EE 5303
Electromagnetic Analysis Using
Finite-Difference Time-Domain

Lecture #22
Waveguide Simulation
Walkthrough

Lecture Outline

• Optical Integrated Circuits
• Walkthrough
  – Step 0 – Problem Definition
  – Step 1 – Define the problem in MATLAB
  – Step 2 – Compute grid
  – Step 3 – Build device on grid
  – Step 4 – Compute source
  – Step 5 – Initialize Fourier transforms
  – Step 6 – Compute the PML
  – Step 7 – Compute update coefficients
  – Step 8 – Initialize FDTD data arrays
  – Step 9 – Main FDTD loop
  – Step 10 – Compute reflectance and transmittance
  – Step 11 – Produce professional looking results
• Results
Optical Integrated Circuits

Integrated Optical Waveguides

- Stripe waveguide
- Diffused waveguide
- Buried-strip waveguide
- Buried-rib waveguide
- Rib waveguide
- Strip-loaded waveguide
Another Fundamental Circuit Element...

To interconnect multiple photonic elements within a single optical integrated circuit, it is necessary to turn waveguides. A critical design parameter is the minimum bend radius that will prevent scattering out of the waveguides.
Typically, we start with a cleaned fused silica substrate. A 4” wafer is common in research labs. Fused silica has $n = 1.52$.

Second, a layer of high index material is deposited onto the silicon wafer. A common process is plasma enhanced chemical vapor deposition (PECVD). A common high index material is silicon nitride (SiN) which has $n = 1.9$. 
Lecture 22 Slide 9

Fabrication of a Rib Waveguide (3 of 6)

Third, a photoresist is spun onto the wafer using a spinner. A common photoresist is PMMA.

Silica substrate with SiN and photoresist

Lecture 22 Slide 10

Fabrication of a Rib Waveguide (4 of 6)

Fourth, the resist is exposed to ultraviolet radiation through a mask in the pattern of the eventual optical integrated circuit. The exposed resist is then washed away leaving behind the unexposed resist.

Silica substrate with SiN and developed photoresist
Fifth, the wafer is etched using a plasma etching process. Both the resist and SiN are etched, but the remaining resist prevent etching of the SiN material directly underneath.

Sixth, the wafer is cleaned by removing the remaining resist. The optical integrated circuit (and our rib waveguide) is complete!
Rib Waveguide Example

\[ \lambda_0 = 1550 \text{ nm} \]
\[ w = 775 \text{ nm} \]
\[ h = 775 \text{ nm} \]
\[ a = 310 \text{ nm} \]
\[ n_{\text{air}} = 1.00 \]
\[ n_{\text{SiN}} = 1.90 \]
\[ n_{\text{SiO}_2} = 1.52 \]

Effective Index Method

Vertically Polarized

\[ n_1 \]

Hz mode

\[ n_{\text{eff},1} \]

Ez mode

\[ n_{\text{eff},2} \]

Requires rigorous full-vector model

\[ n_{\text{eff}} \]

We hope these are very similar results

Requires only a 1D slab model

\[ n_{\text{eff}} \]
It is possible to very accurately simulate an optical integrated circuit in two dimensions using the effective index method.

Effective indices are best computed by modeling the vertical cross section as a slab waveguide.

A simple average index can also produce good results.
Step 0: Problem Definition

An optical integrated circuit is to be composed of stripe waveguides as shown below and operate at a wavelength of 1550 nm. Determine the minimum bend radius that ensures 90% through a 90° bend.

What device are you modeling? -- stripe waveguide with 90° bend
What is its geometry? -- see above
What materials is it made from? -- see above
What do you wish to learn? -- minimum bend radius for >90% transmission

Fused silica (SiO₂) substrate
\( n = 1.52 \)

