Lecture 6 Slide 2

• Review of Lecture 5
• Sequence of Code Development
• FDTD Implementation
  – Numerical boundary conditions
  – Grid resolution
  – Courant stability condition
  – Perfect 1D boundary condition
  – Sources
  – Total number of iterations
  – Revised FDTD Algorithm
Review of Lecture #5

Representing Functions on a Grid

<table>
<thead>
<tr>
<th>Example physical (continuous) 2D function</th>
<th>A grid is constructed by dividing space into discrete cells</th>
<th>Function is known only at discrete points</th>
<th>Representation of what is actually stored in memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Example Image" /></td>
<td><img src="image2.png" alt="Grid Image" /></td>
<td><img src="image3.png" alt="Function Image" /></td>
<td><img src="image4.png" alt="Memory Representation" /></td>
</tr>
</tbody>
</table>

Lecture 6

Slide 4
3D Grids

A three-dimensional grid looks like this:

A unit cell from the grid looks like this:

\[ N_x = 10, \quad N_y = 10, \quad N_z = 15 \]

Yee Cell for 1D, 2D, and 3D Grids

1D Yee Grid

2D Yee Grids

3D Yee Grid

Benefits
• Implicitly satisfies divergence equations
• Naturally handles physical boundary conditions
• Elegant approximation of the curl equations using finite-differences

Consequences
• Field components are in physically different locations
• Field components may reside in different materials even if they are in the same unit cell
• Field components will be out of phase
We normalized the magnetic field,

\[ \tilde{H} = \sqrt{\frac{\mu_0}{\varepsilon_0}} H \quad \Rightarrow \quad \nabla \times \tilde{E} = -\frac{\mu_0}{c_0} \frac{\partial \tilde{H}}{\partial t} \quad \nabla \times \tilde{H} = \frac{\varepsilon_0}{c_0} \frac{\partial \tilde{E}}{\partial t} \]

Assume linear, isotropic, and non-dispersive materials and expand the curl equations.

\[
\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial z} = \frac{\mu_{xx}}{c_0} \frac{\partial \tilde{H}_z}{\partial t} \quad \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial x} = \frac{\mu_{yy}}{c_0} \frac{\partial \tilde{H}_x}{\partial t} \quad \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial y} = \frac{\mu_{zz}}{c_0} \frac{\partial \tilde{H}_y}{\partial t}
\]

\[
\frac{\partial \tilde{H}_x}{\partial y} - \frac{\partial \tilde{H}_y}{\partial z} = \frac{\varepsilon_{xx}}{c_0} \frac{\partial E_z}{\partial t} \quad \frac{\partial \tilde{H}_y}{\partial z} - \frac{\partial \tilde{H}_z}{\partial x} = \frac{\varepsilon_{yy}}{c_0} \frac{\partial E_x}{\partial t} \quad \frac{\partial \tilde{H}_z}{\partial x} - \frac{\partial \tilde{H}_x}{\partial y} = \frac{\varepsilon_{zz}}{c_0} \frac{\partial E_y}{\partial t}
\]
We assumed the problem was uniform in the $x$ and $y$ directions.

Maxwell's equations separated into two sets of equations.

\[
\frac{\partial E_y}{\partial t} - \frac{\partial H_z}{\partial x} = \frac{\varepsilon_0}{c_0} \frac{\partial H_y}{\partial x} - \frac{\partial E_z}{\partial x} = \frac{\mu_0}{c_0} \frac{\partial E_x}{\partial z} - \frac{\partial H_y}{\partial z}
\]

We derived update equations by solving the finite-difference equations for the future time value.

\[
\begin{align*}
\tilde{H}_z^{t+1} & = \tilde{H}_z^t + m_{Bz} \frac{E_y^{t+1} - E_y^t}{\Delta z} \\
E_y^{t+1} & = E_y^t + m_{Ey} \frac{\tilde{H}_z^{t+1} - \tilde{H}_z^t}{\Delta z}
\end{align*}
\]

We arrived at the following FDTD algorithm.

