Introduction to finite-difference frequency-domain (FDFD) method

Wonseok Shin, Ph.D.
Applied Mathematics Instructor
wsshin@mit.edu

Massachusetts Institute of Technology

Why finite difference?

Finite difference method is intuitive and easy.

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x}$$

Choice of Maxwell's Equations

Time-domain

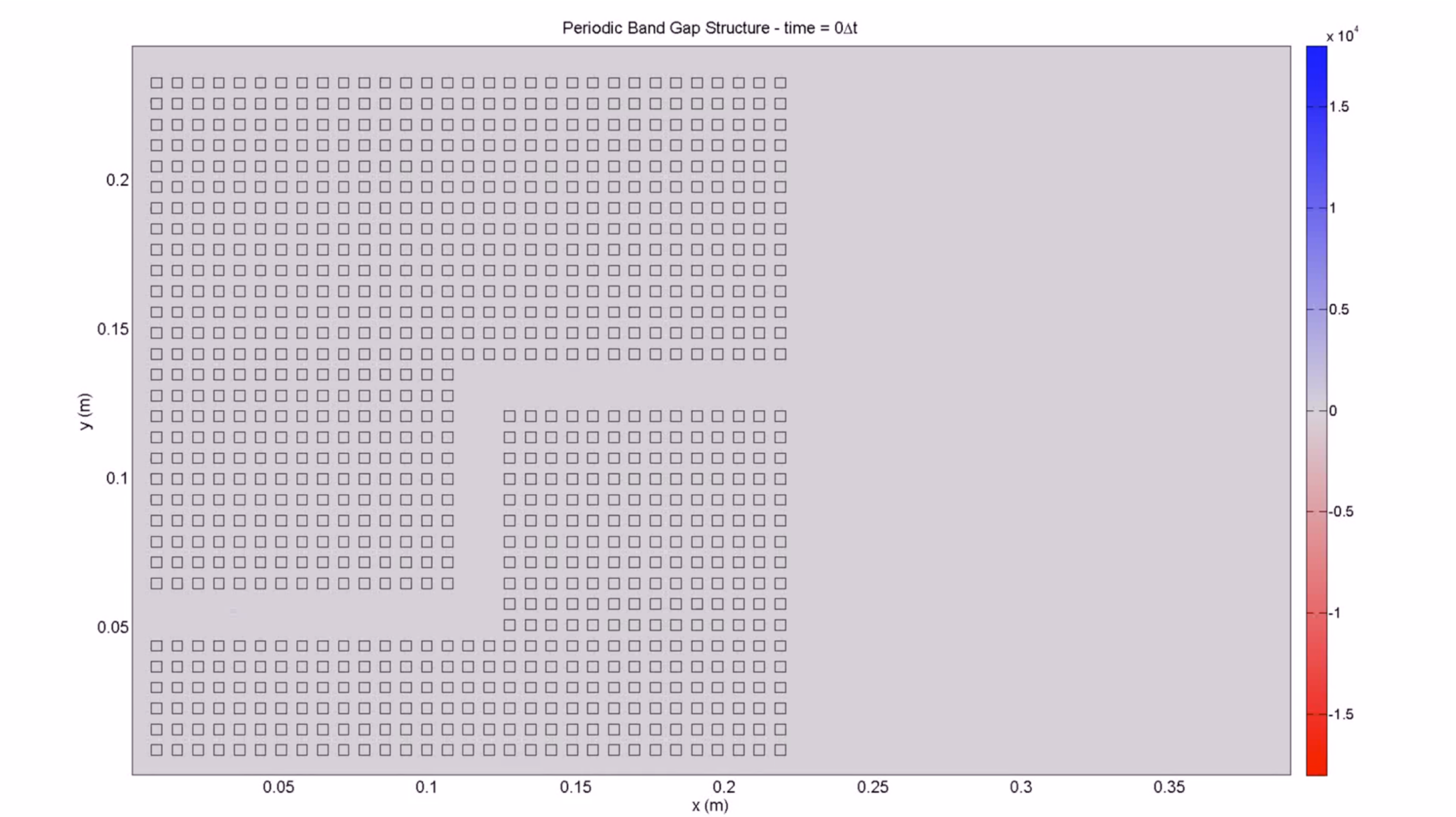
$$\nabla \times \mathbf{E} = -\partial_t \mathbf{B} = -\partial_t (\mu * \mathbf{H})$$
$$\nabla \times \mathbf{H} = \partial_t \mathbf{D} + \mathbf{J} = \partial_t (\varepsilon * \mathbf{E}) + \mathbf{J}$$

- **Shows the transient state.**
- **Steady state takes long. (Main drawback)**

Frequency-domain

$$\nabla \times \mathbf{E} = -i \omega \mu \mathbf{H}$$
$$\nabla \times \mathbf{H} = i \omega \varepsilon \mathbf{E} + \mathbf{J}$$

- **Does not show the transient state.**
- **Steady state obtained immediately.**



Finite-Difference Time-Domain (FDTD) Method

Time-domain Maxwell's eqs.

$$\nabla \times \mathbf{E} = -\partial_t \mathbf{B} = -\partial_t (\mu * \mathbf{H})$$
$$\nabla \times \mathbf{H} = \partial_t \mathbf{D} + \mathbf{J} = \partial_t (\varepsilon * \mathbf{E}) + \mathbf{J}$$

Finite-difference method:

$$\partial_x E_y \approx \frac{\Delta E_y}{\Delta x}, \quad \partial_t B_x \approx \frac{\Delta B_x}{\Delta t}, \quad \ldots$$

Time-domain drawback 2: uniform Δ*t*

Time-domain Maxwell's eqs.

$$\nabla \times \mathbf{E} = -\partial_t (\mu * \mathbf{H})$$
$$\nabla \times \mathbf{H} = \partial_t (\varepsilon * \mathbf{E}) + \mathbf{J}$$