---

Step 1: Define Problem – setup MATLAB

```matlab
% Lecture22_bend.m
% INITIALIZE MATLAB
close all;
clear all;

% UNITS
meters = 1;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
inches = 2.54 * centimeters;
feet = 12 * inches;
seconds = 1;
hertz = 1/seconds;
kilohertz = 1e3 * hertz;
megahertz = 1e6 * hertz;
gigahertz = 1e9 * hertz;

% CONSTANTS
e0 = 8.85418782e-12;
u0 = 1.25663706e-6;
N0 = sqrt(u0/e0);
c0 = 299792458 * meters/seconds;
```

```matlab
% SOURCE
lam0 = 1.55 * micrometers;
f0 = c0/lam0;
k0 = 2*pi/lam0;

% RIB WAVEGUIDE PARAMETERS
nair = 1.0;
nsub = 1.52;
core = 1.9;
w = 0.5 * lam0;
h = 0.5 * lam0;
rbend = 20 * micrometers;

% GRID PARAMETERS
Nmax = max([nair nsub core]);
NRES = 10;
BUF = 5*lam0 * [1 2 1 1];
NPML = [40 40 40 40];
```

---
Step 1: Define Problem – *reduction to 2D*

**Assumption #1: Effective index method**

Performing a fully three-dimensional simulation using standard FDTD is prohibitively computationally intensive. An alternative is to approximate the 3D circuit as a 2D circuit using the effective index method.

For the stripe waveguide, there is no slab waveguide outside of the core. So instead of modeling a slab waveguide, we just estimate the effective by assuming 30% of the energy will reside in the air.

\[
n_{\text{eff,2}} \approx 0.3n_{\text{air}} + 0.7n_{1}
\]

\[
n_{\text{eff,1}} = 1.7795
\]

\[
n_{\text{eff,2}} = 1.3640
\]

---

Step 1: Define Problem – *calculate effective n’s*

% COMPUTE SLAB GRID

\[
Sz = 7*h; \quad \text {%size of grid}
\]

\[
dz = lam0/NRES/nmax; \quad \text {%initial resolution}
\]

\[
Nz = ceil(h/dz); \quad \text {%snap grid to h}
\]

\[
dz = h/Nz; \quad \text {%revised resolution}
\]

\[
Nz = ceil(Sz/dz); \quad \text {%total number of cells}
\]

% INITIALIZE MATERIALS TO FREE SPACE

\[
ERzz = ones(1,Nz); \quad \text {ERxx = ones(1,Nz);}
\]

\[
URyy = ones(1,Nz); \quad \text {%INITIALIZE MATERIALS TO FREE SPACE}
\]

% COMPUTE POSITION INDICES

\[
nz1 = round(3*h/dz); \quad \text {nz1 = round(3*h/dz);}
\]

\[
nz2 = nz1 + round(h/dz) - 1; \quad \text {nz2 = nz1 + round(h/dz) - 1;}
\]

% CLADDING

\[
neff2 = 0.3*nair + 0.7*nsub; \quad \text {neff2 = 0.3*nair + 0.7*nsub;}
\]

% CORE SLAB

\[
ERzz = nair^2 * ones(1,Nz); \quad \text {ERzz = nair^2 * ones(1,Nz);}
\]

\[
ERzz(nz1:nz2) = ncore^2; \quad \text {ERzz(nz1:nz2) = ncore^2;}
\]

\[
ERzz(nz2+1:Nz) = nsub^2; \quad \text {ERzz(nz2+1:Nz) = nsub^2;}
\]

\[
[Ez,Hx,neff1] = ezmode(URxx,URyy,ERzz,k0*dz); \quad \text {[Ez,Hx,neff1] = ezmode(URxx,URyy,ERzz,k0*dz);}
\]
Step 2: Compute Grid – grid construction

Step 2: Compute Grid – grid resolution

Compute Initial Grid Resolution

\[ N_x' = \frac{w}{\Delta x'} = 9.5 \text{ cells} \rightarrow N_x = 10 \]
\[ \Delta x' = \frac{\lambda_{\min}}{n_{\max} N_x} = 81.6 \text{ nm} \]

Snap Grid to Critical Dimensions

\[ N_y' = \frac{w}{\Delta y'} = 9.5 \text{ cells} \rightarrow N_y = 10 \]
\[ \Delta y' = \frac{\lambda_{\min}}{n_{\max} N_y} = 81.6 \text{ nm} \]
Step 2: Compute Grid – grid size