\[
\begin{align*}
m^t_{Bz} & = \frac{\varepsilon_0 \Delta t}{\varepsilon_{xy}} \\
m^t_{Ey} & = \frac{\mu_0 \Delta t}{\mu_{yz}}
\end{align*}
\]
Sequence of Code Development

Step 1 – Basic FDTD Algorithm

- Basic update equations
Step 2 – Add Simple Soft Source

- Basic update equations
- Add a soft source

Step 3 – Add Absorbing Boundary

- Basic update equations
- Add a soft source
- Add perfect boundary condition
Step 4 – Add TF/SF

- Basic update equations
- Add a soft source
- Add perfect boundary condition
- Incorporate TF/SF “one-way” source

Step 5 – Move Source & Add T/R

- Basic update equations
- Add a soft source
- Add perfect boundary condition
- Incorporate TF/SF “one-way” source
- Move position of source
- Calculate transmittance and reflectance
Step 6 – Add Device (Complete Algorithm)

- Basic update equations
- Add a soft source
- Add perfect boundary condition
- Incorporate TF/SF “one-way” source
- Move position of source
- Calculate transmittance and reflectance
- Add a real device

Summary of Code Development Sequence

Step 1 – Implement basic FDTD algorithm

Step 2 – Add the source

Step 3 – Add absorbing boundary

Step 4 – Add “one-way” source

Step 5 – Calculate transmittance and reflectance

Step 6 – Add a device
Numerical Boundary Conditions

A Problem at the Boundary of the Grid

We must implement the update equations for every point in the grid.

\[
\frac{\tilde{H}_x^k}{|_{r+\frac{\Delta}{2}}} = \frac{\tilde{H}_x^k}{|_{r-\frac{\Delta}{2}}} + m_{Hx}^k \left( \frac{E_y^{k+1}}{\Delta z} - \frac{E_y^k}{\Delta z} \right)
\]

What do we do at \( k = N_z \)?
\( E_y^{N_z+1} \) does not exist.

\[
\frac{E_y^k}{|_{t+\Delta t}} = \frac{E_y^k}{|_t} + m_{Ey}^k \left( \frac{\tilde{H}_x^k}{|_{r+\frac{\Delta}{2}}} - \frac{\tilde{H}_x^{k-1}}{|_{r+\frac{\Delta}{2}}} \right)
\]

What do we do at \( k = 1 \)?
\( \tilde{H}_x^0 \) does not exist.
Dirichlet Boundary Condition

One easy thing to do is assume the fields outside the grid are zero. This is called a Dirichlet boundary condition.

We modify the update equations as follows.

\[
\hat{E}_y^{t+1} = E_y^t + m_{E_y} \left( \frac{\hat{H}_y^{t+1} - 0}{\Delta z} \right) \quad k = 1
\]

\[
\hat{H}_x^{t+1} = H_x^t + m_{H_x} \left( \frac{\hat{E}_y^{t+1} - \hat{E}_y^{t}}{\Delta z} \right) \quad k < N_z
\]

\[
\hat{H}_x^{N_z} = H_x^{N_z} + m_{H_x} \left( \frac{0 - E_y^{N_z}}{\Delta z} \right) \quad k = N_z
\]

\[
\hat{E}_y^{N_z} = E_y^{N_z} + m_{E_y} \left( \frac{\hat{H}_x^{N_z} - \hat{H}_x^{N_z-1}}{\Delta z} \right) \quad k > 1
\]

Equations → MATLAB Code

% MAIN FDTD LOOP
% for T = 1 : STEPS
% \% Update H from E (Dirichlet Boundary Conditions)
% for nz = 1 : Nz-1
%     Hx(nz) = Hx(nz) + mHx(nz)*(Ey(nz+1) - Ey(nz))/dz;
% end
% Hx(Nz) = Hx(Nz) + mHx(Nz)*(0 - Ey(Nz))/dz;
% \% Update E from H (Dirichlet Boundary Conditions)
% Ey(1) = Ey(1) + mEy(1)*(Hx(1) - 0)/dz;
% for nz = 2 : Nz
%     Ey(nz) = Ey(nz) + mEy(nz)*(Hx(nz) - Hx(nz-1))/dz;
% end

DO NOT use ‘if’ statements to implement boundary conditions.
Periodic Boundary Condition

Some devices are periodic along a particular direction. When this is the case, the field is also periodic.

