Lecture #15

Implementation of Two-Dimensional FDTD

Lecture Outline

• Review
  – Update equations with PML
  – Code development sequence
• Numerical Boundary Conditions for 3D
• Reduction to Two-Dimensions
• Calculating the PML Parameters
• Implementation for Ez Mode
• Total-Field/Scattered-Field Source
Uniaxial PML

Reflections can be prevented at all angles, all frequencies, and for all polarizations if we allow our absorbing material to be diagonally anisotropic.
Derivation of Update Equation for $H_x$

1. Governing Equation in the Frequency-Domain
   \[
   \frac{d}{dt} \left( \frac{-1}{\varepsilon_0} \nabla \cdot ( \sigma_{y} \nabla \alpha_1 ) \nabla \cdot ( \sigma_{z} \nabla \alpha_3 ) \right) H_x(\omega) = \frac{\sigma_{y}}{\mu} \frac{\partial \alpha_1}{\partial y} + \frac{\sigma_{z}}{\mu} \frac{\partial \alpha_3}{\partial z}
   \]

2. Prepare equation for conversion to time-domain
   \[
   \frac{d}{dt} H_x(\omega) + \frac{\sigma_{y}}{\mu} \frac{\partial \alpha_1}{\partial y} + \frac{\sigma_{z}}{\mu} \frac{\partial \alpha_3}{\partial z} - \frac{1}{j \omega} \frac{\partial}{\partial t} \left( \frac{\sigma_{y}}{\mu} C_1(\omega) \right) - \frac{1}{j \omega} \frac{\partial}{\partial t} \left( \frac{\sigma_{z}}{\mu} C_3(\omega) \right)
   \]

3. Convert each term to the time-domain
   \[
   \frac{\partial}{\partial t} \left( \frac{\sigma_{y}}{\mu} C_1(\omega) \right) - \frac{1}{j \omega} \frac{\partial}{\partial t} \left( \frac{\sigma_{z}}{\mu} C_3(\omega) \right)
   \]

4. Approximate the equation for numerical implementation
   \[
   H_x^{[1/\Delta t]} = \left( \sigma_{y}^{[1/\Delta t]} + \sigma_{z}^{[1/\Delta t]} \right) H_x^{[1/\Delta t]} + \frac{1}{4} \left( \frac{\sigma_{y}^{[1/\Delta t]} + \sigma_{z}^{[1/\Delta t]} \Delta t}{\varepsilon_0} \right) \sum_{i=1}^{N} \left( C_i^{[1/\Delta t]} \right) - \frac{\sigma_{y}^{[1/\Delta t]} + \sigma_{z}^{[1/\Delta t]} \Delta t}{\varepsilon_0} \sum_{i=1}^{N} \left( C_i^{[1/\Delta t]} \right)
   \]

5. Solve for $H$ at the future time value
   \[
   H_x^{[1/\Delta t]} = \left( m_{in} \right) H_x^{[1/\Delta t]} + \left( m_{in} \right)^T C_i^{[1/\Delta t]} + \left( m_{in} \right)^T I_e \left( m_{in} \right) + \left( m_{in} \right)^T I_w \left( m_{in} \right)
   \]

**Code Development Sequence**

- **Step 1** – Basic Update + Dirichlet
- **Step 2** – Basic Update + Periodic BC
- **Step 3** – Add PML
- **Step 4** – TF/SF
- **Step 5** – Calculate Response
- **Step 6** – Add a Device and Benchmark
Lecture 15 Slide 7

**Numerical Boundary Conditions for 3D**

Boundary Conditions

We have formulated our update equations so that the curl can be calculated separate from the update equation. Boundary condition problems occur only in these equations.

\[
\nabla \times \vec{E} \quad \nabla \times \vec{H}
\]

\[
C^{E}_{i,k} = \frac{\vec{E}_{i,k+1}^{\text{new}} - \vec{E}_{i,k}^{\text{old}}}{\Delta y} - \frac{\vec{E}_{i,k}^{\text{old}} - \vec{E}_{i,k-1}^{\text{old}}}{\Delta z} \\
C^{E}_{i,k} = \frac{\vec{E}_{i+1,k}^{\text{old}} - \vec{E}_{i,k}^{\text{old}}}{\Delta x} - \frac{\vec{E}_{i,k}^{\text{old}} - \vec{E}_{i-1,k}^{\text{old}}}{\Delta y}
\]