Courant stability condition: $$\frac{\Delta l_{\min}}{\Delta t} \geq c$$

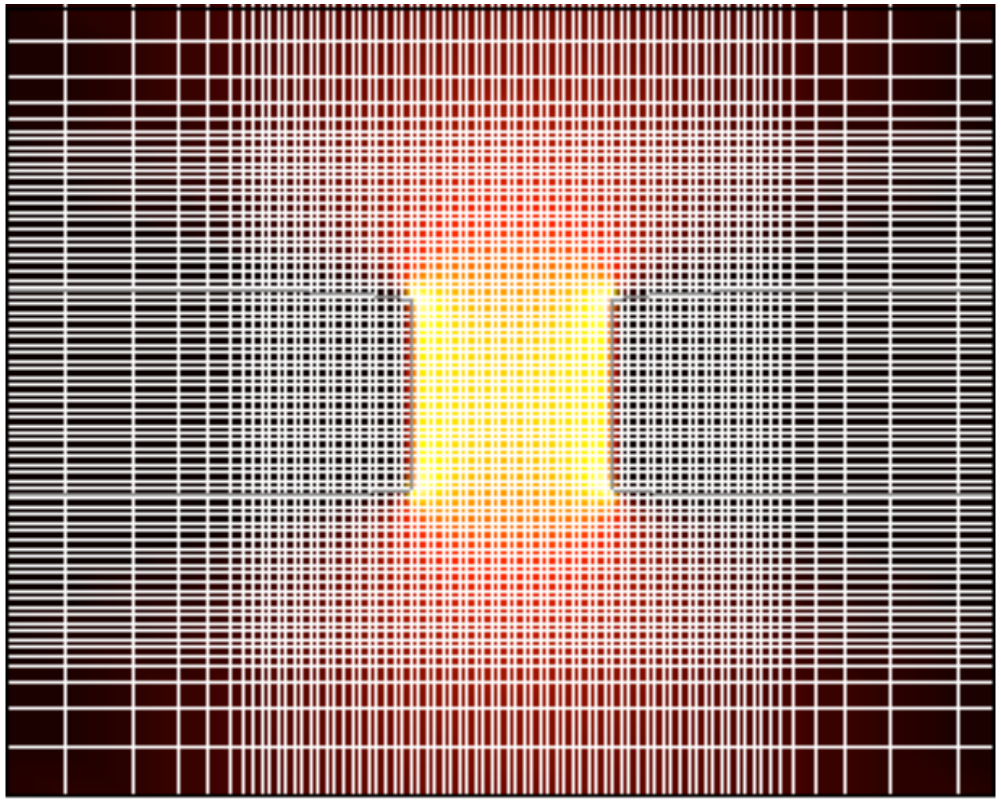
Time-domain drawback 2: uniform Δ*t*

Time-domain Maxwell's eqs.

$$\nabla \times \mathbf{E} = -\partial_t (\mu * \mathbf{H})$$
$$\nabla \times \mathbf{H} = \partial_t (\varepsilon * \mathbf{E}) + \mathbf{J}$$

Courant stability condition: $\dfrac{\Delta l_{\mathrm{min}}}{\Delta t} \geq c$

⇒ Slow for simulation with metallic objects

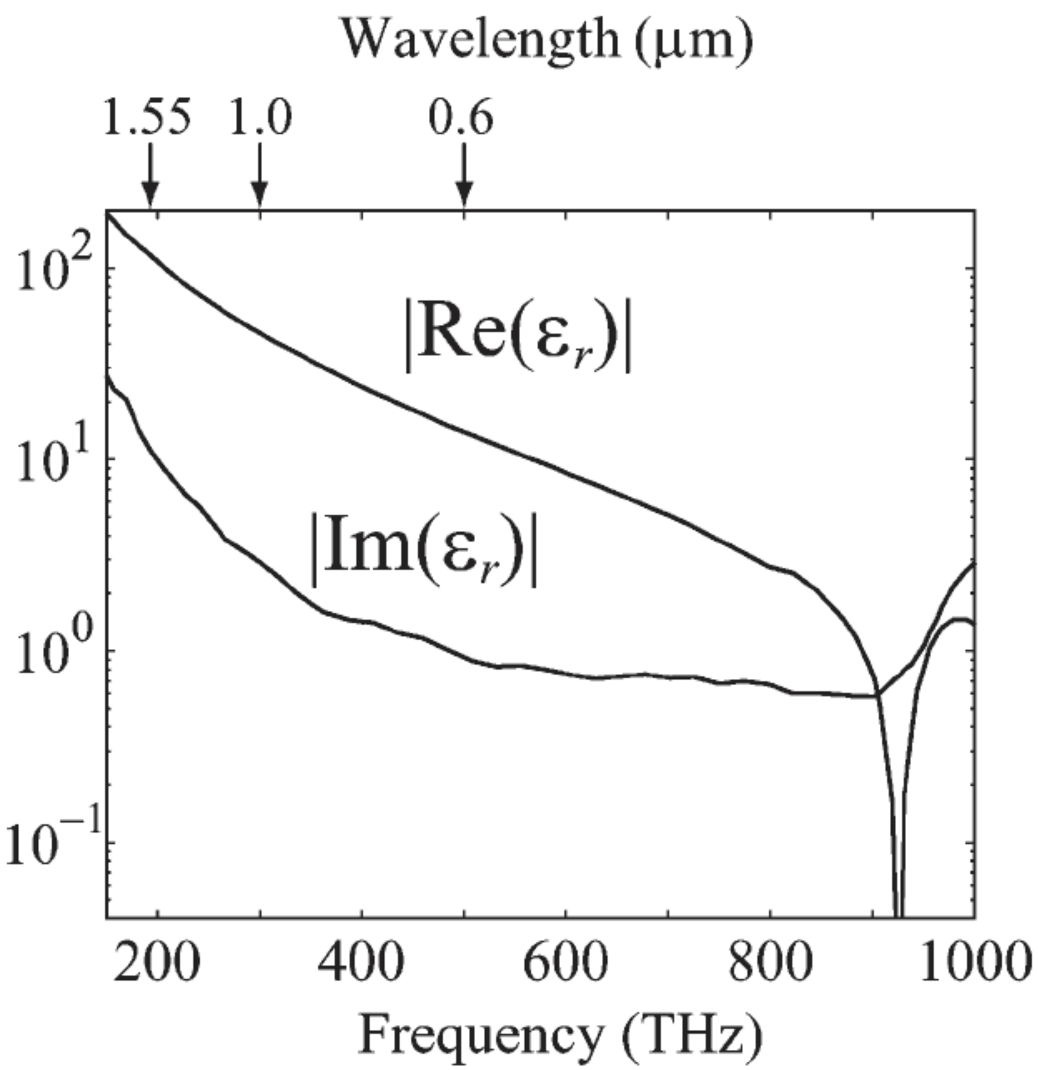


Time-domain drawback 3: modeling *ε* and *μ*

Time-domain Maxwell's eqs.

$$\nabla \times \mathbf{E} = -\partial_t (\mu * \mathbf{H})$$
$$\nabla \times \mathbf{H} = \partial_t (\varepsilon * \mathbf{E}) + \mathbf{J}$$

Inaccurate for dispersive materials



← Need to get *ε*(t) from this.