We modify the update equations as follows.

\[
\begin{align*}
\vec{E}_y^{n+1} |_{z^+} &= \vec{E}_y^n |_{z^+} + \nu \left( \frac{E_y^{n+1} |_{z^+} - E_y^n |_{z^+}}{\Delta z} \right) \\
\vec{H}_z^{n+1} |_{z^+} &= \vec{H}_z^n |_{z^+} + \mu \left( \frac{H_z^{n+1} |_{z^+} - H_z^n |_{z^+}}{\Delta z} \right)
\end{align*}
\]

\[
\begin{align*}
\vec{E}_y^{n+1} |_{z^-} &= \vec{E}_y^n |_{z^-} + \nu \left( \frac{E_y^{n+1} |_{z^-} - E_y^n |_{z^-}}{\Delta z} \right) \\
\vec{H}_z^{n+1} |_{z^-} &= \vec{H}_z^n |_{z^-} + \mu \left( \frac{H_z^{n+1} |_{z^-} - H_z^n |_{z^-}}{\Delta z} \right)
\end{align*}
\]

Grid Resolution
Consideration #1: Wavelength

The grid resolution must be sufficient to resolve the shortest wavelength.

First, determine the smallest wavelength:

\[ \lambda_{\text{min}} = \frac{c_0}{f_{\text{max}} n_{\text{max}}} \]

\( n_{\text{max}} \) is the largest refractive index found anywhere in the grid. \( f_{\text{max}} \) is the maximum frequency in your simulation.

Second, resolve the shortest wavelength with at least 10 cells.

\[ \Delta \lambda \approx \frac{\lambda_{\text{min}}}{N_{\lambda}} \quad N_{\lambda} \geq 10 \]

<table>
<thead>
<tr>
<th>( N_{\lambda} )</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 to 20</td>
<td>Low contrast dielectrics</td>
</tr>
<tr>
<td>20 to 30</td>
<td>High contrast dielectrics</td>
</tr>
<tr>
<td>40 to 60</td>
<td>Most metallic structures</td>
</tr>
<tr>
<td>100 to 200</td>
<td>Plasmonic devices</td>
</tr>
</tbody>
</table>

Consideration #2: Mechanical Features

The grid resolution must be sufficient to resolve the smallest mechanical features of the device.

First, determine the smallest feature:

\[ d_{\text{min}} \]

Second, resolve the smallest feature with at least 1 to 4 cells.

\[ \Delta d \approx \frac{d_{\text{min}}}{N_{d}} \quad N_{d} \geq 1 \]
Calculating the Initial Grid Resolution

Must resolve the minimum wavelength

\[ \Delta_x = \frac{\lambda_{\text{min}}}{N_x} \quad N_x \geq 10 \]

Must resolve the minimum structural dimension

\[ \Delta_d = \frac{d_{\text{min}}}{N_d} \quad N_d \geq 1 \]

Set the initial grid resolution to the smallest number computed above

\[ \Delta_x = \Delta_y = \min[\Delta_x, \Delta_d] \]

Resolving Critical Dimensions (1 of 3)

We have not yet considered the actual dimensions of the device we wish to simulate.

This means we likely cannot resolve the exact dimensions of a device.

Suppose we wish to place a device of length \( d \) onto a grid.
Resolving Critical Dimensions (2 of 3)

To fix this, we first calculate how many cells $N$ are needed to resolve the most important dimension. In this case, let this be $d$.

$$ N = \frac{d}{\Delta_x} $$

$N = 10.5$ cells

Second, we determine how many cells we wish to exactly resolve $d$. We do this by rounding $N$ up to the nearest integer.

$$ N' = \text{ceil} \left( \frac{d}{\Delta_x} \right) $$

$N' = 11$ cells

Resolving Critical Dimensions (3 of 3)

Third, we adjust the value of $\Delta_x$ to represent the dimension $d$ exactly.

$$ \Delta'_x = \frac{d}{N'} $$

We call this step “snapping” the grid to a critical dimension.