Determine Physical Size

\[ a = r_{\text{bend}} + w/2 = 5.3875 \mu m \]
\[ S_x = \text{BUF}(1) + a + \text{BUF}(2) = 11.5875 \mu m \]
\[ S_y = \text{BUF}(3) + a + \text{BUF}(4) = 11.5875 \mu m \]

Calculate Number of Cells

\[ N_x = \text{NPML}(1) + \frac{S_x}{\Delta x} + \text{NPML}(2) = 230 \]
\[ S_x = N_x \cdot \Delta x = 17.825 \mu m \]
\[ N_y = \text{NPML}(3) + 3 + \frac{S_y}{\Delta y} + 2 + \text{NPML}(4) = 230 \]
\[ S_y = N_y \cdot \Delta y = 17.825 \mu m \]

% COMPUTE GRID SIZE
\[ a = r_{\text{bend}} + w/2; \]
\[ S_x = \text{BUF}(1) + a + \text{BUF}(2); \]
\[ N_x = \text{ceil}(S_x/dx) + \text{NPML}(1) + \text{NPML}(2); \]
\[ S_x = N_x \cdot dx; \]
\[ S_y = \text{BUF}(3) + a + \text{BUF}(4); \]
\[ N_y = \text{ceil}(S_y/dy) + \text{NPML}(3) + \text{NPML}(4); \]
\[ S_y = N_y \cdot dy; \]

% COMPUTE GRID AXES
\[ x_a = [0:N_x-1] \cdot dx; \]
\[ y_a = [0:N_y-1] \cdot dy; \]

Step 3: Build Device – initialize grid

Initialize Materials to Free Space

\[ \mu_{xx}(x, y) = 1.0 \]
\[ \mu_{yy}(x, y) = 1.0 \]
\[ \varepsilon_{zz}(x, y) = 1.0 \]

% INITIALIZE MATERIALS TO FREE SPACE
\[ \text{URxx} = \text{ones}(N_x,N_y); \]
\[ \text{URyy} = \text{ones}(N_x,N_y); \]
\[ \text{ERzz} = \text{ones}(N_x,N_y); \]

Compute Position Indices

% COMPUTE START AND STOP INDICES OF BEND WINDOW
\[ n_x = \text{round}(a/dx); \]
\[ n_{x1} = \text{NPML}(1) + \text{round}(\text{BUF}(1)/dx); \]
\[ n_{x2} = n_{x1} + n_x - 1; \]
\[ n_y = \text{round}(a/dy); \]
\[ n_{y1} = \text{NPML}(3) + \text{round}(\text{BUF}(3)/dx); \]
\[ n_{y2} = n_{y1} + n_y - 1; \]
Step 3: Build Device – **construct bend**

**Compute the Center of Curvature of the Waveguide Bend**

% COMPUTE CENTER OF CURVATURE
x0 = nx2*dx;
y0 = ny1*dy;

**Construct the Bend**

% CONSTRUCT BEND
[Y,X] = meshgrid(ya,xa);
R1 = (X - x0).^2 + (Y - y0).^2 <= (rbend + w/2)^2;
R2 = (X - x0).^2 + (Y - y0).^2 >= (rbend - w/2)^2;
R = R1 .* R2;

Step 3: Build Device – **add I/O waveguides**

**Clip the Center Window for a Single 90° Bend**

% CLIP BEND WINDOW
R(1:nx1-1,:) = 0;
R(nx2+1:Nx,1:ny1-1) = 0;
R(:,1:ny1-1) = 0;
R(:,ny2+1:Ny) = 0;

**Add Input Waveguide**

% ADD INPUT WAVEGUIDE
nx = round(w/dx);
nxa = nx1;
nxb = nxa + nx - 1;
R(nxa:nxb,1:ny1-1) = 1;

**Add Output Waveguide**

% ADD OUTPUT WAVEGUIDE
ny = round(w/dy);
nya = ny2 - ny + 1;
nyb = ny2;
R(nx2+1:Nx,nya:nyb) = 1;
Step 3: Build Device – *convert to materials*

Convert “Binary” Waveguide Structure to Materials in ERzz

\[
\text{ERzz} = \text{neff1}^2 + (\text{neff1}^2 - \text{neff2}^2) \cdot R;
\]