⇒ Fit *ε*(*ω*) to an analytic function, then FT.

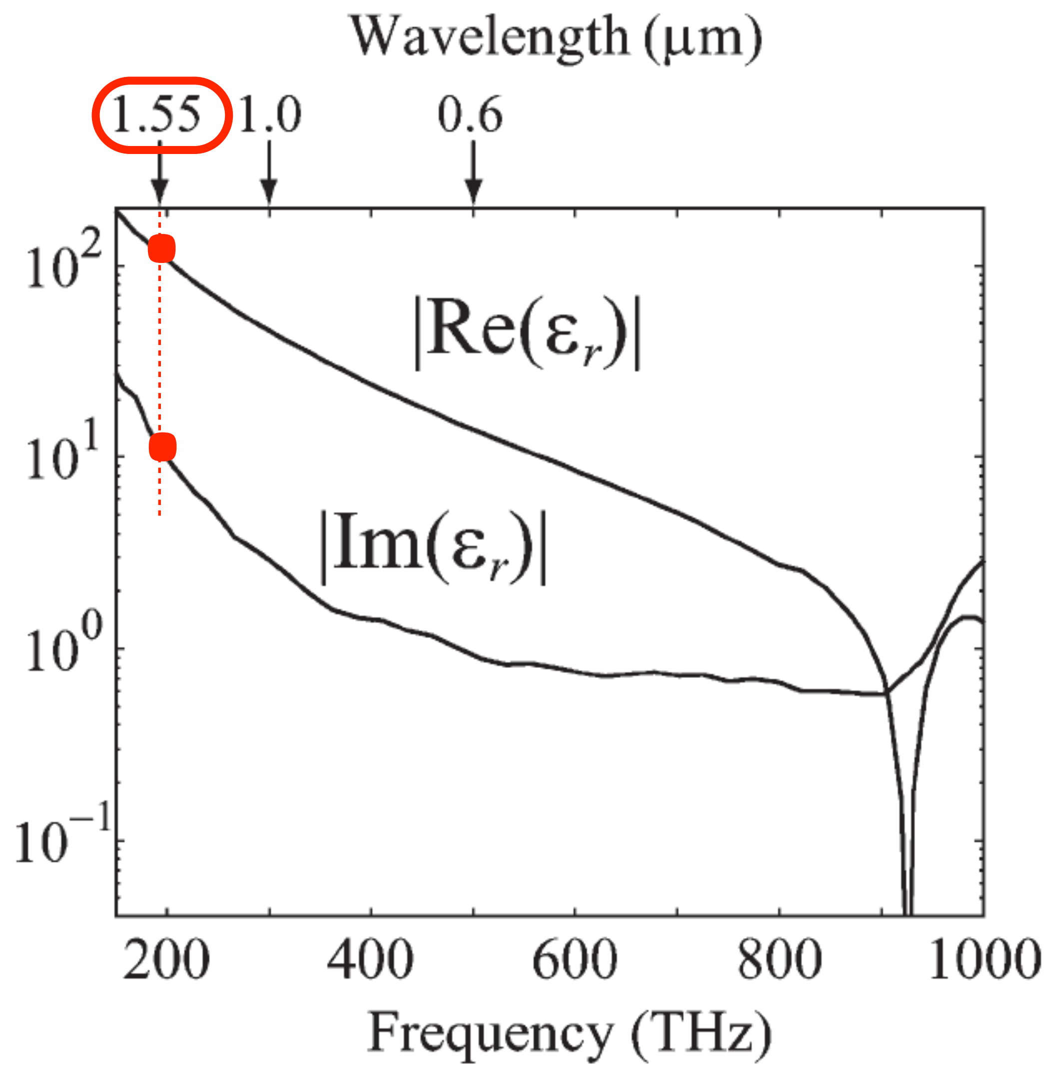
$$\varepsilon(\omega) = \varepsilon_\infty \left(1 + \sum_{i=1}^{N} \frac{\omega_{p,i}^2}{\omega_{0,i}^2 - \omega^2 + i\,\omega\,\Gamma_i} \right)$$

Solution: frequency-domain methods

Frequency-domain Maxwell's eqs.

$$\nabla \times \mathbf{E} = -i\,\omega\,\mu\,\mathbf{H}$$
$$\nabla \times \mathbf{H} = \mathbf{J} + i\,\omega\,\varepsilon\,\mathbf{E}$$

- **No Δt. $\Rightarrow$ No penalty for small Δl.**
- **Use measured material parameters at specific ω.**