Unfortunately, using a uniform grid, we can only do this for one dimension per axis.
“Snap” Grid to Critical Dimensions

Decide what dimensions along each axis are critical

- Typically this is a lattice constant or grating period along $x$
- Typically this is a layer thickness along $y$

$d_x$ and $d_y$

Compute how many grid cells comprise $d_x$ and $d_y$ and round UP

$M_x = \text{ceil}(d_x / \Delta_x)$

$M_y = \text{ceil}(d_y / \Delta_y)$

Adjust grid resolution to fit these dimensions in grid EXACTLY

$\Delta_x = d_x / M_x$

$\Delta_y = d_y / M_y$

Courant Stability Condition
Numerical Propagation Through Grid

During a single iteration, a disturbance in the electric field at one point can only be felt by the immediately adjacent magnetic fields. It takes at least two time steps before that disturbance is felt by an adjacent electric field. This is simply due to how the update equations are implemented during a single iteration.

Physical Propagation Through Space

Electromagnetic waves in vacuum propagate at the speed of light. Inside a material, they propagate at a reduced speed.

\[ v = \frac{c_0}{n} \quad c_0 = 299792458 \text{ m/s} \quad n \equiv \text{refractive index} \]
A Limit on $\Delta t$

Over a time duration of one time step $\Delta t$, an electromagnetic disturbance will travel:

Numerical distance covered in one time step: $\Delta z$

Physical distance covered in one time step: $c_0\Delta t / n$

Because of the numerical algorithm, it is not possible for a disturbance to travel farther than one unit cell in a single time step. We need to make sure that a physical wave would not propagate farther than a single unit cell in one time step.

$$\frac{c_0\Delta t}{n} < \Delta z$$

This places an upper limit on the time step.

$$\Delta t < \frac{n\Delta z}{c_0}$$

$n$ should be set to the smallest refractive index found anywhere in the grid. Usually this is just made to be 1.0 and dropped from the equation.

The Courant Stability Condition

Refractive index is greater than or equal to one, so our condition on $\Delta t$ can be written more simply as:

$$\Delta t < \frac{\Delta z}{c_0}$$

Sort of a worst case. $n=1$ produces the fastest possible physical wave.

For 2D or 3D grids, the condition can be generalized as

$$\Delta t < \frac{1}{c_0 \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}}$$
Practical Implementation of the Stability Condition

The stability condition will be most restrictive along the shortest dimension of the grid unit cell.

$$\Delta_{\text{min}} = \min[\Delta x, \Delta y, \Delta z]$$

A good equation to ensure stability and accuracy on any grid is then

$$\Delta t < \frac{\Delta_{\text{min}}}{2c_0}$$

Note: A factor of 0.5 was included here as a safety margin.

This can be generalized to account for special cases.

$$\Delta t < \frac{n_{\text{min}}\Delta_{\text{min}}}{2c_0}$$

1. Your grid is filled with dielectric and travels slower everywhere.
2. Your model includes dispersive materials with refractive index less than one.

Time Step for Our 1D Grid

$$\Delta t = \frac{n_{bc}\Delta z}{2c_0}$$

$n_{bc}$ = refractive index at the grid boundaries.

This means a wave will travel the distance of one grid cell in exactly two time steps.

It is a necessary condition for the perfect boundary condition we will soon implement.

This implies that we cannot have different materials at the two boundaries using this boundary condition.
Perfect 1D Boundary Condition

The Problem

The finite-difference equation here requires knowledge of the electric field outside of the grid.

\[ \tilde{H}_x^N \bigg|_{l+\frac{1}{2}} = \tilde{H}_x^N \bigg|_{l-\frac{1}{2}} + m_{Hx}^k \left( \frac{E_{Ex}^{N+1}}{v} \frac{E_{Ey}^N}{\Delta z} \right) \]
Implementing the Perfect Boundary Condition

If and only if...