Step 4: Compute Source – *time step*

**Compute Stable Time Step**

\[
\Delta t = \frac{\Delta \text{min}}{2c_0};
\]

“Snap” Time Step so One Wave Cycle is an Integer Number of Time Steps

\[
\text{period} = \frac{1}{f_0};
\]

% SNAP TIME STEP SO WAVE PERIOD IS AN INTEGER NUMBER OF STEPS

\[
\text{Nt} = \text{ceil}(\text{period}/\text{dt});
\]

\[
\text{dt} = \text{period}/\text{Nt};
\]
Step 4: Compute Source – extract slab waveguide

Determine Position of Source

% COMPUTE SOURCE POSITION
nx1_src = NPML(1) + 1;
nx2_src = Nx - NPML(2);
ny_src = NPML(3) + 2;

Extract the Cross Section of the Waveguide

% EXTRACT MATERIALS ACROSS INPUT SLAB WAVEGUIDE
urxx = URxx(nx1_src:nx2_src,ny_src);
uryy = URyy(nx1_src:nx2_src,ny_src);
erzz = ERzz(nx1_src:nx2_src,ny_src);

Step 4: Compute Source – compute fundamental mode

Analyze the Waveguide

% ANALYZE WAVEGUIDE
[Ez_src,Hx_src,neff,EZR,mref] = ezmode(urxx,uryy,erzz,k0*dx);
emax = max(abs(Ez_src));
Step 4: Compute Source – other source data

Compute Delay Between E and H

\[ \delta t = \frac{n_{\text{eff}} \Delta y + \Delta t}{2c_0} \]

% COMPUTE DELAY
\[ \text{delt} = 0.5 \times n_{\text{eff}} \times \text{dy} \times c_0 + \text{dt} / 2; \]

Compute the Number of FDTD Iterations

\[ d = \sqrt{S_x^2 + S_y^2} \]
\[ \tau_{\text{prop}} = n_{\text{max}} d / c_0 \]
\[ \tau_{\text{sim}} = 2 \tau_{\text{prop}} \]
\[ \text{STEPS} = \text{ceil}(\tau_{\text{sim}} / \Delta t) \]

% COMPUTE NUMBER OF TIME STEPS
\[ \text{d} = \text{sqrt}(S_x^2 + S_y^2); \]
\[ \text{tprop} = \text{nmax} \times d / c_0; \]
\[ \text{tsim} = 2 \times \text{tprop}; \]
\[ \text{STEPS} = \text{ceil}(\text{tsim} / \text{dt}); \]

Compute Ramp Function

\[ \tau = 3 / f_0 \]
\[ t_0 \geq 3 \tau \]
\[ r(t) = \begin{cases} 
\exp\left[ -\left(\frac{t-t_0}{\tau}\right)^2 \right] & t \leq t_0 \\
1 & t > t_0
\end{cases} \]

% COMPUTE RAMP FUNCTION
\[ \text{tau} = 3 / f_0; \]
\[ \text{t0} = 3 \times \text{tau}; \]
\[ \text{ta} = [0 : \text{STEPS}-1] \times \text{dt}; \]
\[ \text{ramp} = \exp(-((\text{ta} - \text{t0}) / \text{tau})^2); \]
\[ \text{ind} = \text{find}(\text{ta} \geq \text{t0}); \]
\[ \text{ramp}(\text{ind}) = 1; \]

Step 5: Initialize Output – Fourier transforms

Compute Kernel

\[ K_0 = \exp(-j2\pi \cdot \Delta t \cdot f_0) \]

% KERNEL FOR f0
\[ \text{K} = \exp(-1i \times 2 \pi \times \text{dt} \times f_0); \]

Initialize Steady-State Field Arrays

% INITIALIZE STEADY-STATE FIELDS
\[ \text{Eref} = \text{zeros}(\text{Nx}, 1); \]
\[ \text{Etrn} = \text{zeros}(1, \text{Ny}); \]

Compute Position of Record Planes

% POSITION OF RECORD PLANES
\[ \text{nyref} = \text{NPML}(3) + 1; \]
\[ \text{nxtrn} = \text{Nx} - \text{NPML}(2); \]
Step 5: Initialize Output – output waveguide

