FDFD (Finite-Difference Frequency Domain)

Basic Idea: use same spatial discretization as FDTD, but work in frequency domain.

Use time harmonic fields: $E(r,t) = \mathrm{Re}\{E(r)e^{i\omega t}\}$

Maxwell's equations become

$$\nabla \times E = -\mu \partial_t H \qquad\longrightarrow\qquad \nabla \times E = -i\omega\mu H$$

$$\nabla \times H = \varepsilon \partial_t E + J \qquad\qquad \nabla \times H = i\omega\varepsilon E + J$$

2D case: TM polarization (E_z, H_x, H_y only)

$$\left.\begin{aligned} \partial_y E_z &= -i\omega\mu H_x \\ -\partial_x E_z &= -i\omega\mu H_y \\ \partial_x H_y - \partial_y H_x &= i\omega\varepsilon E_z + J_z \end{aligned}\right\}$$

Same equations as we used in FDTD, except $\partial_t \rightarrow i\omega$

Solving for E_z,

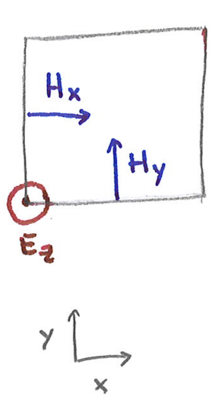
$$\partial_x \mu^{-1}\partial_x E_z + \partial_y \mu^{-1}\partial_y E_z + \omega^2\varepsilon E_z = i\omega J_z$$

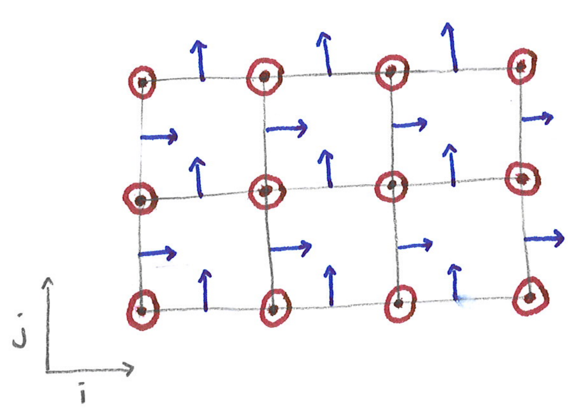
$$\underbrace{\left[\partial_x \mu^{-1}\partial_x + \partial_y \mu^{-1}\partial_y + \omega^2\varepsilon\right]}_{A}\underbrace{E_z}_{e_z} = \underbrace{i\omega J_z}_{b}$$

We now have a linear system of eq'ns for E_z!

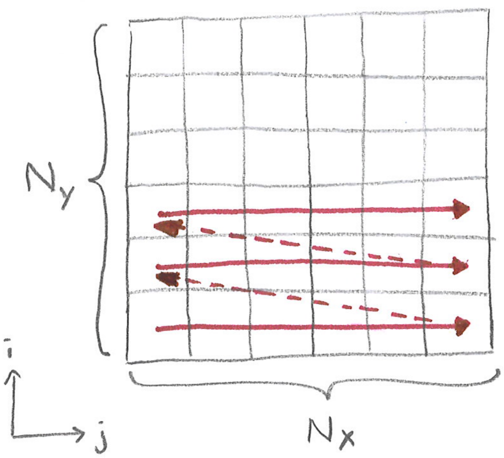
Formulating FDFD

We use the same Yee lattice as for FDTD:





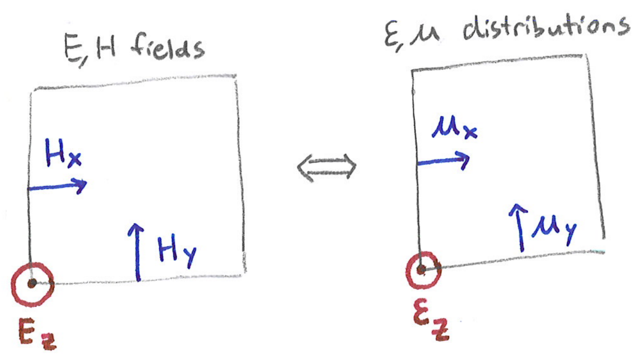
Step 1: reshape E_z^{ij}, H_x^{ij}, H_y^{ij} into vectors.



$$E_z^{ij} \longrightarrow e_z = \begin{bmatrix} E_z^{1,1} \\ E_z^{2,1} \\ \vdots \\ E_z^{N_x,1} \\ \hline E_z^{1,2} \\ E_z^{2,2} \\ \vdots \\ E_z^{N_x,2} \\ \hline \vdots \\ E_z^{N_x,N_y} \end{bmatrix}$$

We store the 2D array <u>row-by-row</u>.

We also need to vectorize the permittivity ε and permeability μ, which are <u>co-located</u> with the E and H fields:



* We vectorize $\varepsilon_z, \mu_x, \mu_y$ in the <u>same</u> way as E_z, H_x, and H_y.