• the field is only travelling outward at the boundaries,
• the materials at the boundaries are linear, homogeneous, isotropic and non-dispersive,
• The refractive index at both boundaries is $n_{bc}$,
• $\Delta t = \frac{n_{bc}\Delta z}{2c_0}$ exactly.

Then...

$$E_y^{N+1} |_{t} = E_y^{N} |_{t-2\Delta t}$$

Visualizing the Perfect Boundary Condition

$$E_x^{N} |_{z=0} = E_x^{N-2}$$
Summary of the 1D Perfect Boundary Condition

Conditions
• Waves at the boundaries are only travelling outward,
• Materials at the boundaries are linear, homogeneous, isotropic and non-dispersive,
• The refractive index is the same at both boundaries and is \( n_{bc} \),
• Time step is chosen so physical waves travel 1 cell in exactly two time steps \( \Delta t = n_{bc} \Delta z/(2c_0) \).

Implementation at \( z \)-Low Boundary
At the \( z \)-low boundary, we need only modify the E-field update equation.

\[
\begin{align*}
h_2 &= h_1 \\
h_1 &= \tilde{H}_s \\
E^i_1 \big|_{z=\Delta z} &= E^i_0 + m E^i_0 \left( \frac{\tilde{H}^i_{1+\frac{1}{2}} - h_2}{\Delta z} \right)
\end{align*}
\]

Implementation at \( z \)-High Boundary
At the \( z \)-high boundary, we need only modify the H-field update equation.

\[
\begin{align*}
e_2 &= e_1 \\
e_1 &= E^N_\gamma \\
\tilde{H}^N_{1-\frac{1}{2}} &= \tilde{H}_s^{N-\frac{1}{2}} + m E^i_0 \left( \frac{e_2 - E^N_\gamma}{\Delta z} \right)
\end{align*}
\]

Sources
The Gaussian Pulse

In FDTD, we typically use short pulses as sources. This approximates an impulse function so we can excite a single simulation with a broad range of frequencies at the same time.

\[
g(t) = \exp\left[-\left(\frac{t-t_0}{\tau}\right)^2\right]
\]

Frequency Content of Gaussian Pulse

The Fourier transform of a Gaussian pulse is another Gaussian function.

\[
g(t) = \exp\left(-\frac{t^2}{\tau^2}\right) \quad \iff \quad G(f) = \frac{1}{\sqrt{\pi \tau}} \exp\left[-\frac{f^2}{\frac{1}{\tau^2}}\right]
\]

The frequency content of the Gaussian pulse extends from DC up to above \(f_{\text{max}}\). The frequency \(f_{\text{max}}\) is actually the 1/e point of the frequency spectrum.

\[
f_{\text{max}} = \frac{1}{\pi \tau}
\]
Designing the Pulse Source (1 of 2)

**Step 1:** Determine the maximum frequency of interest in your simulation.

\[ f_{\text{max}} \]

**Step 2:** Compute the pulse duration to have sufficient energy at \( f_{\text{max}} \).

\[ f_{\text{max}} = \frac{1}{\pi \tau} \quad \Rightarrow \quad \tau \leq \frac{1}{\pi f_{\text{max}}} \]

\[ \tau \approx \frac{0.5}{f_{\text{max}}} \]

**Step 3:** You may need to reduce your time step. Your Gaussian pulse should be resolved by at least 10 to 20 time steps.

\[ \Delta t = \frac{\tau}{N_t} \]

Typically, you determine a first \( \Delta t \) based on the Courant stability condition, then determine a second \( \Delta t \) based on resolving the maximum frequency, and finally go with the smallest value of the two \( \Delta t \)’s.

\[ N_t \geq 10 \]

All of this should be satisfied automatically if \( \Delta t = n\Delta z/(2c_0) \).

Designing the Pulse Source (2 of 2)

**Step 4:** Compute the delay time \( t_0 \)

The pulse source must start at zero and gradually increase. NO STEP FUNCTIONS!!

\[ t_0 \geq 3\tau \]

**WRONG!!**

The step function at the beginning will induce very large field gradients.

**STILL WRONG!!**

While better, this source still starts with a step function that will produce large field gradients.