Special boundary conditions are required at the high grid boundaries when computing the curl of the \( E \) field.

\[
C^{H}_{i,k} = \frac{\vec{H}_{i,k+1}^{\text{new}} - \vec{H}_{i,k}^{\text{old}}}{\Delta y} - \frac{\vec{H}_{i,k}^{\text{old}} - \vec{H}_{i,k-1}^{\text{old}}}{\Delta z} \\
C^{H}_{i,k} = \frac{\vec{H}_{i+1,k}^{\text{old}} - \vec{H}_{i,k}^{\text{old}}}{\Delta x} - \frac{\vec{H}_{i,k}^{\text{old}} - \vec{H}_{i-1,k}^{\text{old}}}{\Delta y}
\]

Special boundary conditions are required at the low grid boundaries when computing the curl of the \( H \) field.
Handling the Boundary Conditions

Boundary conditions are handled by calculating the curl terms at the boundaries separate from the rest of the curl equation. This should be done explicitly without using ‘if’ statements.

For example,

\[ C^E_{i,j,k} = \begin{cases} 
\frac{E_{i,j+1,k} - E_{i,j,k}}{\Delta y} - \frac{E_{i,j,k+1} - E_{i,j,k}}{\Delta z} & j < N_y \text{ and } k < N_z \\
\frac{E_{i,j,k} - E_{i,j,N_z}}{\Delta y} - \frac{E_{i,j+1}^{N_y,N_z,N_z} - E_{i,j}^{N_y,N_z,N_z}}{\Delta z} & j = N_y \text{ and } k < N_z \\
\frac{E_{i,j,k} - E_{i,j,N_y}}{\Delta y} - \frac{E_{i,j,k+1}^{N_y,N_z} - E_{i,j}^{N_y,N_z}}{\Delta z} & j < N_y \text{ and } k = N_z \\
\frac{E_{i,j,N_z} - E_{i,j,k}}{\Delta y} - \frac{E_{i,j,N_z}^{N_y,N_z} - E_{i,j}^{N_y,N_z}}{\Delta z} & j = N_y \text{ and } k = N_z 
\end{cases} \]

Calculating the curl terms separate from the update equations is an excellent way to isolate and modularize the boundary condition problem.

MATLAB Code Implementation (1 of 3)

First, we blindly try to calculate the curl.

```matlab
% Compute CEx
for nx = 1 : Nx
    for ny = 1 : Ny
        for nz = 1 : Nz
            CEx(nx,ny,nz) = (Ez(nx,ny+1,nz) - Ez(nx,ny,nz))/dy ... 
                           - (Ey(nx,ny,nz+1) - Ey(nx,ny,nz))/dz;
        end
    end
end
```

We see that we run into two problems at the y-hi and z-hi sides of the grid.
MATLAB Code Implementation (2 of 3)

Next, we handle the z-hi problem explicitly outside of the nz-loop. We copy the code inside the nz-loop and past it after the nz-loop.

% Compute CEx
for nx = 1 : Nx
    for ny = 1 : Ny
        for nz = 1 : Nz-1
            CEx(nx,ny,nz) = (Ez(nx,ny+1,nz) - Ez(nx,ny,nz))/dy ...  
                - (Ey(nx,ny,nz+1) - Ey(nx,ny,nz))/dz;
        end
        CEx(nx,ny,Nz) = (Ez(nx,ny+1,Nz) - Ez(nx,ny,Nz))/dy ...  
                - (Ey(nx,ny,1) - Ey(nx,ny,Nz))/dz;
    end
end

We still have the problem at the y-hi side of the grid, but now the problem occurs in two places.

MATLAB Code Implementation (3 of 3)

Finally, we handle the y-hi problem explicitly outside of the ny-loop. We copy all of the code inside the ny-loop and paste it after the ny-loop.