Extract Materials at Transmission Record Plane

```matlab
% EXTRACT MATERIALS
ny1 = NPML(3) + 1;
ny2 = Ny - NPML(4);
nx = Nx - NPML(2);

urxx = URxx(nx,ny1:ny2);
uryy = URyy(nx,ny1:ny2);
erzz = ERzz(nx,ny1:ny2);
```

Analyze Output Waveguide

```matlab
% ANALYZE OUTPUT WAVEGUIDE
[Ez,Hx,n,EZT,mtrn] = ezmode(urxx,uryy,erzz,k0*dy);
```

Step 6: Compute the PML

Compute Size of 2x Grid

```matlab
% NUMBER OF POINTS ON 2X GRID
Nx2 = 2*Nx;
Ny2 = 2*Ny;
```

Compute PML Parameters

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

for nx = 1 : 2*NPMU(2)
    nx1 = Nx2 - 2*NPMU(2) + nx;
    sigx(nx1,:) = (0.5*e0/dt)*(nx/2/NPMU(2))^3;
end

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

for ny = 1 : 2*NPMU(4)
    ny1 = Ny2 - 2*NPMU(4) + ny;
    sigy(:,ny1) = (0.5*e0/dt)*(ny/2/NPMU(4))^3;
end
```
Step 7: Compute Update Coefficients -- $H_x$

**Compute Update Coefficients for $H_x$**

Extract the PML parameters from the 2x grid.

\[
\begin{align*}
\text{sigHx} & = \text{sigx(1:2:Nx2,2:2:Ny2)}; \\
\text{sigHy} & = \text{sigy(1:2:Nx2,2:2:Ny2)}; \\
\text{mHx0} & = (1/dt) + \text{sigHy}/(2*e0); \\
\text{mHx1} & = ((1/dt) - \text{sigHy}/(2*e0))/\text{mHx0}; \\
\text{mHx2} & = -c0/\text{URxx}/\text{mHx0}; \\
\text{mHx3} & = -(c0*dt/e0) * \text{sigHx}/\text{URxx}/\text{mHx0};
\end{align*}
\]

\[
\begin{align*}
\text{mHx4}^f & = \frac{1}{\Delta t} + \left( \frac{\sigma^H}{2e_0} \right) \\
mHx1^f & = \frac{1}{\text{mHx4}^f} - \frac{\sigma^H}{2e_0} \\
mHx2^f & = -\frac{c0}{\text{URxx}/\text{mHx0}} \\
mHx3^f & = -\frac{(c0*dt/e0) * \text{sigHx}/\text{URxx}/\text{mHx0}}{}
\end{align*}
\]

Step 7: Compute Update Coefficients -- $H_y$

**Compute Update Coefficients for $H_y$**

Extract the PML parameters from the 2x grid.

\[
\begin{align*}
\text{sigHx} & = \text{sigx(1:2:Nx2,2:2:Ny2)}; \\
\text{sigHy} & = \text{sigy(1:2:Nx2,2:2:Ny2)}; \\
\text{mHy0} & = (1/dt) + \text{sigHx}/(2*e0); \\
