n-\frac{1}{2}}(i, j, k) + \frac{\Delta t \Delta z_e(k)}{L \Delta x_h(i) \Delta y_h(j)} E_z^n(i, j, k). \tag{15.15}$$

15.5 Defining objects snapped to the electric field grid

In Chapter 3, Section 3.4, the definition of material parameters is discussed for the case where all objects in the FDTD space are assumed to snap to the uniform electric field grid. The subcell averaging schemes discussed in Section 3.3 can be employed for the case of nonuniform grid as well. Figure 15.7 shows four Yee cells, each of which has a different size and a different permittivity. Since the material component $\varepsilon_z(i, j, k)$ is tangential to the four surrounding media types the equivalent permittivity for $\varepsilon_z(i, j, k)$ can be given as

$$\varepsilon_z(i, j, k) = \frac{\Delta x_1 \Delta y_1 \varepsilon(i-1, j-1, k) + \Delta x_1 \Delta y_2 \varepsilon(i-1, j, k) + \Delta x_2 \Delta y_1 \varepsilon(i, j-1, k) + \Delta x_2 \Delta y_2 \varepsilon(i, j, k)}{(\Delta x_1 + \Delta x_2) \times (\Delta y_1 + \Delta y_2)}. \tag{15.16}$$

Similarly the electric conductivity material components are defined at the same positions as the permittivity components, the same averaging scheme can be employed to get the equivalent electric conductivity $\sigma_z^e(i, j, k)$ as

$$\sigma_z^e(i, j, k) = \frac{\Delta x_1 \Delta y_1 \sigma^e(i-1, j-1, k) + \Delta x_1 \Delta y_2 \sigma^e(i-1, j, k) + \Delta x_2 \Delta y_1 \sigma^e(i, j-1, k) + \Delta x_2 \Delta y_2 \sigma^e(i, j, k)}{(\Delta x_1 + \Delta x_2) \times (\Delta y_1 + \Delta y_2)}. \tag{15.17}$$

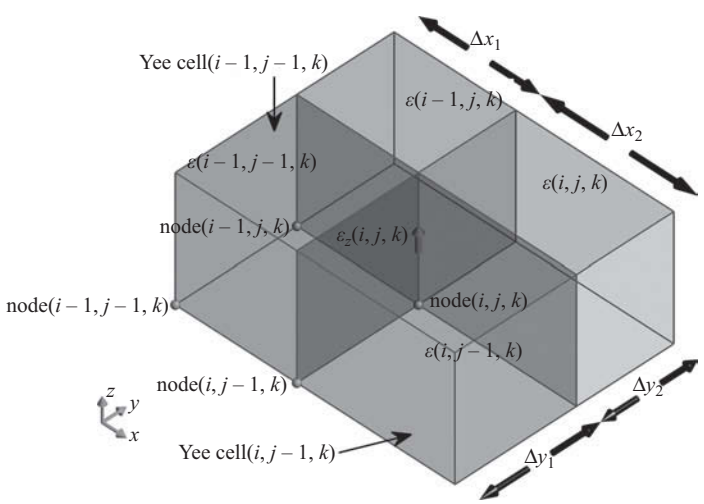

Figure 15.7 Material component $\varepsilon_z(i, j, k)$ located between four Yee cells filled with four different material types.

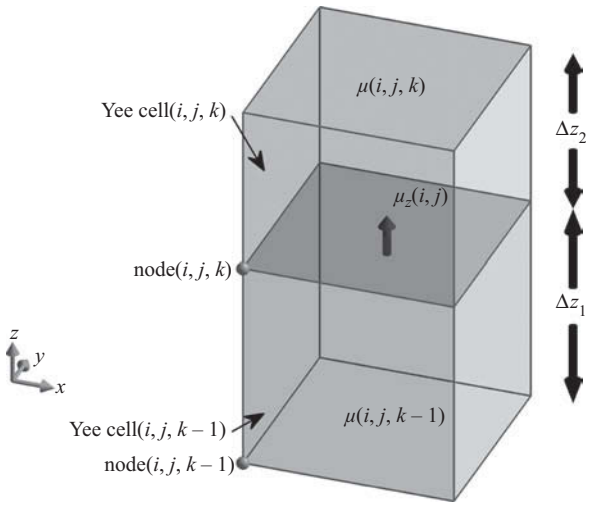

Figure 15.8 Material component $\mu_z(i, j, k)$ located between two Yee cells filled with two different material types.