% Compute CEx
for nx = 1 : Nx
    for ny = 1 : Ny-1
        for nz = 1 : Nz-1
            CEx(nx,ny,nz) = (Ez(nx,ny+1,nz) - Ez(nx,ny,nz))/dy ...  
                - (Ey(nx,ny,nz+1) - Ey(nx,ny,nz))/dz;
        end
        CEx(nx,ny,Nz) = (Ez(nx,ny+1,Nz) - Ez(nx,ny,Nz))/dy ...  
                - (Ey(nx,ny,1) - Ey(nx,ny,Nz))/dz;
    end
end
for nz = 1 : Nz-1
    CEx(nx,Ny,nz) = (Ez(nx,1,nz) - Ez(nx,Ny,nz))/dy ...  
                - (Ey(nx,Ny,nz+1) - Ey(nx,Ny,nz))/dz;
end
CEx(nx,Ny,Nz) = (Ez(nx,1,Nz) - Ez(nx,Ny,Nz))/dy ...  
                - (Ey(nx,Ny,1) - Ey(nx,Ny,Nz))/dz;
end
Reduction to Two Dimensions

Sometimes it is possible to describe a physical device using just two dimensions. Doing so dramatically reduces the numerical complexity of the problem and is ALWAYS GOOD PRACTICE.
2D Grids are Infinite in the 3rd Dimension

Anything represented on a 2D grid, is actually a device that is of infinite extent along the 3rd dimension.

What is Different in Two Dimensions?

- We will assume it is the z direction that is uniform.
- All derivatives in the z direction are zero
  \[
  \frac{\partial}{\partial z} = 0
  \]
- We do not need a PML to terminate the z axis
  \[
  \sigma_z'(z) = 0 \\
  s_z(z) = 1
  \]
Revisions for Update Equation for $H_x$

The update coefficients are computed before the main FDTD loop.

$$m_{x_{1,i,j}}^{i,j} = \frac{1}{\Delta t} \left( \frac{\sigma^f_j}{2\varepsilon_0} \right)$$

$$m_{x_{2,i,j}}^{i,j} = \frac{1}{\Delta t} \left( \frac{\sigma^f_j}{2\varepsilon_0} \right)$$

$$m_{x_{3,i,j}}^{i,j} = \frac{1}{\Delta t} \left( \frac{\sigma^f_j}{2\varepsilon_0} \right)$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{CE}^{i,j} = \sum_{j=0}^{N} C_{E}^{i,j}$$

$$C_{E}^{i,j} = \frac{\bar{E}_{i+1}^{i,j} - \bar{E}_{i}^{i,j}}{\Delta y}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$H_{x_{1,i,j}}^{i,j} = \left( m_{x_{1,i,j}}^{i,j} \right) H_{x_{2,i,j}}^{i,j} + \left( m_{x_{2,i,j}}^{i,j} \right) C_{E}^{i,j} + \left( m_{x_{3,i,j}}^{i,j} \right) I_{CE}^{i,j}$$

2D Update Equation for $H_x$

The update coefficients are computed before the main FDTD loop.

$$m_{x_{1,i,j}}^{i,j} = \frac{1}{\Delta t} \left( \frac{\sigma^f_j}{2\varepsilon_0} \right)$$

$$m_{x_{2,i,j}}^{i,j} = \frac{1}{\Delta t} \left( \frac{\sigma^f_j}{2\varepsilon_0} \right)$$

$$m_{x_{3,i,j}}^{i,j} = \frac{1}{\Delta t} \left( \frac{\sigma^f_j}{2\varepsilon_0} \right)$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{CE}^{i,j} = \sum_{j=0}^{N} C_{E}^{i,j}$$

$$C_{E}^{i,j} = \frac{\bar{E}_{i+1}^{i,j} - \bar{E}_{i}^{i,j}}{\Delta y}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$H_{x_{1,i,j}}^{i,j} = \left( m_{x_{1,i,j}}^{i,j} \right) H_{x_{2,i,j}}^{i,j} + \left( m_{x_{2,i,j}}^{i,j} \right) C_{E}^{i,j} + \left( m_{x_{3,i,j}}^{i,j} \right) I_{CE}^{i,j}$$
2D Update Equation for $H_y$

The update coefficients are computed before the main FDTD loop.