Frequency Domain Equations

+

Finite Difference Method

↓

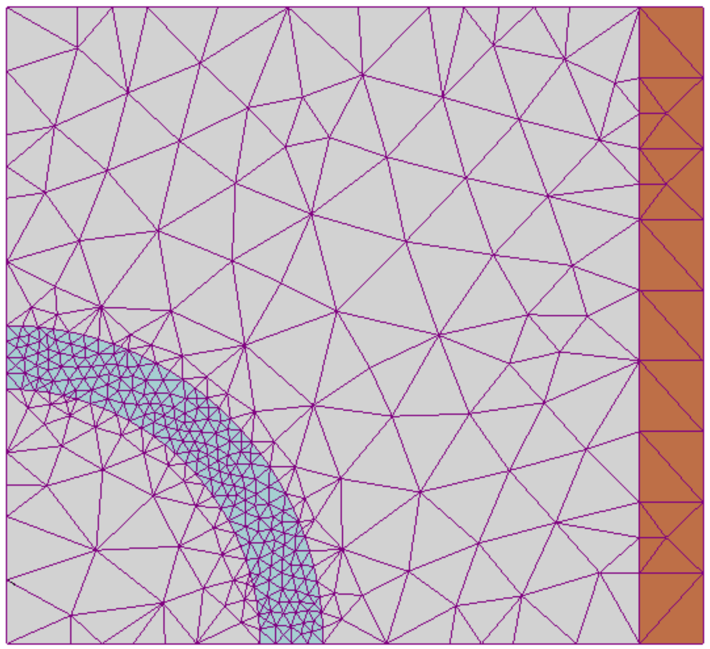
FDFD

Construction of $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$

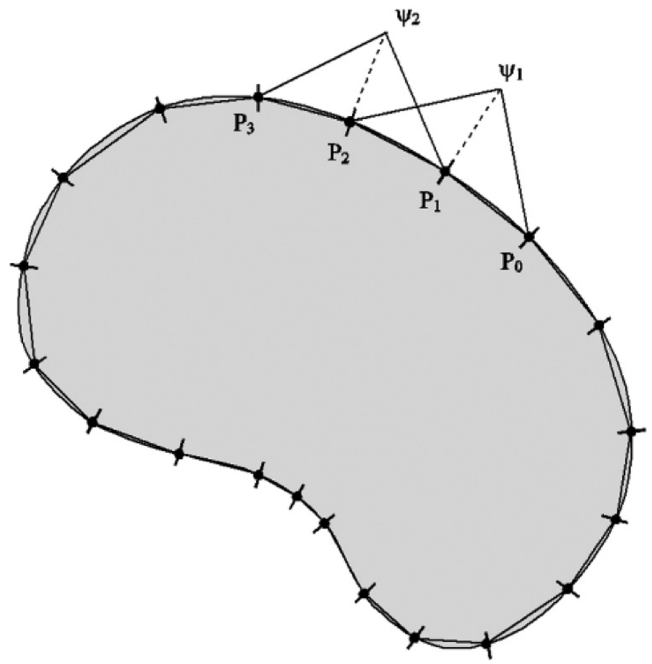
Discretize Maxwell eqs. ⇒ *A x = b*

Discretization Methods

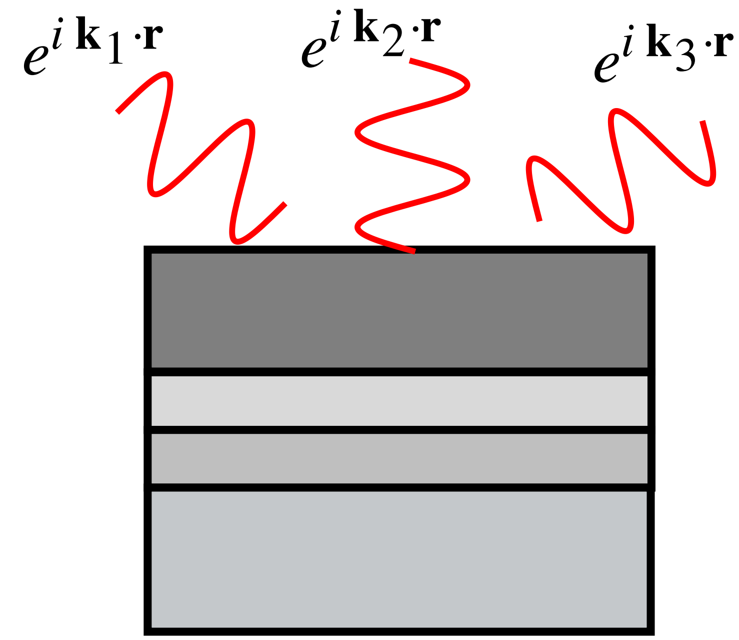
FEM



BEM



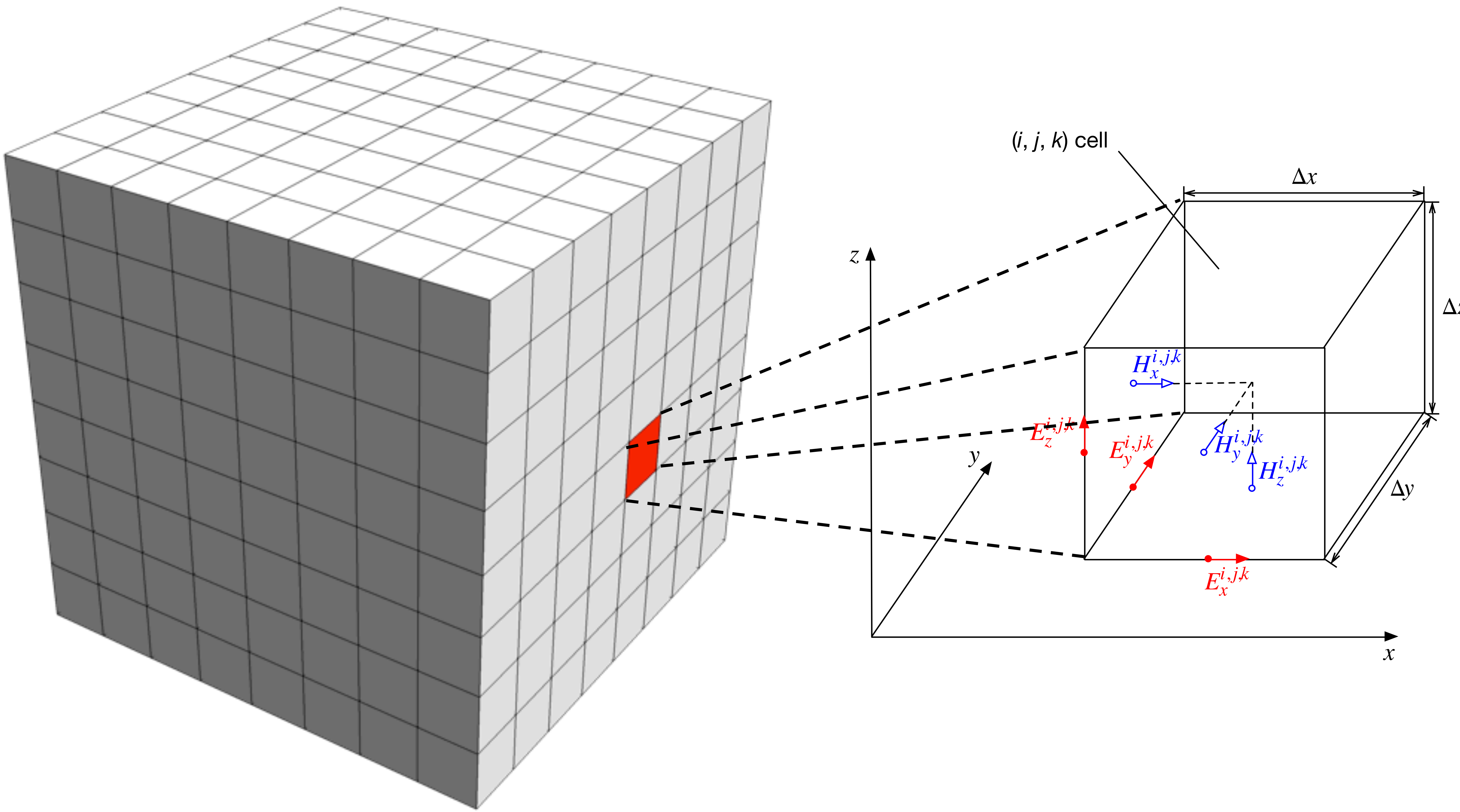
Spectral Method



$$\boxed{A\,x = b}$$

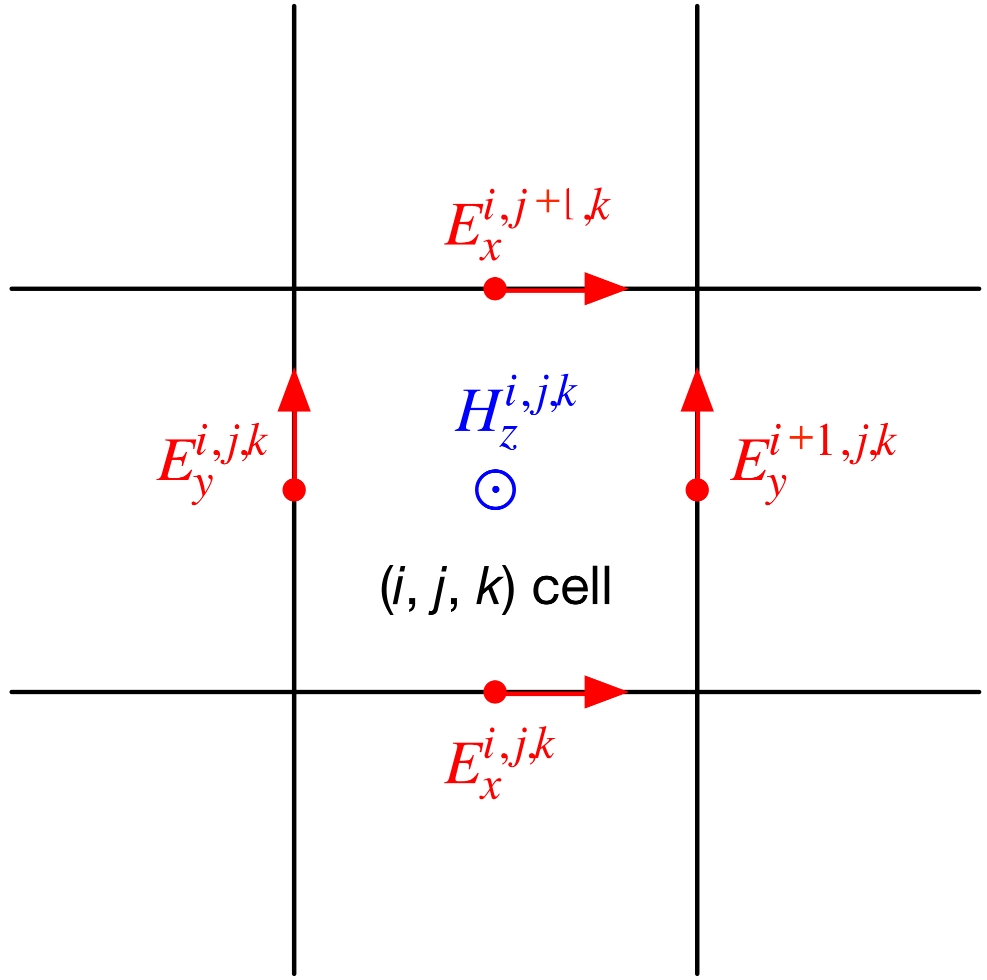
Numerical Linear Algebra Techniques

Finite-different discretization grid



Interlaced E and H grid: crucial for 2nd-order error!

***xy*-plane of grid:**



Faraday's law: $\nabla \times \mathbf{E} = -i\,\omega\,\mu\,\mathbf{H}$

***z*-component:** $\partial_x E_y - \partial_y E_x = -i\,\omega\,\mu\,H_z$

FD approximation: $\dfrac{\Delta E_y}{\Delta x} - \dfrac{\Delta E_x}{\Delta y} = -i\,\omega\,\mu\,H_z$

At (*i,j,k*): $\dfrac{E_y^{(i+1)jk} - E_y^{ijk}}{\Delta x} - \dfrac{E_x^{i(j+1)k} - E_x^{ijk}}{\Delta y} = -i\,\omega\,\mu_z^{ijk}\,H_z^{ijk}$

Interlaced E and H grid: crucial for 2nd-order error!

Forward difference: $$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Central difference: $$f'(x) \approx \frac{f(x+h) - f(x-h)}{2\,h}$$

