Lecture Outline

• 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
• What could possibly go wrong?
Suppose we wish to simulate transmission and reflection from the following sawtooth grating.

What device are you modeling? -- a sawtooth grating
What is its geometry? -- see above
What materials is it made from? -- $\varepsilon_r=9.0$
What do you wish to learn? -- diffraction efficiency of spatial harmonics at 10 GHz
Step 0: Define Problem – *assumptions for 2D*

**Assumption #1: Infinite substrate**
Due to the thickness of the substrate compared to the grating, we can reduce the size of the grid in the vertical dimension by assumption an infinite substrate. This is common practice in photonics because the substrates can be millions of times thicker than the grating.

**Assumption #2: Infinitely periodic**
We can dramatically reduces the size of the grid in the horizontal direction by assuming the device is infinitely periodic. This is a good assumption when the device is used away from its edges.

---

Step 0: Define Problem – *validity of periodic BC*

The field in a finite periodic device is very nearly periodic away from the edges. For this device, the infinitely periodic approximation does not predict the field in the outer four (or so) unit cells.
Step 1: Dashboard – setup MATLAB

**Initialize MATLAB**

```matlab
%% Lecture22_sawtooth.m
%% INITIALIZE MATLAB
close all;
clc;
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;
```

**Dashboard**

```matlab
%% SOURCE PARAMETERS
NFREQ = 500;
fmax = 15 * gigahertz;
FREQ = linspace(5,15,NFREQ) * gigahertz;

f0 = 10 * gigahertz;
lam0 = c0/f0;

%% GRATING PARAMETERS
L = 1.5 * centimeters;
d = 1.0 * centimeters;
er1 = 1.0;
er2 = 9.0;

%% GRID PARAMETERS
nmax = sqrt(max([er1 er2]));
NRES = 10;
NPML = [0 0 20 20];
BUF = 0.5*lam0 * [1 1];

Note: nothing is “hard coded” after this!
```

---

Step 2: Compute Grid – grid resolution

**Compute Initial Grid Resolution**

\[
N_x = 10 \\
N_y = \sqrt{\varepsilon_2} = 3.0 \\
\lambda_{\text{min}} = \frac{c_0}{f_{\text{max}}} = 2.0 \text{ cm} \\
\Delta x' = \frac{\lambda_{\text{min}}}{N_{\text{max}}N_x} = 666 \mu m \\
\Delta y' = \frac{\lambda_{\text{min}}}{N_{\text{max}}N_y} = 666 \mu m
\]

**Snap Grid to Critical Dimensions**

\[
N_x' = \frac{A}{\Delta x'} = 22.51 \text{ cells} \quad \text{round up} \quad N_x = 2 \text{ceil} \left( \frac{N_x'}{2} \right) + 1 = 25
\]

\[
\Delta x = \frac{A}{N_x} = 0.6000 \text{ mm}
\]

\[
N_y' = \frac{d}{\Delta y'} = 15.01 \text{ cells} \quad \text{round up} \quad N_y = 16
\]

\[
\Delta y = \frac{d}{N_y} = 0.625 \text{ mm}
\]

Note: nothing is “hard coded” after this!
Step 2: Compute Grid – grid size

Determine Physical Size

\[ S_x = \Lambda = 1.5 \text{ cm} \]
\[ S_y = d + \text{BUF}(1) + \text{BUF}(2) = 4.0 \text{ cm} \]

Calculate Number of Cells

\[ N_x = \frac{S_x}{\Delta x} = 25 \quad \text{We already know this because } \Lambda = 1.5 \text{ cm}. \]
\[ N_y = \text{NPML}(3) + 3 + \frac{S_y}{\Delta y} + 2 + \text{NPML}(4) = 109 \]
\[ S_y = N_y \cdot \Delta y = 6.8125 \text{ cm} \]

\% COMPUTE GRID SIZE

\[ Sx = L_x; \]
\[ Sy = \text{sum(BUF)} + d; \]
\[ Ny = \text{ceil}(Sy/dy) + \text{NPML}(3) + \text{NPML}(4) + 5; \]
\[ Sy = Ny \cdot dy; \]