**CORRECT!!**

This source “eases” into the Gaussian source.
Visualizing the Arrays

Usually several hundred points.

Ey, Hx, ER, UR, mEy, and mHx are stored in arrays of length Nz.

Two Ways to Incorporate a Simple Source

Source #1: Simple Hard Source
The simple hard source is the easiest to implement. After updating the field across the entire grid, one field component at one point on the grid is overwritten with the source. This approach injects power into the model, but the source point behaves like a perfect electric conductor or perfect magnetic conductor and will scatter waves.

\[
\hat{H}^4_{i, \pm} = g_H \bigg|_{k} \quad \text{and/or} \quad \hat{E}^4_{i, \pm} = g_E \bigg|_{k}
\]

OVERWRITE
Not usually a practical source.
We won’t use it.

Source #2: Simple Soft Source
The simple soft source is better than the hard source because it is transparent to scattered waves passing through it. After updating the field across the entire grid, the source function is added to one field component at one point on the grid. This approach injects power into the model in both directions. It is great for testing boundary conditions.

\[
\hat{H}^4_{i, \pm} = \hat{H}^4_{i, \pm} + g_H \bigg|_{k} \quad \text{and/or} \quad \hat{E}^4_{i, \pm} = \hat{E}^4_{i, \pm} + g_E \bigg|_{k}
\]

ADD TO
Rarely used.
Use this until we learn TF/SF.
Movie of Source #1 – Simple Hard Source

Movie of Source #2 – Simple Soft Source
Total Number of Iterations
Considerations for Estimating the Total Number of Iterations

The total number of iterations depends heavily on the device being modeled and what properties of it are being calculated.

Device Considerations
1. Highly resonant devices typically require more iterations.
2. Purely scattering devices typically require very few iterations.
3. More iterations are needed the more times the waves bounce around in the grid.

Information Considerations
1. Calculating abrupt spectral shapes requires many iterations.
2. Calculating fine frequency resolution requires many iterations.
3. Calculating only the approximate position of resonances often requires fewer iterations. Great for hunting resonances!

A Rule of Thumb

How long does it take a wave to propagate across the grid (worst case)?

\[
\text{time} = \frac{\text{distance}}{\text{velocity}} \rightarrow t_{\text{prop}} = \frac{n_{\text{max}} N \Delta z}{c_0}
\]

Simulation time \( T \) must include the entire pulse of duration \( \tau \).

\[ T \geq 12 \tau \]

Simulation time should allow for \( \sim 5 \) bounces.

\[ T \geq 5 \Delta t_{\text{prop}} \]

A rule-of-thumb for total simulation time is then

\[ T = 12 \tau + 5 \Delta t_{\text{prop}} \quad \text{Note: For highly resonant devices, this will NOT be enough time!} \]

Given the time step \( \Delta t \), the total number of iterations is then

\[ \text{STEPS} = \text{round} \left( \frac{T}{\Delta t} \right) \quad \text{This must be an integer quantity.} \]
Revised FDTD Algorithm

- Compute Grid Resolution
  \[ \Delta x = \min \left[ \frac{d_x}{N_x}, \frac{d_y}{N_y}, \frac{d_z}{N_z} \right] \]
  \[ N = \text{round} \left[ \frac{d}{\Delta x} \right] \]
  \[ \Delta t = \frac{\Delta x}{c} \]

- Build Device
  EMpy and UEM

- Compute Time Step
  \[ \Delta t = \frac{c}{N \Delta x} \]

- Compute Source
  \[ g(t) = \exp \left[ -\frac{t - t_c}{\tau} \right] \]

- Compute Update Coeff's
  \[ c_s = \frac{c_m}{\varepsilon_s}, m_s = \frac{\mu_s}{\varepsilon_m} \]

- Initialize Fields
  \[ E_0^x = 0, H_0^z = 0 \]

- Initialize Boundary Terms
  \[ h_0 = b_0 = \epsilon_0 = \mu_0 = 0 \]

- Loop over time

  \[ t \]

  - Inject Source
    \[ E_y^j = \frac{g(t) \sigma_y}{\mu_0} \]

  - Visualize fields

  - Done?