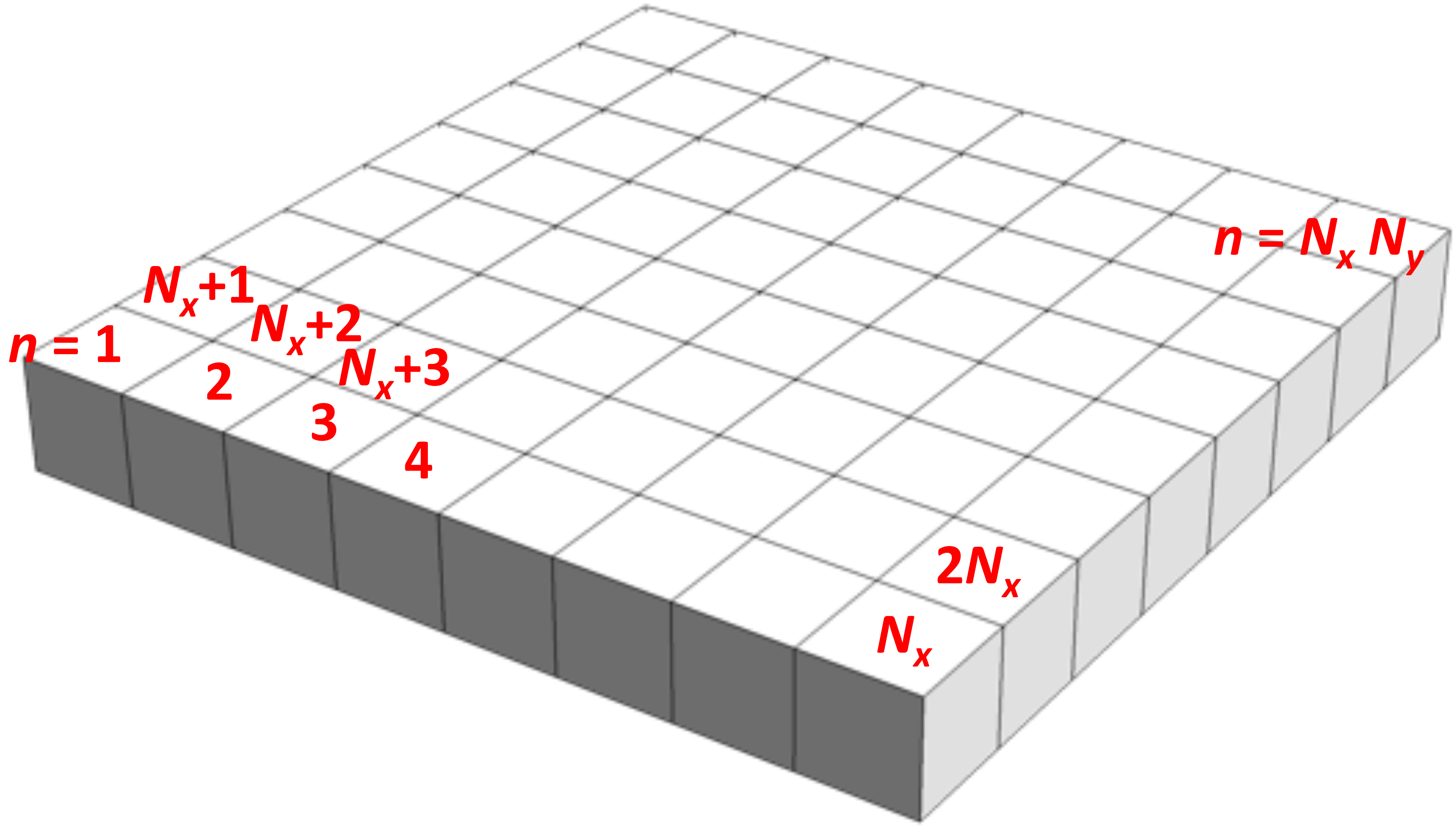
Taylor expansion: $$f(x+h) = f(x) + h\,f'(x) + \frac{1}{2}\,h^2\,f''(x) + \frac{1}{6}\,h^3\,f'''(x) \cdots$$

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2}\,h\,f''(x) + \frac{1}{6}\,h^2\,f'''(x) + \cdots = f'(x) + \boxed{O(h)}$$

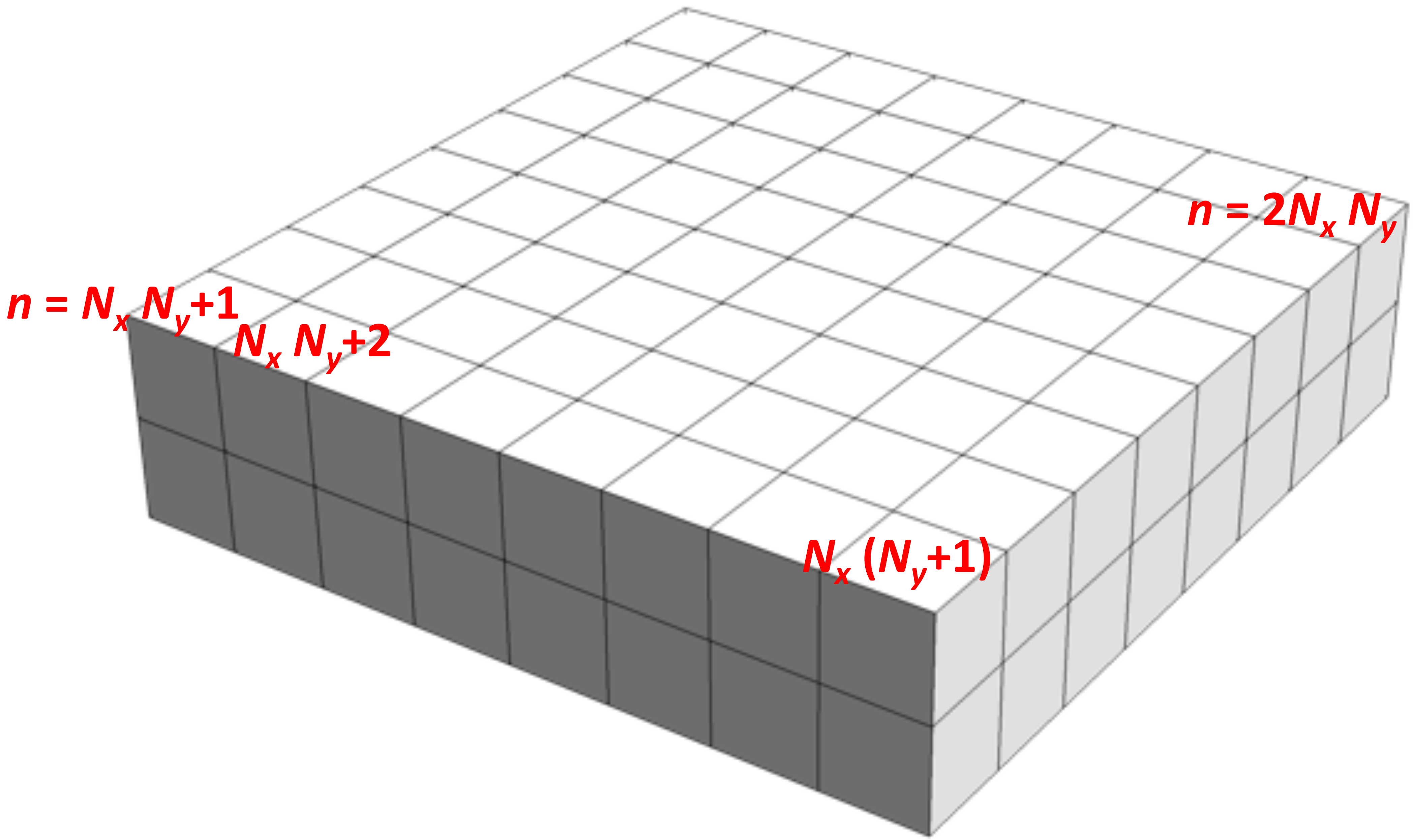
Taylor expansion:
(opposite direction) $$f(x-h) = f(x) - h\,f'(x) + \frac{1}{2}\,h^2\,f''(x) - \frac{1}{6}\,h^3\,f'''(x) \cdots$$