$$m_{x0}^{i,j} = \frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

$$m_{y0}^{i,j} = \frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

$$m_{x0}^{i,j} = -\frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

$$m_{y1}^{i,j} = -\frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{C_{x0}^{i,j}} = \sum_{k=i}^{i+1} C_{x0}^{k,j}$$

$$C_{x0}^{i,j} = -\frac{E^{i-1,j} - E^{i,j}}{\Delta x}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$H_{x1}^{i,j} = \left( m_{x0}^{i,j} \right) H_{x0}^{i,j} + \left( m_{y0}^{i,j} \right) C_{y0}^{i,j} + \left( m_{y1}^{i,j} \right) I_{C_{x0}^{i,j}}$$

2D Update Equation for $H_z$

The update coefficients are computed before the main FDTD loop.

$$m_{z0}^{i,j} = \frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

$$m_{z0}^{i,j} = \frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

$$m_{z0}^{i,j} = -\frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

$$m_{z0}^{i,j} = -\frac{1}{\Delta} \left( \sigma_0^{i,j} \right)$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{H_{z0}^{i,j}} = \sum_{k=j}^{j+1} H_{z0}^{k,j}$$

$$C_{z0}^{i,j} = \frac{E^{i,j-1} - E^{i,j}}{\Delta x} - \frac{E^{i+1,j} - E^{i,j}}{\Delta y}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$H_{z1}^{i,j} = \left( m_{z0}^{i,j} \right) H_{z0}^{i,j} + \left( m_{z0}^{i,j} \right) C_{z0}^{i,j} + \left( m_{z0}^{i,j} \right) I_{H_{z0}^{i,j}}$$
2D Update Equation for $D_x$

The update coefficients are computed before the main FDTD loop.

$$m_{j+1/2}^{i+1/2} = \frac{1}{\Delta t} \left[ \frac{\sigma_x}{2\epsilon_0} \right]$$

$$m_{j+1/2}^{i-1/2} = \frac{1}{\Delta t} \left[ \frac{\sigma_x}{2\epsilon_0} \right]$$

$$m_{j-1}^{i+1/2} = \frac{c_0}{m_{j+1/2}^{i+1/2}}$$

$$m_{j-1}^{i-1/2} = \frac{c_0\Delta t \sigma_x^{i+1/2}}{\epsilon_0}$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{c,j}^{i+1/2} = \sum_{y=-N/2}^{N/2} C_{j,y}^{i}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$\hat{D}_x^{i+1} = \left( m_{Dx}^{i+1/2} \right) \hat{D}_x^{i} + \left( m_{Dx}^{i-1/2} \right) C_{x}^{i} \frac{\hat{D}_x^{i}}{2} + \left( m_{Dx}^{i} \right) I_{c,j}^{i+1/2}$$

2D Update Equation for $D_y$

The update coefficients are computed before the main FDTD loop.

$$m_{j+1/2}^{i+1/2} = \frac{1}{\Delta t} \left[ \frac{\sigma_y}{2\epsilon_0} \right]$$

$$m_{j+1/2}^{i-1/2} = \frac{1}{\Delta t} \left[ \frac{\sigma_y}{2\epsilon_0} \right]$$

$$m_{j-1}^{i+1/2} = \frac{c_0}{m_{j+1/2}^{i+1/2}}$$

$$m_{j-1}^{i-1/2} = \frac{c_0\Delta t \sigma_y^{i+1/2}}{\epsilon_0}$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{c,j}^{i+1/2} = \sum_{y=-N/2}^{N/2} C_{j,y}^{i}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$\hat{D}_y^{i+1} = \left( m_{Dy}^{i+1/2} \right) \hat{D}_y^{i} + \left( m_{Dy}^{i-1/2} \right) C_{y}^{i} \frac{\hat{D}_y^{i}}{2} + \left( m_{Dy}^{i} \right) I_{c,j}^{i+1/2}$$
2D Update Equation for $D_z$

The update coefficients are computed before the main FDTD loop.