Since all objects are assumed to snap to the electric field grid, then the material components associated with magnetic field components, permeability and magnetic conductivity, are located between two cells and are oriented normal to the cell boundaries as illustrated in Figure 15.8. In this case, the permeability $\mu_z(i, j, k)$ can be given as

$$\mu_z(i, j, k) = \frac{\mu(i, j, k-1)\mu(i, j, k) \times (\Delta z_1 \Delta z_2)}{\mu(i, j, k-1)\Delta z_2 + \mu(i, j, k) \times \Delta z_1}, \tag{15.18}$$

similarly, the magnetic conductivity $\sigma_z^m(i, j, k)$ becomes

$$\sigma_z^m(i, j, k) = \frac{\sigma^m(i, j, k-1)\sigma^m(i, j, k) \times (\Delta z_1 \Delta z_2)}{\sigma^m(i, j, k-1)\Delta z_2 + \sigma^m(i, j, k) \times \Delta z_1}. \tag{15.19}$$

So far we have discussed the development of the updating equations and construction of the material grids for the case of nonuniform cell sizes. It is straightforward to develop the required equations for other types of calculations within FDTD, such as near-field to far-field transformation, plane-wave incidence etc., since the same principles apply.

15.6 MATLAB® implementation of nonuniform grids

The previously presented base MATLAB program, which uses constant values of Δx, Δy, and Δz over the entire problem space, can be modified to support nonuniform grids. In the following sections, an implementation of nonuniform grids will be presented, which assumes that a problem space is initially constructed based on a uniform coarse grid that uses constant

Δx, Δy, and Δz values as the base cell sizes. Then fine uniform grid subregions are inserted into the main coarse grid and nonuniform transitions are established between the coarse and fine subregions as illustrated in Figure 15.5.

15.6.1 Definition of subregions

In the presented implementation, first it is assumed that the problem space is constructed based on a uniform grid and the base uniform grid cell size is defined as usual by assigning values to Δx, Δy, and Δz in the ***define_problem_space_parameters*** subroutine, as illustrated in Listing 15.1. Then subregions that will be inserted into the base grid are defined in the same file. Listing 15.1 shows definition of five subregions: three along the x direction while the other two along the y direction using a structure named as **subregions**. The field **cell_size**

Listing 15.1 define_problem_space_parameters.m

```
% Dimensions of a unit cell in x, y, and z directions (meters)
dx = 0.002;
dy = 0.002;
dz = 0.00095;

% direction can be 'x', 'y', or 'z'
subregions(1).direction = 'x';
subregions(1).cell_size = 1e-3;
subregions(1).region_start = 27e-3;
subregions(1).region_end   = 33e-3;
subregions(1).transition_length = 5e-3;

subregions(2).direction = 'y';
subregions(2).cell_size = 1e-3;
subregions(2).region_start = 7e-3;
subregions(2).region_end   = 13e-3;
subregions(2).transition_length = 5e-3;

subregions(3).direction = 'x';
subregions(3).cell_size = 1e-3;
subregions(3).region_start = -2e-3;
subregions(3).region_end   = 4e-3;
subregions(3).transition_length = 5e-3;

subregions(4).direction = 'x';
subregions(4).cell_size = 1e-3;
subregions(4).region_start = 56e-3;
subregions(4).region_end   = 62e-3;
subregions(4).transition_length = 5e-3;

subregions(5).direction = 'y';
subregions(5).cell_size = 1e-3;
subregions(5).region_start = 28e-3;
subregions(5).region_end   = 32e-3;
subregions(5).transition_length = 5e-3;
```

defines the size of a cell of the uniform fine subregion along the respective direction. These cells will fill the region determined by fixed coordinates **region_start** and **region_end**. The **transition_length** is the length on which the cell sizes are gradually increased or decreased to establish a smooth transition between the base cell size and the subregion cell size.

15.6.2 Initialization of subregions

As discussed in Chapter 3, one of the first tasks is to determine the size of a problem space, based on the objects in the domain and the boundary definitions, in the subroutine ***initialize_fdtd_material_grid*** by calling the subroutine ***calculate_domain_size***. Then another subroutine, named as ***initialize_subregions***, is called to initialize the subregions.