Step 3: Build Device – initialize grid

Initialize Materials to Free Space

\[ \mu_{xx}(x, y) = 1.0 \quad \% \text{INITIALIZE MATERIALS TO FREE SPACE} \]
\[ \mu_{yy}(x, y) = 1.0 \]
\[ \varepsilon_{zz}(x, y) = \varepsilon_{r1} \]

\% INITIALIZE MATERIALS TO FREE SPACE

\[ \text{URxx} = \text{ones}(Nx,Ny); \]
\[ \text{URyy} = \text{ones}(Nx,Ny); \]
\[ \text{ERzz} = \varepsilon_{r1} \times \text{ones}(Nx,Ny); \]

Compute Position Indices

\[ n_{x1} = \text{NPML}(3) + 3 + \text{round} \left( \frac{\text{BUF}(1)}{\Delta y} \right) \]
\[ n_{x2} = n_{x1} + \text{round} \left( \frac{d}{\Delta y} \right) - 1 \]

\% COMPUTE POSITION INDICES

\[ \text{ny1} = \text{NPML}(3) + 3 + \text{round}(\text{BUF}(1)/dy); \]
\[ \text{ny2} = \text{ny1} + \text{round}(d/dy) - 1; \]
Step 3: Build Device – add materials

Add Sawtooth

% ADD GRATING
for ny = ny1 : ny2
    f   = (ny - ny1 + 1)/(ny2 - ny1 + 2);
    nx = round(f*Nx);
    nx2 = Nx;
    nx1 = nx2 - nx + 1;
    ERzz(nx1:nx2,ny) = er2;
end

Add Infinite Substrate

% ADD INFINITE SUBSTRATE
ERzz(:,ny2+1:Ny) = er2;

Step 4: Compute Source – Gaussian parameters

Compute Stable Time Step

\[
\Delta t = \Delta_{\text{min}} \left/ \left(2c_0 \right) \right.
\]

% COMPUTE STABLE TIME STEP
\[
d_{\text{min}} = \min(\{dx, dy\});
\]
\[
dt = d_{\text{min}} / (2c_0);
\]

Compute Source Position

\[
n_y,_{\text{src}} = \text{NPML}(3) + 2
\]

% SOURCE POSITION
\[
n_{y,\text{src}} = \text{NPML}(3) + 2;
\]

Compute Source Parameters

\[
\tau = \frac{0.5}{f_{\text{max}}}, \quad \delta t = \frac{n_{y,\text{src}} \Delta y}{2c_0} + \frac{\Delta t}{2}
\]

% COMPUTE TIME PARAMETERS
\[
t_0 = 0.5/f_{\text{max}};
\]
\[
t_0 = 6*\tau;
\]
\[
A = \sqrt{\varepsilon_{r,\text{src}} / \mu_{r,\text{src}}}
\]
\[
delt = 0.5*dy/c_0 + dt/2;
\]
Step 4: Compute Source – source functions

**Compute Number of Iterations**

\[ \tau_{\text{prop}} = \frac{n_{\max} S_y}{c_0} \]

\[ \tau_{\text{sim}} \approx 2t_0 + 10\tau_{\text{prop}} \]

\[ \text{STEPS} = \left\lceil \frac{\tau_{\text{sim}}}{\Delta t} \right\rceil \]

**Compute Source Functions**

\[ \tilde{E}_{z,\text{src}} = \exp\left[-\frac{(t-t_0)^2}{\tau}\right] \]

\[ H_{x,\text{src}} = A\exp\left[-\frac{(t-t_0+\delta t)^2}{\tau}\right] \]

% TIME STEPS

\[ \text{proptime} = n_{\max}\cdot S_y/c_0; \]

\[ \text{simtime} = 2t_0 + 10\cdot \text{proptime}; \]

\[ \text{STEPS} = \left\lceil \frac{\text{simtime}}{\Delta t} \right\rceil; \]

% COMPUTE GAUSSIAN SOURCES

\[ t_a = \left[0: \text{STEPS}-1\right]\cdot \Delta t; \]

\[ \tilde{E}_z_{\text{src}} = \exp\left(-\frac{(t_a-t_0)^2}{\tau}\right); \]

\[ H_{x,\text{src}} = A\exp\left(-\frac{(t_a-t_0+\delta t)^2}{\tau}\right); \]