$$\frac{f(x+h) - f(x-h)}{2\,h} = f'(x) + \frac{1}{3}\,h^2\,f'''(x) + \cdots = f'(x) + \boxed{O\left(h^2\right)}$$

Linearize (i, j, k) to n



Linearize (i, j, k) to n



Linearize (*i*, *j*, *k*) to *n*

$$e_x = \begin{bmatrix} E_x^{111} \\ E_x^{211} \\ E_x^{311} \\ \vdots \\ E_x^{N_x N_y N_z} \end{bmatrix}, \; e_y = \cdots, \;\; e_z = \cdots$$

$$h_x = \begin{bmatrix} H_x^{111} \\ H_x^{211} \\ H_x^{311} \\ \vdots \\ H_x^{N_x N_y N_z} \end{bmatrix}, \; h_y = \cdots, \;\; h_z = \cdots$$

Collect discretized equations

z-comp of Faraday at (*i,j,k*): $\dfrac{E_y^{(i+1)jk} - E_y^{ijk}}{\Delta x} - \dfrac{E_x^{i(j+1)k} - E_x^{ijk}}{\Delta y} = -i\,\omega\,\mu_z^{ijk}\,H_z^{ijk}$

Collect from all points:

$$\frac{1}{\Delta x}\begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \end{bmatrix} e_y - \frac{1}{\Delta y}\begin{bmatrix} -1 & 1 & & \\ & -1 & 1 & \\ & & \ddots & \ddots \end{bmatrix} e_x = -i\,\omega \begin{bmatrix} \mu_z^{111} & & \\ & \mu_z^{211} & \\ & & \ddots \end{bmatrix} h_z$$

$$D_x^e\, e_y - D_y^e\, e_x = -i\,\omega\, T_\mu^z\, h_z$$

Collect *x*, *y*, *z*-comps:

$$\begin{bmatrix} & -D_z^e & D_y^e \\ D_z^e & & -D_x^e \\ -D_y^e & D_x^e & \end{bmatrix}\begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = -i\,\omega \begin{bmatrix} T_\mu^x & & \\ & T_\mu^y & \\ & & T_\mu^z \end{bmatrix}\begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix}$$

$$C_e\, e = -i\,\omega\, T_\mu\, h$$

$$\nabla \times \mathbf{E} = -i\,\omega\,\mu\,\mathbf{H}$$

Repeat for Ampere's law

Ampere's law: $\nabla \times \mathbf{H} = i\,\omega\,\varepsilon\,\mathbf{E} + \mathbf{J}$

Discretize:

$$\begin{bmatrix} & -D_z^h & D_y^h \\ D_z^h & & -D_x^h \\ -D_y^h & D_x^h & \end{bmatrix} \begin{bmatrix} h_x \\ h_y \\ h_z \end{bmatrix} = i\,\omega \begin{bmatrix} T_\varepsilon^x & & \\ & T_\varepsilon^y & \\ & & T_\varepsilon^z \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} + \begin{bmatrix} j_x \\ j_y \\ j_z \end{bmatrix}$$

$$C_h\,h = i\,\omega\,T_\varepsilon\,e + j$$

Faraday's law: $C_e\,e = -i\,\omega\,T_\mu\,h \iff h = i\,\omega^{-1}\,T_\mu^{-1}\,C_e\,e$

Eliminate *h*:

$$C_h\left(i\,\omega^{-1}\,T_\mu^{-1}\,C_e\right)e = i\,\omega\,T_\varepsilon\,e + j$$