$$m_{u0j}^{i,j} = \frac{1}{\Delta t} \left[ \sum_{\sigma=r}^{\sigma=r} \sigma_r^{i,j} + \frac{\sigma_r^{i,j}}{2\epsilon_0} + \frac{\sigma_r^{i,j}}{4\epsilon_0} \right]$$

$$m_{u0i}^{i,j} = \frac{1}{\Delta t} \left[ \sum_{\sigma=r}^{\sigma=r} \sigma_r^{i,j} - \frac{\sigma_r^{i,j}}{2\epsilon_0} - \frac{\sigma_r^{i,j}}{4\epsilon_0} \right]$$

$$m_{u0j}^{i,j} = -\frac{c_0^{i,j}}{m_{u0j}^{i,j}}$$

$$m_{u0i}^{i,j} = -\frac{\Delta t}{c_0^{i,j}} \left[ \sigma_r^{i,j} \right]$$

The integration terms are computed inside the main FDTD loop, but before the update equation.

$$I_{z0}^{i,j} = \sum_{j=0}^{J} D_z^{j,j}$$

$$C_{z}^{i,j} = \frac{H_t^{j,j}}{\Delta t} - \frac{H_t^{j,j}}{\Delta y}$$

The update equation is computed inside the main FDTD loop immediately after the integration terms are updated.

$$D_z^{i,j}_{+z} = \left( m_{u0i}^{i,j} \right) D_z^{i,j} + \left( m_{u0i}^{i,j} \right) C_{z}^{i,j}_{+z}$$

$$I_{z0}^{i,j}_{+z}$$

---

2D Update Equations for $E_x$, $E_y$, and $E_z$

The update coefficients are computed before the main FDTD loop.

$$m_{uxt}^{i,j} = \frac{1}{\epsilon_0 x|^{i,j}}$$

$$m_{uyt}^{i,j} = \frac{1}{\epsilon_0 y|^{i,j}}$$

$$m_{uzt}^{i,j} = \frac{1}{\epsilon_0 z|^{i,j}}$$

The update equations are computed inside the main FDTD loop.

$$E_x^{i,j}_{+x} = \left( m_{uxt}^{i,j} \right) D_x^{i,j}_{-x}$$

$$E_y^{i,j}_{+y} = \left( m_{uyt}^{i,j} \right) D_y^{i,j}_{-y}$$

$$E_z^{i,j}_{+z} = \left( m_{uzt}^{i,j} \right) D_z^{i,j}_{-z}$$

No changes here!
The update equation for $H_i$ was

$$H_i^{[i]}_{i-\Delta t/2} = \left( m_{H_1}^{[i]} \right) H_i^{[i]}_{i-\Delta t/2} + \left( m_{H_2}^{[i]} \right) C_{H_{1}}^{[i]} + \left( m_{H_3}^{[i]} \right) I_{CE_{H_{1}}}^{[i]}$$

This could be implemented in MATLAB as follows

```matlab
for ny = 1 : Ny
    for nx = 1 : Nx
        Hx(nx,ny) = mHx1(nx,ny)*Hx(nx,ny) + mHx2(nx,ny)*CEx(nx,ny) + mHx3(nx,ny)*ICEx(nx,ny);
    end
end
```

However, it is much simpler and faster to implement it using “vectorized” MATLAB commands.

```matlab
Hx = mHx1.*Hx + mHx2.*CEx + mHx3.*ICEx;
```

**Calculating the PML Parameters**
**MATLAB Code to Calculate PML Parameters**

To calculate the PML parameters $\sigma_x$ and $\sigma_y$, we employ the $2\times$ grid concept.

First, we calculate $\sigma_x$ and $\sigma_y$ on the $2\times$ grid.

```matlab
% COMPUTE PML PARAMETERS
Nx2 = 2*Nx;
Ny2 = 2*Ny;
sigx = zeros(Nx2,Ny2);
for nx = 1 : 2*NPML(1)
    nx1 = 2*NPML(1) - nx + 1;
    sigx(nx1,:) = (0.5*e0/dt)*(nx/2/NPML(1))^3;
end
for nx = 1 : 2*NPML(2)
    nx1 = Nx2 - 2*NPML(2) + nx;
    sigx(nx1,:) = (0.5*e0/dt)*(nx/2/NPML(2))^3;
end

sigy = zeros(Nx2,Ny2);
for ny = 1 : 2*NPML(3)
    ny1 = 2*NPML(3) - ny + 1;
    sigy(:,ny1) = (0.5*e0/dt)*(ny/2/NPML(3))^3;
end
for ny = 1 : 2*NPML(4)
    ny1 = Ny2 - 2*NPML(4) + ny;
    sigy(:,ny1) = (0.5*e0/dt)*(ny/2/NPML(4))^3;
end
```

This $2'$ is here because we are calculating the fictitious conductivity terms on the $2\times$ grid, but NPML contains the size of the PML on the $1\times$ grid.