\text{mHy1} & = ((1/dt) - \text{sigHx}/(2*e0))/\text{mHy0}; \\
\text{mHy2} & = -c0/\text{URyy}/\text{mHy0}; \\
\text{mHy3} & = -(c0*dt/e0) * \text{sigHy}/\text{URyy}/\text{mHy0};
\end{align*}
\]

\[
\begin{align*}
\text{mHy4}^f & = \frac{1}{\Delta t} + \left( \frac{\sigma^H}{2e_0} \right) \\
mHy1^f & = \frac{1}{\text{mHy4}^f} - \frac{\sigma^H}{2e_0} \\
mHy2^f & = -\frac{c0}{\text{URyy}/\text{mHy0}} \\
mHy3^f & = -\frac{(c0*dt/e0) * \text{sigHy}/\text{URyy}/\text{mHy0}}{}
\end{align*}
\]
Step 7: Compute Update Coefficients --- $D_z$

**Compute Update Coefficients for $D_z$**

Extract the PML parameters from the 2× grid.

% COMPUTE DZ UPDATE COEFFICIENTS

\[
\text{sigDx} = \text{sigx}(1:2:Nx2, 1:2:Ny2);
\text{sigDy} = \text{sigy}(1:2:Nx2, 1:2:Ny2);
\text{mDz0} = (1/dt) + (\text{sigDx} + \text{sigDy})/(2*e0) + \text{sigDx}.*\text{sigDy}*(dt/4/e0^2);
\text{mDz1} = (1/dt) - (\text{sigDx} + \text{sigDy})/(2*e0) - \text{sigDx}.*\text{sigDy}*(dt/4/e0^2);
\text{mDz1} = \text{mDz1} ./ \text{mDz0};
\text{mDz2} = \text{c0} ./ \text{mDz0};
\text{mDz4} = - (dt/e0^2)*\text{sigDx}.*\text{sigDy}./\text{mDz0};
\]

Compute the update coefficients

\[
m_{\text{Dz0}} = \frac{1}{\Delta t} \frac{\sigma_\theta^{ij} + \sigma_\phi^{ij}}{2\epsilon_0} \left( \sigma_\theta^{ij} \sigma_\phi^{ij} \right) \Delta t
\]

\[
m_{\text{Dz1}} = \frac{1}{\Delta t} \frac{\sigma_\theta^{ij} + \sigma_\phi^{ij}}{2\epsilon_0} \left( \sigma_\theta^{ij} \sigma_\phi^{ij} \right) \Delta t
\]

\[
m_{\text{Dz2}} = \frac{\epsilon_0}{\Delta t} \frac{\sigma_\theta^{ij} \sigma_\phi^{ij}}{\epsilon_0} \Delta t
\]

\[
m_{\text{Dz4}} = - \frac{\epsilon_0}{\Delta t} \frac{\sigma_\theta^{ij} \sigma_\phi^{ij}}{\epsilon_0} \Delta t
\]

Step 7: Compute Update Coefficients --- $E_z$

**Compute Update Coefficients for $E_z$**

% COMPUTE EZ UPDATE COEFFICIENT

\[
m_{\text{Ez1}} = 1./\text{ERzz};
\]
Step 8: Initialize FDTD Data Arrays

Initialize FDTD Data Arrays