Step 5: Initialize Fourier Transforms

**Compute Kernels**

\[ K(f) = \exp\left(-j2\pi \cdot \Delta t \cdot f\right) \]

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

% COMPUTE FOURIER TRANSFORM KERNELS

\[ K = \exp\left(-1i\cdot2\pi\cdot \Delta t \cdot \text{FREQ}\right); \]

\[ K_0 = \exp\left(-1i\cdot2\pi\cdot \Delta t \cdot 0\right); \]

**Initialize Steady-State Fields**

% INITIALIZE STEADY-STATE FIELDS

\[ E_{\text{ref}} = \text{zeros}(\text{Nx}, \text{NFREQ}); \]

\[ E_{\text{trn}} = \text{zeros}(\text{Nx}, \text{NFREQ}); \]

\[ \text{SRC} = \text{zeros}(1, \text{NFREQ}); \]

\[ E_{\text{ref}}_0 = \text{zeros}(\text{Nx}, 1); \]

\[ E_{\text{trn}}_0 = \text{zeros}(\text{Nx}, 1); \]

\[ \text{ssSRC} = 0; \]

**Define Position of Record Planes**

% POSITION OF RECORD PLANES

\[ n_{\text{ref}} = \text{NPML}(3) + 1; \]

\[ n_{\text{trn}} = \text{By} - \text{NPML}(4); \]

**Compute Refractive Indices in Record Planes**

% COMPUTE REFRACTIVE INDICES IN RECORD PLANES

\[ n_{\text{ref}} = \sqrt{\text{ERzz}(1, n_{\text{ref}})\cdot \text{URxx}(1, n_{\text{ref}})}; \]

\[ n_{\text{trn}} = \sqrt{\text{ERzz}(1, n_{\text{trn}})\cdot \text{URxx}(1, n_{\text{trn}})}; \]
**Step 6: Compute the PML**

**Compute Size of 2x Grid**

\[
\begin{align*}
N_{x2} &= 2 \times N_x; \\
N_{y2} &= 2 \times N_y;
\end{align*}
\]

**Compute \( \sigma_y \)**

There is no PML at the x-axis boundary for this problem.

\[
\sigma_y'(y) = \frac{\varepsilon_0}{2\Delta} \left( \frac{y}{L_y} \right)^3
\]

**Compute \( \sigma_x \)**

**Step 7: Compute Update Coefficients -- \( H_x \)**

**Compute Update Coefficients for Hx**

Extract the PML parameters from the 2x grid.

\[
\begin{align*}
\text{sigHx} &= \text{zeros}(1:2:Nx2,2:2:Ny2); \\
\text{sigHy} &= \text{zeros}(1:2:Nx2,2:2:Ny2); \\
\text{mHx0} &= (1/dt) + \text{sigHy}/(2*e0); \\
\text{mHx1} &= ((1/dt) - \text{sigHy}/(2*e0))./\text{mHx0}; \\
\text{mHx2} &= -c0./URxx./\text{mHx0}; \\
\text{mHx3} &= -(c0*dT/e0) * \text{sigHx}/URxx ./ \text{mHx0};
\end{align*}
\]
Step 7: Compute Update Coefficients -- $H_y$

**Compute Update Coefficients for $H_y$**

Extract the PML parameters from the 2x grid.

```matlab
% COMPUTE HY UPDATE COEFFICIENTS
sigHx = sigx(2:Nx2,1:Ny2);
sigy = sigy(2:Nx2,1:Ny2);
mHy0 = (1/dt) + sigHx/(2*e0);
mHy1 = ((1/dt) - sigHx/(2*e0))./mHy0;
mHy2 = c0/URyy./mHy0;
mHy3 = -(c0*dt/e0) * sigHy./URyy ./ mHy0;
```

Compute the update coefficients

\[
m_{Hy0} = \left(1 \frac{\sigma_y^2}{2e_0} \right)
m_{Hy1} = \left(1 \frac{\sigma_x^2}{2e_0} \right)
m_{Hy2} = \frac{c_0}{\mu_0 t}
m_{Hy3} = \frac{c_0 \Delta}{e_0} \sigma_y^2 \sigma_x^2
\]