**Visualizing the PML Parameters**

If calculated correctly, the PML conductivity terms should look like:

\[
\sigma'_x(x) = \frac{\varepsilon_0}{2\Delta t} \left( \frac{x}{L_x} \right)^3
\]

\[
\sigma'_y(y) = \frac{\varepsilon_0}{2\Delta t} \left( \frac{y}{L_y} \right)^3
\]

$e_0 = 8.8542\times10^{-12}$

$dt = 1.6678\times10^{-10}$
Next, we overlay the PML functions onto the 1x grid to calculate the update coefficients containing PML terms.

% COMPUTE UPDATE COEFFICIENTS
sigHx = sigx(1:2:Nx2,2:2:Ny2);
sigy = sigy(1:2:Nx2,2:2:Ny2);
mHx0 = (1/dt) + sigHy/(2*e0);
mHx1 = ((1/dt) - sigHy/(2*e0))./mHx0;
mHx2 = - c0./URxx./mHx0;
mHx3 = - (c0*dt/e0) * sigHx./URxx ./ mHx0;
sigDx = sigx(2:2:Nx2,1:2:Ny2);
sigDy = sigy(2:2:Nx2,1:2:Ny2);
mHy0 = (1/dt) + sigHx/2*e0;
mHy1 = ((1/dt) - sigHx/2*e0))./mHy0;
mHy2 = - c0./URyy./mHy0;
mHy3 = - (c0*dt/e0) * sigHx./URyy ./ mHy0;
sigDz = sigx(1:2:Nx2,1:2:Ny2);
sigDz = sigy(1:2:Nx2,1:2:Ny2);
ndz0 = (1/dt) + (sigDx + sigDy)/(2*e0) ...
+ sigDx.*sigDy*(dt/4*e0^2);
ndz1 = (1/dt) - (sigDx + sigDy)/(2*e0) ...
- sigDx.*sigDy*(dt/4*e0^2);
ndz2 = ndz1 ./ ndz0;
ndz4 = - (dt/e0^2)*sigDx.*sigDy./ndz0;

Implementation

$E_z$ Mode
Block Diagram for $E_z$ Mode (1 of 2)

Define Device Parameters
Define FDTD Parameters
Compute Grid Parameters
Build Device on Grid
Compute Time Step
Compute Source
Compute PML Parameters

Initialization...

Compute Update Coefficients
mHx1, mHx2, mHx3
mHy1, mHy2, mHy3
mDz1, mDz2, mDz4
mEz1

Initialize Fields
Hx, Hy, Dz, Ez

Initialize Curl Arrays
CEx, CEy, CHz

Initialize Integration Arrays
ICEx, ICEy, IDz

Block Diagram for $E_z$ Mode (2 of 2)

Build Device on Grid

Initialization...

Compute Update Coefficients
mHx1, mHx2, mHx3
mHy1, mHy2, mHy3
mDz1, mDz2, mDz4
mEz1

Initialize Fields
Hx, Hy, Dz, Ez

Initialize Curl Arrays
CEx, CEy, CHz

Initialize Integration Arrays
ICEx, ICEy, IDz

Main loop...

Computing curl of E
CEx, CEy

Updating H integrations
ICEx = ICEx + CEx
ICEy = ICEy + CEy

Computing curl of H
CHz

Updating D integrations
IDz = IDz + Dz

Updating D
Dz = mDz1.*Dz + mDz2.*CHz + mDz4.*IDz;

Injecting source
Dz(nxs,nys) = Dz(nxs,nys) + g(T);

Updating E
Ez = mEz1.*Dz;

Visualizing fields

Done? yes
Finished!