Listing 15.2 shows implementation of ***initialize_subregions***. Here, the base uniform grid is constructed using the problem space coordinates and dimensions, and the base cell sizes: three arrays are constructed to store the coordinates of the nodes of the electric field grid along x, y, and z directions (**node_coordinates_xe**, **node_coordinates_ye**, and

Listing 15.2 initialize_subregions.m

```
number_of_subregions = 0;
if exist('subregions','var')
   number_of_subregions = size(subregions,2);
end

% number of cells in x, y, and z directions
nx = round(fdtd_domain.size_x/dx);
ny = round(fdtd_domain.size_y/dy);
nz = round(fdtd_domain.size_z/dz);
node_coordinates_xe = (0:nx)*dx + fdtd_domain.min_x;
node_coordinates_ye = (0:ny)*dy + fdtd_domain.min_y;
node_coordinates_ze = (0:nz)*dz + fdtd_domain.min_z;

node_coordinates_xh = (0:nx+1)*dx + fdtd_domain.min_x - dx/2;
node_coordinates_yh = (0:ny+1)*dy + fdtd_domain.min_y - dy/2;
node_coordinates_zh = (0:nz+1)*dz + fdtd_domain.min_z - dz/2;

for ind = 1:number_of_subregions
   switch (subregions(ind).direction)
   case 'x'
   [node_coordinates_xe, node_coordinates_xh] ...
    = insert_subregion_into_domain ...
   (node_coordinates_xe, node_coordinates_xh, subregions(ind), dx);
   case 'y'
   [node_coordinates_ye, node_coordinates_yh] ...
    = insert_subregion_into_domain ...
   (node_coordinates_ye, node_coordinates_yh, subregions(ind), dy);
   case 'z'
   [node_coordinates_ze, node_coordinates_zh] ...
   = insert_subregion_into_domain ...
   (node_coordinates_ze, node_coordinates_zh, subregions(ind), dz);
   end
end
```

```
if isfield(boundary,'is_nonuniform_cpml')
    if (boundary.is_nonuniform_cpml)
        set_nonuniform_cpml_grid;
    end
end

% reset number of cells
nx = size(node_coordinates_xe, 2)-1;
ny = size(node_coordinates_ye, 2)-1;
nz = size(node_coordinates_ze, 2)-1;

% create cell size arrays
cell_sizes_xe = node_coordinates_xe(2:nx+1) - node_coordinates_xe(1:nx);
cell_sizes_ye = node_coordinates_ye(2:ny+1) - node_coordinates_ye(1:ny);
cell_sizes_ze = node_coordinates_ze(2:nz+1) - node_coordinates_ze(1:nz);

% create cell size arrays conforming the magnetic field grid
cell_sizes_xh = node_coordinates_xh(2:nx+2) - node_coordinates_xh(1:nx+1);
cell_sizes_yh = node_coordinates_yh(2:ny+2) - node_coordinates_yh(1:ny+1);
cell_sizes_zh = node_coordinates_zh(2:nz+2) - node_coordinates_zh(1:nz+1);

fdtd_domain.node_coordinates_xe = node_coordinates_xe;
fdtd_domain.node_coordinates_ye = node_coordinates_ye;
fdtd_domain.node_coordinates_ze = node_coordinates_ze;

fdtd_domain.node_coordinates_xh = node_coordinates_xh;
fdtd_domain.node_coordinates_yh = node_coordinates_yh;
fdtd_domain.node_coordinates_zh = node_coordinates_zh;

% some frequently used auxiliary parameters
nxp1 = nx+1;   nyp1 = ny+1;   nzp1 = nz+1;
nxm1 = nx-1;   nxm2 = nx-2;   nym1 = ny-1;
nym2 = ny-2;   nzm1 = nz-1;   nzm2 = nz-2;
```

node_coordinates_ze), whereas three other arrays are constructed to store the coordinates of the nodes of the magnetic field grid (**node_coordinates_xh**, **node_coordinates_yh**, and **node_coordinates_zh**). Then a function, ***insert_subregion_into_domain***, is called to insert nonuniform subregions into the base grid and node coordinate arrays are determined accordingly. After node coordinates are obtained, the distances between the consecutive grid nodes are evaluated and stored as cell size arrays. For instance, **cell_sizes_xe** corresponds to the cells sizes Δx_e in Figure 15.4, whereas, **cell_sizes_xh** corresponds to the cells sizes Δx_h.

Listing 15.3 shows the implementation of ***insert_subregion_into_domain***. In this implementation, first, the cell size of the uniform part of the subregion is adjusted: if the distance between **region_start** and **region_end** is not an integer multiple of the **subregion.cell_size**, then the **subregion.cell_size** is adjusted such that the cells fit between the **region_start** and **region_end**. Then, the length of the transition region that is before the uniform subregion is adjusted: The node of the base electric field grid that is closest to the transition start point is set as the new transition start point. Then, the transition cell sizes are calculated according to the procedure described in Section 15.2.