  - yes

  - no

  - Update \( E \) from \( H \)
    \[ E_y^{k+1} = \frac{1}{k} \left( E_y^k - \Delta t \left( \mu_s \frac{\partial H_y^k}{\partial z} + \epsilon_s \frac{\partial H_z^k}{\partial x} \right) \right) \]
    \[ E_z^k = \frac{k - 1}{k} E_z^k \]

  - Update \( H \) from \( E \)
    \[ H_y^{k+1} = \frac{1}{k} \left( H_y^k - \Delta t \left( \epsilon_s \frac{\partial E_x^k}{\partial z} - \mu_s \frac{\partial E_z^k}{\partial x} \right) \right) \]
    \[ H_z^{k+1} = \frac{k - 1}{k} H_z^k \]

  - Record \( H \)-Field Boundary Term
    \[ b_0 = h_0, h_1 = H_1^z \]

  - Record \( E \)-Field Boundary Term
    \[ e_0 = e_0, e_1 = E_1^x \]

  - Include:
    - Basic FDTD engine
    - Dirichlet BC's
    - Calculate source parameters
    - Simple soft source
    - Perfectly absorbing BC's

  - Exclude:
    - RF/RF source
    - Fourier transforms
    - Reflectance/transmittance
    - Calculate grid parameters
    - Incorporate device

- Finished!
Equations → MATLAB Code

Grid Resolution

\[ \Delta z' = \min \left( \frac{\Delta z}{N_d}, \frac{d}{N_c} \right) \]

\[ N = \text{round} \left( \frac{d}{\Delta z} \right) \]

\[ \Delta z = d / N \]

% COMPUTE DEFAULT GRID RESOLUTION

dz1 = min(LAMBDA) / nmax / NRES;
dz2 = min(dmin/NDRES);
dz = min(dz1, dz2);

% SNAP GRID TO CRITICAL DIMENSIONS

N  = ceil(dc/dz);
dz = dc/N;

% MAIN FDTD LOOP

% for T = 1 : STEPS

Update Equations

\[ R_{1z} = R_{2z} + \Delta z \left( E_{1z} - E_{2z} \right) \quad k < N_z \]

\[ R_{kz} = R_{2z} + \Delta z \left( E_{k-1z} - E_{kz} \right) \quad k > 1 \]

\[ R_{N_z} = R_{2z} + \Delta z \left( E_{N_z-1z} - E_{N_zz} \right) \quad k = N_z \]

\[ E_{1z} = E_{1z} + m_{Ez} \left( E_{1z} - R_{1z} \right) \quad k < 1 \]

\[ E_{kz} = E_{kz} + m_{Ez} \left( E_{kz} - R_{kz} \right) \quad k > 1 \]

\[ E_{N_zz} = E_{N_zz} + m_{Ez} \left( E_{N_zz} - R_{N_zz} \right) \quad k = N_z \]

% Update H from E (Perfect Boundary Conditions)

\[ H_{2z} = H_{1z}; \quad H_{1z} = Hx(1); \]

\[ \text{for} \ nz = 1 : N_z - 1 \]

\[ Hx(nz) = Hx(nz) + m_{Hx}(nz) \left( E_{y}(nz+1) - E_{y}(nz) \right) / dz; \]

\[ \text{end} \]

\[ Hx(N_z) = Hx(N_z) + m_{Hx}(N_z) \left( E_{2z} - E_{y}(N_z) \right) / dz; \]

% Update E from H (Perfect Boundary Conditions)

\[ E_{2z} = E_{1z}; \quad E_{1z} = Ey(N_z); \]

\[ \text{for} \ nz = 2 : N_z \]

\[ Ey(nz) = Ey(nz) + m_{Ey}(nz) \left( Hx(nz) - Hx(nz-1) \right) / dz; \]

\[ \text{end} \]

% Inject Soft Source

\[ Ey(nzsrc) = Ey(nzsrc) + g(T); \]

% end

% for T = 1 : STEPS

end

end

end

end