BC's

Done? no
Block Diagram for $H_z$ Mode (1 of 2)

- Define Device Parameters
- Define FDTD Parameters
- Compute Grid Parameters
- Compute Time Step
- Compute Source
- Compute PML Parameters
- Compute Update Coefficients
  - $mHz_1$, $mHz_2$, $mHz_4$
  - $mDx_1$, $mDx_2$, $mDx_3$
  - $mDy_1$, $mDy_2$, $mDy_3$
  - $mEx_1$, $mEy_1$
- Initialize Fields
  - $Hz$, $Dx$, $Dy$, $Ex$, $Ey$
- Initialize Curl Arrays
  - $CEx$, $CEy$, $CHz$
- Initialize Integration Arrays
  - $ICHx$, $ICHy$, $IHz$

Initialization...

Update $E$ Field

$$Ex = mEx_1 .* Dx; \quad Ey = mEy_1 .* Dy;$$

Main loop...

Update $H$ Field

$$Hz = mHz_1 .* Hz + mHz_2 .* CEz + mHz_4 .* IHz;$$

Update D Field

$$Dx = mDx_1 .* Dx + mDx_2 .* CHx + mDx_3 .* ICHx; \quad Dy = mDy_1 .* Dy + mDy_2 .* CHy + mDy_3 .* ICHy;$$

Inject Source

$$Hz(nxs,nys) = Hz(nxs,nys) + g(T);$$

Visualize Fields

BC’s

Done? yes

Finished!
The total-field/scattered-field (TF/SF) condition is a technique to inject a “one-way” source.

Benefits
- Eliminates backward propagating power
  - Needed for calculation of reflection
  - Minimizes power incident on PML
- Ensures waves at the boundaries are only travelling outward
- 100% of power injected by the source is incident on the device being simulated.
Typical 2D FDTD Grid Layout For Modeling Periodic Structures

The position of the TF/SF source really can be anywhere above the device. It cannot slice through the device.

Typical 2D FDTD Grid Layout For Modeling General Scatterers
The Total-Field/Scattered-Field Framework

Problem Points?

2D FDTD Grid

We must subtract the source from \( \mathbf{j}_{\text{src}} \) to make it look like a scattered-field quantity.

Correction to Finite-Difference Equations at the Problem Cells (1 of 2)

On the scattered-field side of the TF/SF interface, the finite-difference equation contains a term from the total-field side. Due to the staggered nature of the Yee grid, this only occurs in the update equation for a magnetic field. In fact, this only occurs in the computation of the curl of \( \mathbf{E} \) used in the \( \mathbf{H} \) field update equations.

\[
C_E \left[ \begin{array}{c} E_x^{i+1/2,m-1} \\ E_y^{i+1/2,m-1} \\ E_z^{i+1/2,m-1} \\ E_{x,m}^{i+1/2} \\ E_{y,m}^{i+1/2} \\ E_{z,m}^{i+1/2} \\ \end{array} \right] = \frac{\Delta y}{\Delta} \left( \begin{array}{c} \frac{E_x^{i+1/2,m-1} - E_x^{i-1/2,m-1}}{\Delta y} \\ \frac{E_y^{i+1/2,m-1} - E_y^{i-1/2,m-1}}{\Delta y} \\ \frac{E_z^{i+1/2,m-1} - E_z^{i-1/2,m-1}}{\Delta y} \\ \frac{1}{\Delta y} E_{x,m}^{i+1/2} \\ \frac{1}{\Delta y} E_{y,m}^{i+1/2} \\ \frac{1}{\Delta y} E_{z,m}^{i+1/2} \\ \end{array} \right)
\]

This is an equation in the scattered-field, but \( E_{x,m}^{i+1/2} \) is a total-field quantity.