```
% INITIALIZE FIELDS
Hx = zeros(Nx,Ny);
Hy = zeros(Nx,Ny);
Dz = zeros(Nx,Ny);
Ez = zeros(Nx,Ny);

% INITIALIZE CURL
CEx = zeros(Nx,Ny);
CEy = zeros(Nx,Ny);
CHz = zeros(Nx,Ny);

% INITIALIZE INTEGRATION TERMS
ICEx = zeros(Nx,Ny);
ICEy = zeros(Nx,Ny);
IDz = zeros(Nx,Ny);
```

Step 9: Run FDTD (Main Loop)

```
Compute Curl of E-Field
Correct Curl of E-Field for TF/SF
Update H-Field Integration Terms
Update H-Field

Compute Curl of H-Field
Correct Curl of H-Field for TF/SF
Update D-Field Integration Terms
Update D-Field

Update E-Field
Update Fourier Transforms
Visualize Simulation

Done?
```

Finished!
Step 9: Main Loop – *Curl of E*

Compute the *x*-Component of the *Curl of E*

\[
C_x^{i,j} = \begin{cases} 
\frac{\tilde{E}_x^{i+1,j} - \tilde{E}_x^{i,j}}{\Delta y} & \text{for } j < N_y, \\
\frac{\tilde{E}_x^{i,j} - \tilde{E}_x^{i,N_y}}{\Delta y} & \text{for } j = N_y 
\end{cases}
\]

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

Compute the *y*-Component of the *Curl of E*

\[
C_y^{i,j} = \begin{cases} 
-\frac{\tilde{E}_y^{i+1,j} - \tilde{E}_y^{i,j}}{\Delta x} & \text{for } i < N_x, \\
\frac{\tilde{E}_y^{i,j} - \tilde{E}_y^{i,N_x}}{\Delta x} & \text{for } i = N_x 
\end{cases}
\]

% Compute CEy
for nx = 1 : Nx-1
    for ny = 1 : Ny
        CEy(nx,ny) = - (Ez(nx+1,ny) - Ez(nx,ny))/dx;
    end
end
for ny = 1 : Ny
    CEy(Nx,ny) = - (Ez(1,ny) - Ez(Nx,ny))/dx;
end

Step 9: Main Loop – Correct *Curl-E* for TF/SF

Correct CEx for TF/SF Framework

\[
C_x^{i,j} = \frac{\frac{\tilde{E}_x^{i+1,j} - \tilde{E}_x^{i,j}}{\Delta y}}{\Delta y} - \frac{1}{\Delta y} E_x^{i,j} 
\]

We already calculated this.

We just need to incorporate this correction term for all values of *i*.

% TF/SF Correction
ezsrc = ramp(T) * real(Ez_src*exp(-1i*2*pi*f0*T*dt));
CEx(nx1_src:nx2_src,ny_src-1) = CEx(nx1_src:nx2_src,ny_src-1) - ezsrc/dy;
Step 9: Main Loop – Update H

Update Integration Terms

% Update Integration Terms
ICEx = ICEx + CEx;
ICEy = ICEy + CEy;

Update Hx and Hy

% Update Hx and Hy
Hx = mHx1.*Hx + mHx2.*CEx + mHx3.*ICEx;
Hy = mHy1.*Hy + mHy2.*CEy + mHy3.*ICEy;

Step 9: Main Loop – Curl of H

Compute the z-Component of the Curl of H

\[
C_{Hz}^{ij} = \begin{cases} 
\frac{H_x^{i+1,j} - H_x^{i-1,j}}{2 \Delta x} - \frac{H_y^{i,j+1} - H_y^{i,j-1}}{2 \Delta y} & \text{for } i > 1 \text{ and } j > 1 \\
\frac{H_x^{i+1,j} - H_x^{i-1,j}}{2 \Delta x} - \frac{H_y^{i,j+1} - H_y^{i,j-1}}{2 \Delta y} & \text{for } i = 1 \text{ and } j > 1 \\
\frac{H_x^{i+1,j} - H_x^{i-1,j}}{2 \Delta x} - \frac{H_y^{i,j+1} - H_y^{i,j-1}}{2 \Delta y} & \text{for } i > 1 \text{ and } j = 1 \\
\frac{H_x^{i+1,j} - H_x^{i-1,j}}{2 \Delta x} - \frac{H_y^{i,j+1} - H_y^{i,j-1}}{2 \Delta y} & \text{for } i = 1 \text{ and } j = 1 
\end{cases}
\]

% Compute CHz
CHz(1,2) = (Hy(1,1) - Hy(1,2))/dx ... - (Hx(1,1) - Hx(1,2))/dy;
for nx = 2 : Nx
    CHz(nx,1) = (Hy(nx,1) - Hy(nx-1,1))/dx ... - (Hx(nx,1) - Hx(nx-1,1))/dy;
    end
for ny = 2 : Ny
    CHz(1,ny) = (Hy(1,ny) - Hy(1,ny-1))/dx ... - (Hx(1,ny) - Hx(1,ny-1))/dy;
    for nx = 2 : Nx
        CHz(nx,ny) = (Hy(nx,ny) - Hy(nx-1,ny))/dx ... - (Hx(nx,ny) - Hx(nx,ny-1))/dy;
        end
    end
Step 9: Main Loop – Correct Curl-H for TF/SF

Correct CHz for TF/SF Framework