Listing 15.3 insert_subregion_into_domain.m

```
function [node_coordinates_e, node_coordinates_h] ...
    = insert_subregion_into_domain ...
    (node_coordinates_e, node_coordinates_h, subregion, base_cell_size)

% adjust subregion cell size
subregion_length = ...
    subregion.region_end - subregion.region_start;
n_subregion_cells = ...
    round(subregion_length/subregion.cell_size);
subregion.cell_size = subregion_length/n_subregion_cells;

bcs  = base_cell_size;
srcs = subregion.cell_size;

n_nodes_e = size(node_coordinates_e,2);

% transition before the subregion (indicated with bsr)
% adjust transition length
tr_start_position = (subregion.region_start - ...
    subregion.transition_length);
[mval, I] = min(abs(node_coordinates_e - tr_start_position));
I = find(node_coordinates_e == node_coordinates_e(I(1)));
tr_start_node = I;
tr_length = subregion.region_start ...
    - node_coordinates_e(tr_start_node);

% find number of cells for transition
R = (tr_length + bcs)/(tr_length + srcs);
number_of_transition_cells = ...
    floor(log10(bcs/srcs)/log10(R))-1;

% find transition cell sizes
transition_cell_sizes = ...
    srcs*R.^(1:number_of_transition_cells);

% adjust cell sizes so that total length is equal to
% transition length
transition_cell_sizes = transition_cell_sizes ...
    * (tr_length / sum(transition_cell_sizes));

bsr_transition_cell_sizes = fliplr(transition_cell_sizes);
bsr_tr_start_node = tr_start_node;

% transition after the subregion (indicated with asr)
% adjust transition length
tr_end_position = (subregion.region_end + ...
    subregion.transition_length);
[mval,I] = min(abs(node_coordinates_e - tr_end_position));
I = find(node_coordinates_e == node_coordinates_e(I(1)));
tr_end_node = I(1);
tr_length = node_coordinates_e(tr_end_node) ...
    - subregion.region_end;
```

```
% find number of cells for transition
R = (tr_length + bcs)/(tr_length+srcs);
number_of_transition_cells = ...
    floor(log10(bcs/srcs)/log10(R))-1;

% find transition cell sizes
transition_cell_sizes = ...
    srcs*R.^(1:number_of_transition_cells);

% adjust cell sizes so that total length is equal to
% transition length
transition_cell_sizes = transition_cell_sizes ...
    * (tr_length / sum(transition_cell_sizes));

asr_transition_cell_sizes = transition_cell_sizes;
asr_tr_end_node = tr_end_node;

% create cell size array for subregion
subregion_cell_sizes_e = ...
    srcs * ones(1, n_subregion_cells);

% adjust node coordinates
sr_all_cs_e = [bsr_transition_cell_sizes ...
    subregion_cell_sizes_e asr_transition_cell_sizes];

nodes_before_subregion_e = node_coordinates_e(1:bsr_tr_start_node);
nodes_after_subregion_e = node_coordinates_e(asr_tr_end_node:end);
a_node = node_coordinates_e(bsr_tr_start_node);

n_subregion_cells = size(sr_all_cs_e,2);
subregion_node_coordinates_e = zeros(1, n_subregion_cells-1);

for si = 1:n_subregion_cells-1
    a_node = a_node + sr_all_cs_e(si);
    subregion_node_coordinates_e(si) = a_node;
end

node_coordinates_e = [nodes_before_subregion_e ...
    subregion_node_coordinates_e nodes_after_subregion_e];

n_nodes = size(node_coordinates_e, 2);

% magnetic field components are placed at the centers of the cells
node_coordinates_h = ...
    0.5*(node_coordinates_e(1:n_nodes-1)+node_coordinates_e(2:n_nodes));

node_coordinates_h = [node_coordinates_e(1)-bcs/2 node_coordinates_h ...
    node_coordinates_e(n_nodes)+bcs/2];
```

Similarly, the transition cell sizes are calculated for the transition region after the uniform region part as well. Once the cell sizes in the entire nonuniform subregion are obtained, corresponding nodes are inserted in the positions, where nonuniform subregion overlaps with the base grid, and the electric field grid array is updated. After the coordinates of the nodes in the electric field grid are determined, the coordinates of the nodes of the magnetic field grid

are calculated as the center points of the electric field grid cells. It should be noted that two more cells are padded to the magnetic field grid, each on one side of the grid. Therefore, the first and the last electric field components are located in the middle of these two cells, and the lengths of these two cells can be used to update the corresponding electric field components.

15.6.3 Initialization of updating coefficients

For nonuniform cell sizes the FDTD program code needs to be modified accordingly wherever cell sizes are used. In this section, we will discuss the modification of the basic updating equations to illustrate the application of nonuniform grids. The same logic can be employed to modify other parts of the code as well wherever needed.