We must subtract the source from \( E_{x,m}^{i+1/2} \) to make it look like a scattered-field quantity.

\[
C_E \left[ \begin{array}{c} E_x^{i+1/2,m-1} \\ E_y^{i+1/2,m-1} \\ E_z^{i+1/2,m-1} \\ E_{x,m}^{i+1/2} \\ E_{y,m}^{i+1/2} \\ E_{z,m}^{i+1/2} \\ \end{array} \right] = \frac{\Delta y}{\Delta} \left( \begin{array}{c} \frac{E_x^{i+1/2,m-1} - E_x^{i-1/2,m-1}}{\Delta y} \\ \frac{E_y^{i+1/2,m-1} - E_y^{i-1/2,m-1}}{\Delta y} \\ \frac{E_z^{i+1/2,m-1} - E_z^{i-1/2,m-1}}{\Delta y} \\ \frac{1}{\Delta y} E_{x,m}^{i+1/2} \\ \frac{1}{\Delta y} E_{y,m}^{i+1/2} \\ \frac{1}{\Delta y} E_{z,m}^{i+1/2} \\ \end{array} \right)
\]

This is a correction term that can be implemented after calculating the curl to inject a source.
Correction to Finite-Difference Equations at the Problem Cells (2 of 2)

On the total-field side of the TF/SF interface, the finite-difference equation contains a term from the scattered-field side. Due to the staggered nature of the Yee grid, this only occurs in the update equation for an $D$ field. In fact, this only occurs in the computation of the curl of $H$ used in the $D$ field update equations.

$$C_H^{ij} = \frac{H^{ij+1/2} - H^{ij-1/2}}{\Delta x} - \frac{H^{i+1/2,j} - H^{i-1/2,j}}{\Delta y}$$

This is an equation in the scattered-field, but $H^{i,j-1}$ is a total-field quantity.

We must add the source to $\tilde{H}^{i,j-1}$ to make it look like a total-field quantity.

$$C_N^{ij} = \frac{H^{ij+1/2} - H^{ij-1/2}}{\Delta x} - \frac{H^{i+1/2,j} - H^{i-1/2,j}}{\Delta y}$$

We need to make a few observations that must be accounted for before we can calculate these source functions correctly.

1. The amplitude of these functions can be different as $E$ and $H$ are related through the material impedance.
2. These functions are a half grid cell apart and have a small time delay between them.
3. These functions exist at different time steps.

The Two Source Terms

From the previous slides, we now know that we need to calculate two source functions before the main FDTD loop. These are:

$$\tilde{H}_x^{\text{src}} \left|_{t + \frac{\Delta t}{2}}^{i+j_{\text{src}}^{-1}} \right.$$  

$$E_z^{\text{src}} \left|_t^{i+j_{\text{src}}} \right.$$  

We need to make a few observations that must be accounted for before we can calculate these source functions correctly.

1. The amplitude of these functions can be different as $E$ and $H$ are related through the material impedance.
2. These functions are a half grid cell apart and have a small time delay between them.
3. These functions exist at different time steps.
Calculation of the Source Functions \((E_z \text{ Mode})\)

We calculate the electric field as

\[
E_{z, \text{src}}^{\text{src}, \text{ij}}(z, t) = g(t)
\]

We calculate the magnetic field as

\[
H_{x, \text{src}}^{\text{src}, \text{ij}}(t) = \sqrt{\frac{\varepsilon_r}{\mu_r}} g \left( t + \frac{n_{\text{src}} \Delta y}{2c_0} + \frac{\Delta t}{2} \right)
\]

Amplitude due to Maxwell's equations

Delay through one half of a grid cell

Half time step difference

TF/SF Block Diagram for \(E_z \text{ Mode}\)

Main loop...

1. Done? yes → Finished!
   - no
   - Compute Curl of E
     - Inject TF/SF Source into curl of E
       - Update H Integrations
       - Update H Field
     - Compute Curl of H
       - Inject TF/SF Source into curl of H
         - Update D Integrations
         - Update Dz
         - Update Ez
         - Visualize Fields

Lecture 15 Slide 43
Lecture 15 Slide 44
TF/SF Block Diagram for $H_\zeta$ Mode

Main loop...

- **Done?**
  - yes → Finished!
  - no → Main loop...

- **Compute Curl of E** → **Inject TF/SF Source into curl of E** → **Update H Integrations** → **Update H Field**

- **Visualize Fields** → **Compute Curl of H** → **Inject TF/SF Source into curl of H** → **Update D Integrations** → **Update D Field** → **Update E Field**

Lecture 15 Slide 45