\[ \frac{\Delta y}{\Delta x} \left( \frac{H_y^{i+1/2} - H_y^{i-1/2}}{\Delta x} - \frac{H_z^{i+1/2} - H_z^{i-1/2}}{\Delta y} \right) - \frac{1}{\Delta y} \frac{\partial}{\partial y} \left( \frac{1}{\mu_y} \frac{\partial H_z}{\partial y} \right) \]

We already calculated this.

% TF/SF Correction
hxsrc = ramp(T) * real(Hx_src*exp(-1i*2*pi*f0*(T*dt + delt)));
CHz(nx1_src:nx2_src,ny_src) = CHz(nx1_src:nx2_src,ny_src) - hxsrc/dy;

Step 9: Main Loop – Update D and E

Update Integration Term for D-Field Update

% Update Integration Term
IDz = IDz + Dz;

Update Dz

% Update Dz
Dz = mDz1.*Dz + mDz2.*CHz + mDz4.*IDz;

Update Ez

% Update Ez
Ez = mEz1.*Dz;
Step 9: Main Loop – Update Fourier Transforms

Update the Fourier Transforms During Last Wave Cycle

\[
\begin{align*}
E_{ref}^{(n)} & \approx 2\Delta t \cdot f_0 \sum_{y_n} \left( e^{-j2\pi f_0 \Delta y} \right)^n \cdot E_m^{(n)} \\
E_{trn}^{(n)} & \approx 2\Delta t \cdot f_0 \sum_{y_n} \left( e^{-j2\pi f_0 \Delta y} \right)^n \cdot E_m^{(n)}
\end{align*}
\]

These are calculated at each frequency and at each point across the entire grid.

% Update f0 Fourier Transform
if T>(STEPS-Nt)
    Eref = Eref + (K*(T-STEPS+Nt))*Ez(:,nyref);
    Etrn = Etrn + (K*(T-STEPS+Nt))*Ez(nxtrn,:);
end

Note: the constants are incorporated after the main loop is complete.

% FINISH TRANSFORMS
Eref = Eref * (2*dt/period);
Etrn = Etrn * (2*dt/period);

Step 9: Main Loop – Visualize Simulation

Draw the Field Superimposed on the Materials

% Update Graphical Status
if ~mod(T,10)
    % draw field
    draw2d(xa,ya,ERzz,Ez,NPML,0.5); axis equal tight;
    title([num2str(T) ' of ' num2str(STEPS)]);
    % force MATLAB to draw graphics
drawnow;
end

Graphics are very slow!
This if statement only updates the visualization every 10 iterations.
Step 10: Reflectance and Transmittance

Compute Mode Amplitude Coefficients

\[ \tilde{e}_z = V a \quad \Rightarrow \quad a = V^{-1} \tilde{e}_z \]

\[ \begin{align*}
\tilde{e}_z & \quad v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5 \\
1 & \quad + & \quad d_2 & \quad + & \quad d_3 & \quad + & \quad d_4 & \quad + & \quad d_5
\end{align*} \]

\[ \begin{align*}
\text{reflected field} & \quad \text{transmitted field}
\end{align*} \]

Compute Transmission and Reflection

\[ p_{\text{ref}} = \left| \frac{a_{\text{ref}}}{a_{\text{inc}}} \right|^2 \quad p_{\text{im}} = \left| \frac{a_{\text{im}}}{a_{\text{inc}}} \right|^2 \]

\[ \begin{align*}
\text{REF} & = \text{abs}(a_{\text{ref}}(m_{\text{ref}}))^2; \\
\text{TRN} & = \text{abs}(a_{\text{im}}(m_{\text{trn}}))^2;
\end{align*} \]

Step 11: Report the Results

\[ \begin{align*}
\text{disp}([\text{"Reflectance: \ num2str(100*REF,'%4.1f') \"}])); \\
\text{disp}([\text{"Transmittance: \ num2str(100*TRN,'%4.1f') \"}]);
\end{align*} \]
Results

Convergence

We first test for convergence with a small grid to determine an appropriate grid resolution for the remaining simulations.

- NRES = 15: Good choice for doing multiple preliminary simulations.
- NRES = 30: Good choice as a final simulation.
Minimum Bend Radius for 90% Transmission

Given $\lambda/15$ grid resolution, we iterate bend radius to find the value that produces greater than 90% transmission.

- $r = 5 \mu m$, $T = 69.0\%$
- $r = 8 \mu m$, $T = 86.4\%$
- $r = 8.5 \mu m$, $T = 89.9\%$
- $r = 10 \mu m$, $T = 93.0\%$
- $r = 9 \mu m$, $T = 92.8\%$