Vectorized Notation

$E_z \rightarrow e_z$	$J_z \rightarrow j_z$
$H_x \rightarrow h_x$	$\mu \rightarrow \mu_x, \mu_y$
$H_y \rightarrow h_y$	$\varepsilon \rightarrow \varepsilon_z$

Step 2: Differential operators

We want to represent the differential operators as <u>matrices</u>:

$\partial_x \longrightarrow D_x^f$ and D_x^b where $f \rightarrow$ forward derivative

$\partial_y \longrightarrow D_y^f$ and D_y^b $\quad b \rightarrow$ backward derivative

We use exactly the same central differencing for the spatial derivatives as FDTD. However, we need to switch to integer indexing (rather than the half-integer indexing we used for FDTD) because we are using a vectorized representation of fields.

X-derivative:

$$[\partial_x E_z]^{ij} = \frac{E_z^{i+1,j} - E_z^{ij}}{\Delta x} \qquad \text{Forward Derivative}$$

$$[\partial_x H_y]^{ij} = \frac{H^{ij} - H^{i-1,j}}{\Delta x} \qquad \text{Backwards Derivative}$$

$$D_x^f = \frac{1}{\Delta x} \underbrace{\left(\begin{array}{c|c|c} \underbrace{\begin{matrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{matrix}}_{N_x \times N_x} & & \\ \hline & \begin{matrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 1 & & & & -1 \end{matrix} & \\ \hline & & \ddots \end{array}\right)}_{N_y \text{ copies}}$$

D_x^b is the same, but shifted by 1 column:

$$D_x^b = \left(\begin{array}{c|c|c} \begin{matrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{matrix} & & \\ \hline & \begin{matrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{matrix} & \\ \hline & & \ddots \end{array}\right)$$

Note: we are using periodic boundary conditions.

y-derivative:

$$[\partial_y E_z]^{ij} = \frac{E_z^{i,j+1} - E_z^{ij}}{\Delta y} \qquad \text{Forward Derivative}$$

$$[\partial_y H_x]^{ij} = \frac{H_x^{ij} - H_x^{i,j-1}}{\Delta y} \qquad \text{Backwards Derivative}$$

$$D_y^f = \frac{1}{\Delta y} \underbrace{\left(\begin{array}{c|c|c|c}
\underbrace{\begin{matrix} -1 & & \\ & \ddots & \\ & & -1 \end{matrix}}_{N_x * N_x} & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} & & \\
\hline
 & \begin{matrix} -1 & & \\ & \ddots & \\ & & -1 \end{matrix} & \begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} & \\
\hline
 & & \ddots & \\
\hline
\begin{matrix} 1 & & \\ & \ddots & \\ & & 1 \end{matrix} & & & \begin{matrix} -1 & & \\ & \ddots & \\ & & -1 \end{matrix}
\end{array}\right)}_{N_y \text{ copies}} = \frac{1}{\Delta y} \begin{pmatrix} -I_x & I_x & & & \\ & -I_x & I_x & & \\ & & \ddots & & \\ & & & -I_x & I_x \\ I_x & & & & -I_x \end{pmatrix}$$

where $I_x \in \mathbb{R}^{N_x * N_x}$

D_y^b is the same, but shifted by one block:

$$D_y^b = \frac{1}{\Delta y} \begin{pmatrix} I_x & & & & -I_x \\ -I_x & I_x & & & \\ & & \ddots & & \\ & & -I_x & I_x & \\ & & & -I_x & I_x \end{pmatrix}$$

$\boxed{\text{Step 3:}}$ Construct the matrix A

Rewriting our equations for the TM fields in linear algebra form,

$$D_y^f e_z = -i\omega T_{\mu_x} h_x \qquad \text{where} \qquad T_{\mu_x} = \text{diag}(\mu_x)$$

$$-D_x^f e_z = -i\omega T_{\mu_y} h_y \qquad\qquad T_{\mu_y} = \text{diag}(\mu_y)$$

$$D_x^b h_y - D_y^b h_x = i\omega T_{\varepsilon_z} e_z + j_z \qquad\qquad T_{\varepsilon_z} = \text{diag}(\varepsilon_z)$$

Solving for e_z yields our final matrix

$$\underbrace{\left[D_x^b T_{\mu_x}^{-1} D_x^f + D_y^b T_{\mu_y}^{-1} D_y^f + \omega^2 T_{\varepsilon_z} \right]}_{A} \underset{\downarrow}{e_z} = \underbrace{i\omega j_z}_{b}$$

$$e_z = b$$

We can solve this matrix equation using standard matrix solvers:

1) Direct Solvers: essentially uses Gaussian elimination
   - → slow and expensive in memory, but exact to numerical precision
   - → great for small problems
2) Iterative Solvers: iteratively minimizes error (residual) in matrix equation
   - → less memory intensive, and generally faster
   - → required for large problems

Finally, we reconstruct E_z by reshaping e_z.

Get time-domain field using:

$$E_z(x,y,t) = \text{Re}\{E_z(x,y,t)\,e^{i\omega t}\}$$