Step 7: Compute Update Coefficients -- $D_z$

**Compute Update Coefficients for $D_z$**

Extract the PML parameters from the 2x grid.

```matlab
% COMPUTE DZ UPDATE COEFFICIENTS
sigDx = sigx(1:Nx2,1:Ny2);
sigDy = sigy(1:Nx2,1:Ny2);
mbz0 = (1/dt) + (sigDx + sigDy)/(2*e0) ... + sigDx.*sigDy/(dt/4/e0^2);
mbz1 = (1/dt) - (sigDx + sigDy)/(2*e0) ... - sigDx.*sigDy/(dt/4/e0^2);
mbz2 = c0/mbz0;
mbz4 = -(dt/e0^2)*sigDx.*sigDy./mbz2;
```

Compute the update coefficients

\[
m_{Dz0} = \left(1 \frac{\sigma_y^2}{2e_0} \right) + \frac{\sigma_x^2}{4e_0^2} \Delta t
m_{Dz1} = \left(1 \frac{\sigma_y^2}{2e_0} \right) + \frac{\sigma_x^2}{4e_0^2} \Delta t
m_{Dz2} = \frac{c_0}{\mu_0 t}
m_{Dz3} = \frac{c_0 \Delta}{e_0} \sigma_y^2 \sigma_x^2
\]
Step 7: Compute Update Coefficients -- $E_z$

Compute Update Coefficients for $E_z$

$$m_{Ez}^{i,j} = \frac{1}{\varepsilon_{z}^{i,j}}$$

% COMPUTE EZ UPDATE COEFFICIENT
mEz1 = 1./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

Step 9: Main Loop – Curl of E

Compute the x-Component of the Curl of E

\[
C_{E}^{x} = \begin{cases} 
\frac{E_{j+1}^{z} - E_{j-1}^{z}}{2 \Delta y} & \text{for } j < N_{y} \\
\frac{E_{j}^{z} - E_{j}^{z}}{2 \Delta y} & \text{for } j = N_{y} 
\end{cases}
\]

% 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

\[
C_{E}^{y} = \begin{cases} 
\frac{E_{i+1}^{x} - E_{i-1}^{x}}{2 \Delta x} & \text{for } i < N_{x} \\
\frac{E_{i}^{x} - E_{i}^{x}}{2 \Delta x} & \text{for } i = N_{x} 
\end{cases}
\]

% 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(nx,1) - Ez(nx,Ny))/dx;
end
Step 9: Main Loop – Correct Curl-E for TF/SF

Correct CEx for TF/SF Framework

\[ C_{E_x}^{\text{cor}} = \frac{\tilde{E}_y^{\text{src}} - \tilde{E}_x^{\text{src}}}{\Delta y} - \frac{1}{\Delta y} \left[ \tilde{E}_y^{\text{src}} \right]^{\text{src}} \]

We already calculated this.

We just need to incorporate this correction term for all values of \( i \).

% TF/SF
\[ C_{E_x}(i, ny_{src}-1) = C_{E_x}(i, ny_{src}-1) - \frac{E_z^{\text{src}}(i)}{\Delta y} \]

---

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

\[
\frac{\Delta y}{\Delta x} \left( \frac{H_{x(i,j)}^{(i,j)}}{\Delta x} - \frac{H_{x(i+1,j)}^{(i,j)}}{\Delta y} \right) \\
\frac{\Delta y}{\Delta x} \left( \frac{H_{x(i,j)}^{(i,j)}}{\Delta x} - \frac{H_{x(i+1,j)}^{(i,j)}}{\Delta y} \right) \\
\frac{\Delta y}{\Delta x} \left( \frac{H_{x(i,j)}^{(i,j)}}{\Delta x} - \frac{H_{x(i+1,j)}^{(i,j)}}{\Delta y} \right)
\]

for \( i > 1 \) and \( j > 1 \)

for \( i = 1 \) and \( j > 1 \)

for \( i > 1 \) and \( j = 1 \)

for \( i = 1 \) and \( j = 1 \)