The modified implementation of the subroutine ***initialize_updating_coefficients*** is shown in Listing 15.4. For instance, the updating coefficient **Ceyhz** requires a division by Δx.

Listing 15.4 initialize_updating_coefficients.m

```
% General electric field updating coefficients
% Coeffiecients updating Ex
[DX, DY, DZ] = ndgrid(cell_sizes_xe, cell_sizes_yh, cell_sizes_zh);
Cexe  =  (2*eps_r_x*eps_0 - dt*sigma_e_x)./(2*eps_r_x*eps_0 + dt*sigma_e_x);
Cexhz =  (2*dt/DY)./(2*eps_r_x*eps_0 + dt*sigma_e_x);
Cexhy = -(2*dt/DZ)./(2*eps_r_x*eps_0 + dt*sigma_e_x);

% Coeffiecients updating Ey
[DX, DY, DZ] = ndgrid(cell_sizes_xh, cell_sizes_ye, cell_sizes_zh);
Ceye  =  (2*eps_r_y*eps_0 - dt*sigma_e_y)./(2*eps_r_y*eps_0 + dt*sigma_e_y);
Ceyhx =  (2*dt/DZ)./(2*eps_r_y*eps_0 + dt*sigma_e_y);
Ceyhz = -(2*dt/DX)./(2*eps_r_y*eps_0 + dt*sigma_e_y);

% Coeffiecients updating Ez
[DX, DY, DZ] = ndgrid(cell_sizes_xh, cell_sizes_yh, cell_sizes_ze);
Ceze  =  (2*eps_r_z*eps_0 - dt*sigma_e_z)./(2*eps_r_z*eps_0 + dt*sigma_e_z);
Cezhy =  (2*dt/DX)./(2*eps_r_z*eps_0 + dt*sigma_e_z);
Cezhx = -(2*dt/DY)./(2*eps_r_z*eps_0 + dt*sigma_e_z);

% General magnetic field updating coefficients
% Coeffiecients updating Hx
[DX, DY, DZ] = ndgrid(cell_sizes_xh, cell_sizes_ye, cell_sizes_ze);
Chxh  =  (2*mu_r_x*mu_0 - dt*sigma_m_x)./(2*mu_r_x*mu_0 + dt*sigma_m_x);
Chxez = -(2*dt/DY)./(2*mu_r_x*mu_0 + dt*sigma_m_x);
Chxey =  (2*dt/DZ)./(2*mu_r_x*mu_0 + dt*sigma_m_x);

% Coeffiecients updating Hy
[DX, DY, DZ] = ndgrid(cell_sizes_xe, cell_sizes_yh, cell_sizes_ze);
Chyh  =  (2*mu_r_y*mu_0 - dt*sigma_m_y)./(2*mu_r_y*mu_0 + dt*sigma_m_y);
Chyex = -(2*dt/DZ)./(2*mu_r_y*mu_0 + dt*sigma_m_y);
Chyez =  (2*dt/DX)./(2*mu_r_y*mu_0 + dt*sigma_m_y);

% Coeffiecients updating Hz
[DX, DY, DZ] = ndgrid(cell_sizes_xe, cell_sizes_ye, cell_sizes_zh);
Chzh  =  (2*mu_r_z*mu_0 - dt*sigma_m_z)./(2*mu_r_z*mu_0 + dt*sigma_m_z);
Chzey = -(2*dt/DX)./(2*mu_r_z*mu_0 + dt*sigma_m_z);
Chzex =  (2*dt/DY)./(2*mu_r_z*mu_0 + dt*sigma_m_z);
```

Ceyhz is used to update **Ey**, which requires the distance between the **Hz** components before and after **Ey** as illustrated in Figure 15.4. These **Hz** components are on the magnetic field grid, thus the distance between them is denoted as Δx_h, which corresponds to the **cell_sizes_xh** array in the code. Then, **Ceyhz** can be constructed as a three-dimensional array in open form as

```
for i=1:nx+1
for j=1:ny
        for k=1:nz+1
                Ceyhz(i,j,k) = -(2*dt/cell_sizes_xh(i)) ...
                        ./(2*eps_r_y(i,j,k)*eps_0+dt*sigma_e_y(i,j,k));
                end
        end
end
```

Here, if **cell_sizes_xh** can be reshaped as a three-dimensional array of size ($nx + 1$, ny, $nz + 1$), which varies only in the x direction, then the for loops in the above code can be eliminated. For instance, a three-dimensional array, named as **DX**, that has the size ($nx + 1$, ny, $nz + 1$) and values of **cell_sizes_xh** in the x direction is used below to replace the above code:

```
Ceyhz(i,j,k) = -(2*dt/DX)./(2*eps_r_y*eps_0 + dt*sigma_e_y);
```