% Compute CHz

\[
CHz(1,1) = \frac{(Hy(1,1) - Hy(Nx,1))}{dx} \\
\cdot \frac{(Hx(1,1) - Hx(1,Ny))}{dy};
\]

for \( nx = 2 : Nx \)

\[
CHz(nx,1) = \frac{(Hy(nx,1) - Hy(nx-1,1))}{dx} \\
\cdot \frac{(Hx(nx,1) - Hx(nx,Ny))}{dy};
\]

end

for \( ny = 2 : Ny \)

\[
CHz(1,ny) = \frac{(Hy(1,ny) - Hy(1,ny-1))}{dx} \\
\cdot \frac{(Hx(1,ny) - Hx(1,ny))}{dy};
\]

end

We already calculated this.

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

Correct CHz for TF/SF Framework

\[
CHz(i,ny_{src}) = CHz(i,ny_{src}) + \frac{Hx_{src}(i,ny_{src})}{\frac{dy}{dy}};
\]

We just need to incorporate this correction term for all values of \( i \).
**Step 9: Main Loop – Update D and E**

**Update Integration Term for D-Field Update**

\[
\text{% Update Integration Term} \\
\text{IDz} = \text{IDz} + \text{Dz};
\]

**Update Dz**

\[
\text{% Update Dz} \\
\text{Dz} = m\text{Dz1} \cdot \text{Dz} + m\text{Dz2} \cdot \text{CHz} + m\text{Dz4} \cdot \text{IDz};
\]

**Update Ez**

\[
\text{% Update Ez} \\
\text{Ez} = m\text{Ez1} \cdot \text{Dz};
\]

**Step 9: Main Loop – Update Fourier Transforms**

**Update the Fourier Transforms for the Frequency Sweep**

\[
E_{\text{ref}} (f) \bigg|_{m,\text{ref}} = \Delta f \sum_{n=1}^{M} (e^{(-2\pi j/f_n)})^m \cdot E_{m,\text{ref}} \\
E_{\text{trn}} (f) \bigg|_{m,\text{trn}} = \Delta f \sum_{n=1}^{M} (e^{(-2\pi j/f_n)})^m \cdot E_{m,\text{trn}} \\
E_{\text{src}} (f) = \Delta f \sum_{n=1}^{M} (e^{(-2\pi j/f_n)})^m \cdot E_{\text{src}}
\]

These are recorded at each frequency and at each point across the entire grid in the x direction.

Only a single value is stored for each frequency.

**% Update Fourier Transforms**

\[
\text{for nfreq = 1 : NFREQ} \\
\text{Eref(1,nfreq) = Eref(1,nfreq) + (K(nfreq)^T) * Ez(1,ny_ref) * dt;} \\
\text{Etrn(1,nfreq) = Etrn(1,nfreq) + (K(nfreq)^T) * Ez(1,ny_trn) * dt;} \\
\text{SRC(nfreq) = SRC(nfreq) + (K(nfreq)^T) * Ez_src(T) * dt;} \\
\text{end}
\]

**% Update f0 Fourier Transform**

\[
\text{Eref0 = Eref0 + (K0^T) * Ez(1,ny_ref) * dt;} \\
\text{Etrn0 = Etrn0 + (K0^T) * Ez(1,ny_trn) * dt;} \\
\text{ssSRC = ssSRC + (K0^T) * Ez_src(T) * dt;}
\]
Step 9: Main Loop – Visualize Simulation

Draw the Field Superimposed on the Materials

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

Graphics are very slow! This `if` statement updates the visualization only every 10 iterations.

Step 10: R & T at f₀ – Wave vector components

**Compute Free Space Wave Number**

\[ \lambda_0 = c_0 / f_0 \quad k_0 = 2\pi / \lambda_0 \]

**Compute Incident Wave Vector**

\[ k_{x,inc} = 0 \]

\[ k_{y,inc} = k_0 n_{ref} \]

Assuming normal incidence.