Similarly, a three-dimensional array, named as **DZ**, that has the size ($nx + 1$, ny, $nz + 1$) and values of **cell_sizes_zh** in the z direction can be used to update **Ceyhx**:

```
Ceyhx = (2*dt/DZ)./(2*eps_r_y*eps_0 + dt*sigma_e_y);
```

The three-dimensional arrays **DX** and **DZ**, used to update **Ceyhz** and **Ceyhx** above, can be constructed simultaneously by using MATLAB function ***ndgrid*** as

```
[DX, DY, DZ] = ndgrid(cell_sizes_xh, cell_sizes_ye, cell_sizes_zh);
```

Here, ***ndgrid*** generates three-dimensional arrays as **DX**, **DY**, and **DZ**, using the size and data of the one-dimensional arrays **cell_sizes_xh**, **cell_sizes_ye**, **cell_sizes_zh**. For instance, the length of **cell_sizes_xh** is $N_x + 1$, the length of **cell_sizes_ye** is N_y, and the length of **cell_sizes_zh** is $N_z + 1$. Then each of **DX**, **DY**, and **DZ** has the size of $(N_x + 1) \times N_y \times (N_z + 1)$. **DX** contains the data of **cell_sizes_xh** listed along its first dimension that is copied $N_y \times (N_z + 1)$ times on other dimensions. **DY** contains the data of **cell_sizes_ye** listed along its second dimension that is copied $(N_x + 1) \times (N_z + 1)$ times on other dimensions. Similarly, **DZ** contains the data of **cell_sizes_zh** listed along its third dimension that is copied $(N_x + 1) \times N_y$ times on other dimensions. One can refer to MATLAB help for further details of the ***ndgrid*** function.

Listing 15.5 initialize_fdtd_parameters_and_arrays.m

```
% Duration of a time step in seconds
min_dx = min(cell_sizes_xe);
min_dy = min(cell_siazes_ye);
min_dz = min(cell_sizes_ze);

dt = 1/(c*sqrt((1/min_dx^2)+(1/min_dy^2)+(1/min_dz^2)));
dt = courant_factor*dt;
```

Similarly, the updating coefficient of a magnetic field component requires the distances between the neighboring electric field components, which lie on the electric field grid, as illustrated in Figure 15.4. Thus, the cell sizes of the electric field grid are used in the updating equations of the magnetic field components as shown in Listing 15.4.

15.6.4 Initialization of time step duration

The Courant-Friedrichs-Lewy (CFL) condition requires that the time increment Δt has a value given by (2.6) as discussed in Section 2.1. If nonuniform grid is used, then the dimensions of the smallest cell need to be used in (2.6) to determine Δt. The subroutine ***initialize_fdtd_parameters_and_arrays*** can be modified as shown in Listing 15.5 such that the minimum values of Δx,Δy, and Δz are obtained from the respective cell size arrays and are used to determine **dt**.

15.7 Simulation examples

Two simulation examples will be presented here. These simulations were executed on personal computer with Intel(R) Core(TM)2 Quad CPU Q9550 running at 2.83 GHz.

15.7.1 Microstrip patch antenna

In this demonstration, the microstrip rectangular patch antenna discussed in Exercise 9.3 is simulated by using a fine grid, a coarse grid, and a nonuniform grid to evaluate the accuracy and computational advantage of the nonuniform grid. Table 15.1 shows the details of these three grids. It should be noted that the base grid of the nonuniform grid simulation is the same as the grid of the coarse grid simulation, which has Δx and Δy as 2 mm. In the nonuniform grid simulation, a fine grid subregion with a cell size of Δx and Δy equal to 1 mm is used around the port and along the four edges of the microstrip patch. Using finer cells improve the accuracy of the input impedance as well as the return loss calculations. Figure 15.9 illustrates the cell view of the patch antenna based on the nonuniform grid, where the fine regions and the nonuniform transitions to the coarse underlying grid can be observed. The definitions of the nonuniform subregions for this particular example are shown in Listing 15.1. In the fine grid simulation, the entire problem space has a cell size of Δx and Δy equal to 1 mm.

Table 15.1 Comparison of fine grid, coarse grid, and nonuniform grid simulations of a microstrip rectangular patch antenna.

Grid type	(Δx, Δy, Δz) (mm)	($N_x \times N_y \times N_z$)	Total number of cells	Number of time steps	Simulation time (minutes)
Fine	(1,1,0.95)	$(96 \times 76 \times 38)$	277,248	8,000	33.0
Coarse	(2,2,0.95)	$(66 \times 56 \times 38)$	140,448	5,755	10.6
Nonuniform	(2,2,0.95)	$(77 \times 62 \times 38)$	181,412	8,000	19.6

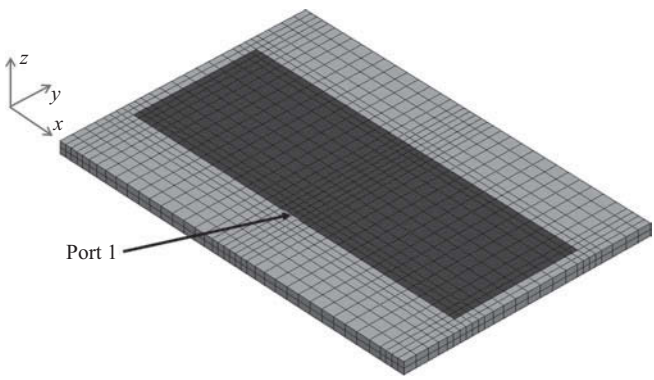

Figure 15.9 Cell view of the microstrip rectangular patch antenna.

The fine grid and the nonuniform grid simulations are performed for 8,000 time steps, while the coarse grid simulation is performed for 5,755 time steps using a Gaussian pulse with $\tau = 3.336 \times 10^{-11}$ as a source. It should be noted that in the fine grid and the nonuniform grid simulations the minimum cell sizes are the same, thus the duration of a time step, Δt, that is dictated by the CFL stability condition, is the same. Since the minimum cell size in the coarse grid simulation is larger, the time step is larger. The different numbers of time steps chosen in these simulations are necessary in order to simulate the transient response of these three cases for the same duration, which is 13.62 ns.

Each of these three cases is simulated and computation time is recorded in Table 15.1. Compared with the coarse grid simulation, the computational overhead of nonuniform grid is significantly less than that of the fine grid.

Figure 15.10 shows the comparison of the power reflection coefficient of fine grid, coarse grid, and nonuniform grid simulations of this patch antenna. Here we can use the fine grid simulation result as a reference for accuracy comparison with the coarse and the nonuniform grids. The results imply that accuracy of simulations can be improved by modeling critical areas of a geometry using fine grid subregions, thus employing nonuniform grids, compared with coarse grid simulations with the overhead of increased memory requirement and computation time, which are still significantly less than the fine grid requirements.

15.7.2 Three-pole microstrip low-pass filter

The second example is a three-pole low-pass filter presented in [78]. Dimensions of this filter are shown in Figure 15.11. The boundaries of the problem space are terminated by PEC on

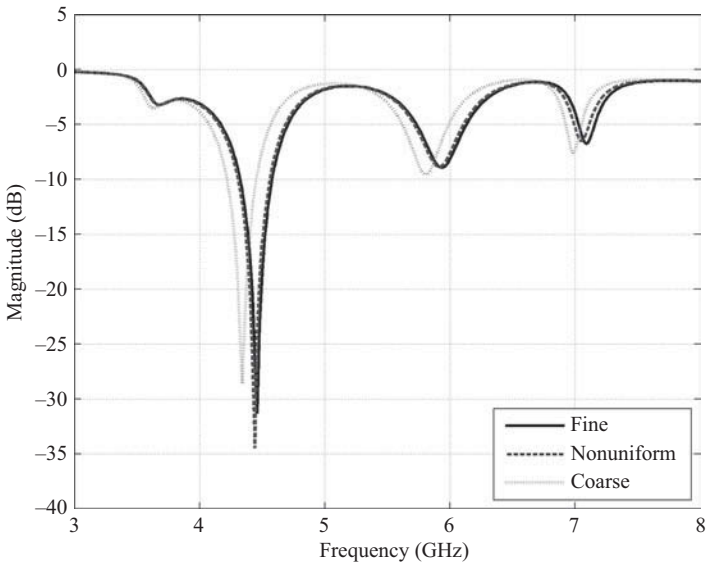

Figure 15.10 Comparison of power reflection coefficient of fine grid, coarse grid, and nonuniform grid simulations of a microstrip rectangular patch antenna.