**Compute Transverse Wave Vector Expansion**

\[ k_x(m) = k_{x,inc} - \frac{2\pi m}{\Lambda_x} \quad m = \text{floor} \left( \frac{1}{2} N_x \right), \ldots, \text{floor} \left( \frac{1}{2} N_x \right) \]

**Compute Longitudinal Components of the Wave Vectors**

\[ k_{y,ref}(m) = \sqrt{(k_y n_{ref})^2 - k_x^2(m)} \quad k_{y,ma}(m) = \sqrt{(k_y n_{ma})^2 - k_x^2(m)} \]

\[ \Lambda_x \] and \( S_x \) are the same parameter.
Step 10: R & T at $f_0$ – Reflectance

Normalize the Field Amplitude to the Source

$$E_{\text{ref}}(x, f_0) \approx E_{\text{ref}}(x, f_0) + E_{\text{src}}(f_0)$$

This makes it look like the source at $f_0$ had an amplitude of 1.

Calculate the Amplitudes of the Spatial Harmonics

$$\text{FFT} \left[ E_{\text{ref}}(x) \right] = [S_{-N} \cdots S_2 S_1 S_0 S_1 S_2 \cdots S_N]$$

Calculate the Diffraction Efficiencies of the Spatial Harmonics

$$\text{DE}_\text{ref}(m) = \left| S_{m} \right|^2 \frac{\text{Re} \left[ k_{\text{ref}}(m) \right]}{|k_{\text{inc}}|}$$

Calculate the Overall Reflectance

$$R(f_0) = \sum_{N/2} \text{DE}_\text{ref}(m)$$

Step 10: R & T at $f_0$ – Transmittance

Normalize the Field Amplitude to the Source

$$E_{\text{trn}}(x, f_0) \approx E_{\text{trn}}(x, f_0) + E_{\text{src}}(f_0)$$

This makes it look like the source at $f_0$ had an amplitude of 1.

Calculate the Amplitudes of the Spatial Harmonics

$$\text{FFT} \left[ E_{\text{trn}}(x) \right] = [S_{-N} \cdots S_2 S_1 S_0 S_1 S_2 \cdots S_N]$$

Calculate the Diffraction Efficiencies of the Spatial Harmonics

$$\text{DE}_\text{trn}(m) = \left| S_{m} \right|^2 \frac{\text{Re} \left[ k_{\text{trn}}(m) \mu_{\text{ref}} \right]}{|k_{\text{inc}}| \mu_{\text{trn}}}$$

Calculate the Overall Transmittance

$$T(f_0) = \sum_{N/2} \text{DE}_\text{trn}(m)$$
Step 10: R & T for Frequency Sweep

Initialize the Reflectance and Transmittance Data Arrays

% INITIALIZE REFLECTANCE AND TRANSMITTANCE
REF = zeros(1,NFREQ); TRN = zeros(1,NFREQ);

Note: we only need one value per frequency.

Condense the Last Few Slides Inside a Loop Over Frequency

% LOOP OVER FREQUENCY
for nfreq = 1 : NFREQ
  % Compute Wave Vector Components
  lam0 = c0/FREQ(nfreq); %free space wavelength
  k0 = 2*pi/lam0; %free space wave number
  kyinc = k0*nref; %incident wave vector
  m = [-floor(Nx/2):floor(Nx/2)']'; %spatial harmonic orders
  kx =  - 2*pi*m/Sx; %wave vector expansion
  kyR = sqrt((k0*nref)^2 - kx.^2); %ky in reflection region
  kyT = sqrt((k0*ntrn)^2 - kx.^2); %ky in transmission region

  % Compute Reflectance
  ref = Eref(:,nfreq)/SRC(nfreq); %normalize to source
  ref = fftshift(fft(ref))/Nx; %compute spatial harmonics
  ref = real(kyR/kyinc) .* abs(ref).^2; %compute diffraction eff.
  REF(nfreq) = sum(ref); %compute reflectance

  % Compute Transmittance
  trn = Etrn(:,nfreq)/SRC(nfreq); %normalize to source
  trn = fftshift(fft(trn))/Nx; %compute spatial harmonics
  trn = real(kyT/kyinc) .* abs(trn).^2; %compute diffraction eff.
  TRN(nfreq) = sum(trn); %compute transmittance
end

Step 10: R & T at f₀ – Energy conservation

Compute the Conservation of Energy

\[ R(f_0) + T(f_0) = ? \]

If your simulation does not contain loss or gain, this sum should be very close to 100%.