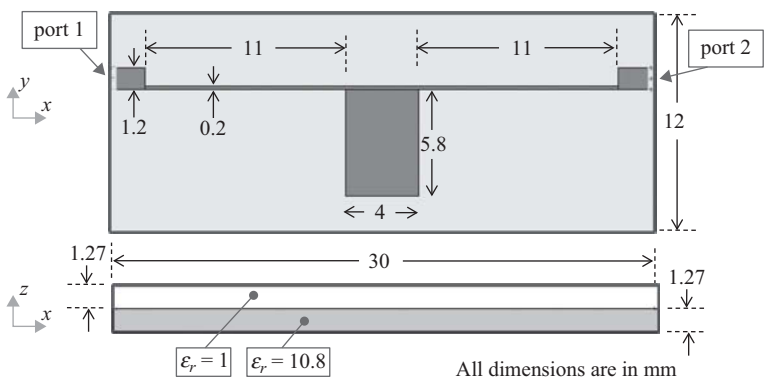

Figure 15.11 Geometry and dimensions of a three-pole low-pass filter.

all sides. The most critical dimension in this circuit is the 0.2 mm wide microstrip lines that connect the center stub to the wide feeding lines on each side. Similar to the previous example, the circuit is simulated by using a fine grid, a coarse grid, and a nonuniform grid to evaluate the performance of the nonuniform grid.

Table 15.2 shows the details of these three grids. Here, the base grid of the nonuniform grid simulation is the same as the grid of the coarse grid simulation. In the nonuniform grid simulation, as shown in Listing 15.6, a fine grid subregion with a cell size of Δy equal to 0.1 mm is used around the narrow strip to model that region more accurately.

Table 15.2 Comparison of fine grid, coarse grid, and nonuniform grid simulations of a three-pole low-pass microstrip filter.

Grid type	(Δx, Δy, Δz) (mm)	($N_x \times N_y \times N_z$)	Total number of cells	Number of time steps	Simulation time (minutes)
Fine	(0.1,0.1,0.127)	$(300 \times 120 \times 20)$	720,000	200,000	1,800
Coarse	(0.2,0.2,0.127)	$(150 \times 60 \times 20)$	180,000	130,764	275
Nonuniform	(0.2,0.2,0.127)	$(150 \times 76 \times 20)$	228,000	168,966	460

Listing 15.6 define_problem_space_parameters.m

```
% Dimensions of a unit cell in x, y, and z directions (meters)
dx = 0.2e-3;
dy = 0.2e-3;
dz = 0.127e-3;

% direction can be 'x', 'y', or 'z'
subregions(1).direction = 'y';
subregions(1).cell_size = 0.1e-3;
subregions(1).region_start = 7.4e-3;
subregions(1).region_end   = 9.4e-3;
subregions(1).transition_length = 2e-3;
```

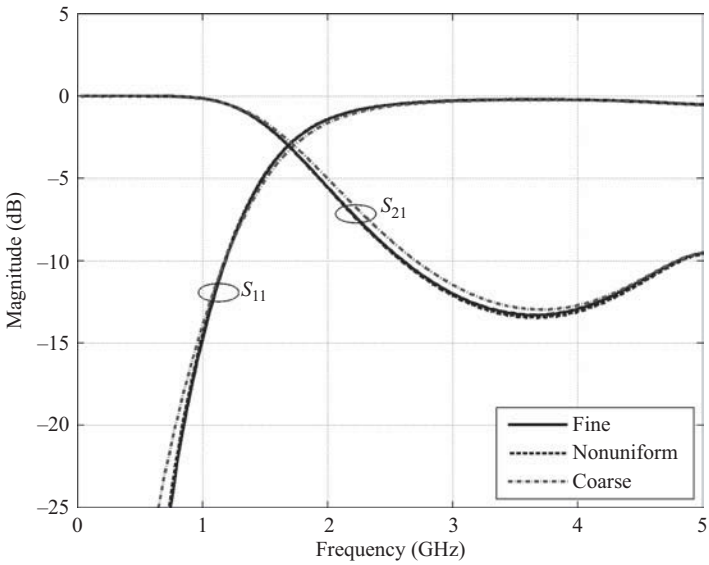

Figure 15.12 Comparison of scattering parameters of fine grid, coarse grid, and nonuniform grid simulations of a three-pole low-pass microstrip filter.

Each of these three cases is simulated for 37 ns using a Gaussian pulse with $\tau = 4.2362 \times 10^{-12}$ as a source. The number of time steps and total computation times are recorded in Table 15.2. Time results show that although the nonuniform simulation takes longer time than the coarse simulation, it takes much shorter time than the fine grid simulation. Figure 15.12 shows the scattering parameters of these three cases compared with each other, which further justifies the benefit of appropriate use of fine grid subregions through nonuniform grids where needed despite the slightly increased computational overhead.