% COMPUTE ENERGY CONSERVATION
CON0 = REF0 + TRN0;
CON = REF + TRN;

% REPORT RESULTS AT DESIGN FREQUENCY
disp(['Reflectance = ' num2str(100*REF0,'%4.1f') ' %']);
disp(['Transmittance = ' num2str(100*TRN0,'%4.1f') ' %']);
disp(['Conservation = ' num2str(100*CON0,'%4.1f') ' %']);
Step 11: Produce Professional Looking Results

% INITIALIZE FIGURE WINDOW
close all;
fig = figure('Color','w');

% PLOT LINEAR REFLECTANCE, TRANSMITTANCE AND ENERGY CONSERVATION
h = plot(FREQ/gigahertz,100*REF,'-r','LineWidth',2);
hold on;
plot(FREQ/gigahertz,100*TRN,'-b','LineWidth',2);
plot(FREQ/gigahertz,100*CON,:k','LineWidth',2);
hold off;
axis([FREQ(1)/gigahertz FREQ(NFREQ)/gigahertz 0 105]);
h2 = get(h,'Parent');
set(h2,'FontSize',14,'LineWidth',2);
h = legend('Reflectance','Transmittance','Conservation');
set(h,'Location','NorthEastOutside');
xlabel('Frequency (GHz)');
ylabel('%','Rotation',0,'HorizontalAlignment','right');

Results
Diffraction Efficiencies at 10 GHz

\[ \Delta \approx \frac{\lambda}{40} \]

Conservation = 100.5%

\[ \varepsilon_{r1} = 1.0 \]

\[ \varepsilon_{r2} = 9.0 \]

Convergence (1 of 2)

\[ \Delta = \frac{\lambda}{2} \]
\[ R(10 \text{ GHz}) = 10.6\% \]
\[ T(10 \text{ GHz}) = 114.3\% \]
\[ C(10 \text{ GHz}) = 124.8\% \]

\[ \Delta = \frac{\lambda}{4} \]
\[ R(10 \text{ GHz}) = 12.8\% \]
\[ T(10 \text{ GHz}) = 92.4\% \]
\[ C(10 \text{ GHz}) = 165.1\% \]

\[ \Delta = \frac{\lambda}{6} \]
\[ R(10 \text{ GHz}) = 12.0\% \]
\[ T(10 \text{ GHz}) = 90.2\% \]
\[ C(10 \text{ GHz}) = 102.3\% \]

\[ \Delta = \frac{\lambda}{10} \]
\[ R(10 \text{ GHz}) = 11.0\% \]
\[ T(10 \text{ GHz}) = 99.3\% \]
\[ C(10 \text{ GHz}) = 100.8\% \]

\[ \Delta = \frac{\lambda}{20} \]
\[ R(10 \text{ GHz}) = 11.5\% \]
\[ T(10 \text{ GHz}) = 88.7\% \]
\[ C(10 \text{ GHz}) = 100.3\% \]
**Convergence (2 of 2)**

The best way to assess convergence is to plot the desired output parameter as a function of grid resolution for this model.

We could conclude that $\lambda/10$ is an optimum compromise between simulation time and "accuracy."

**Effect of Short PML**

Short PMLs cause non-physical reflections from the boundaries. These reflections cause standing waves that produce "rolling" in the spectral response.

**Graphical Data**

- For $\lambda = 10$ GHz:
  - Reflectance: $11.2\%$
  - Transmittance: $90.3\%$
  - Conservation: $101.6\%$

- For $\lambda = 10$ GHz:
  - Reflectance: $11.5\%$
  - Transmittance: $88.7\%$
  - Conservation: $100.3\%$
What Could Possibly Go Wrong?

Good Simulation of a GMR Filter
Aside from potentially garbage results, poor grid resolution produces large numerical dispersion that tends to shift spectra to lower frequencies, or longer wavelengths.

Reflections from boundaries produces a rolling frequency response that also violates conservation.
Small spacer regions provide an escape path for power by allowing evanescent fields to couple to the PML's. In this simulation, the largest evanescent field occurs due to the guided mode on resonance.

It is sometimes difficult to make conclusions when multiple things are wrong. It is surprising that this simulation is as good as it is.