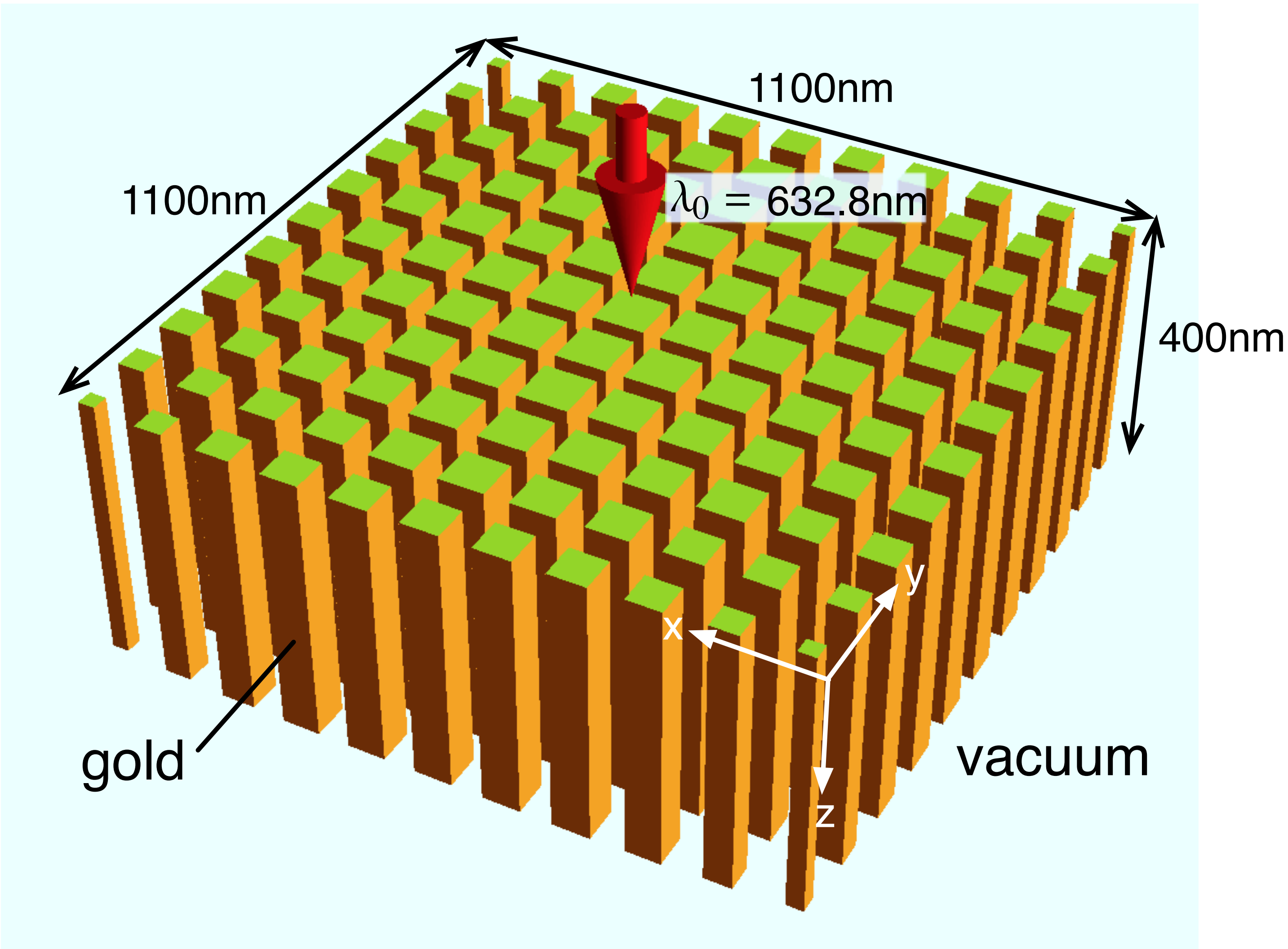
$$\left(C_h\,T_h^{-1}\,C_e - \omega^2\,T_\varepsilon\right)e = -i\,\omega\,j$$

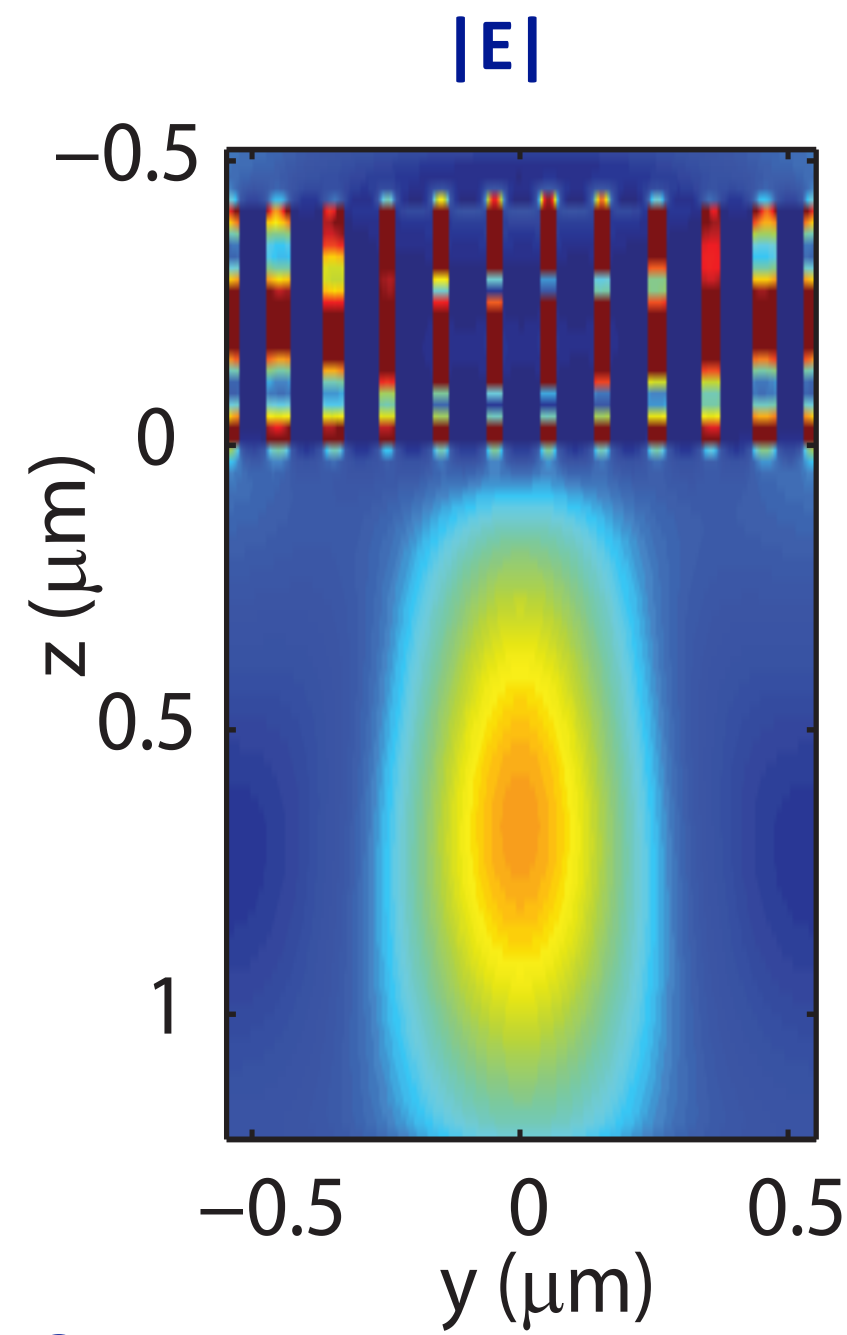
$$\left(C_h\, T_h^{-1}\, C_e - \omega^2\, T_\varepsilon\right) e = -i\,\omega\, j$$

$$A\,x = b$$

$$\left[\left(\nabla \times \mu^{-1}\ \nabla \times\right) - \omega^2\, \varepsilon\right] \mathbf{E} = -i\,\omega\,\mathbf{J}$$

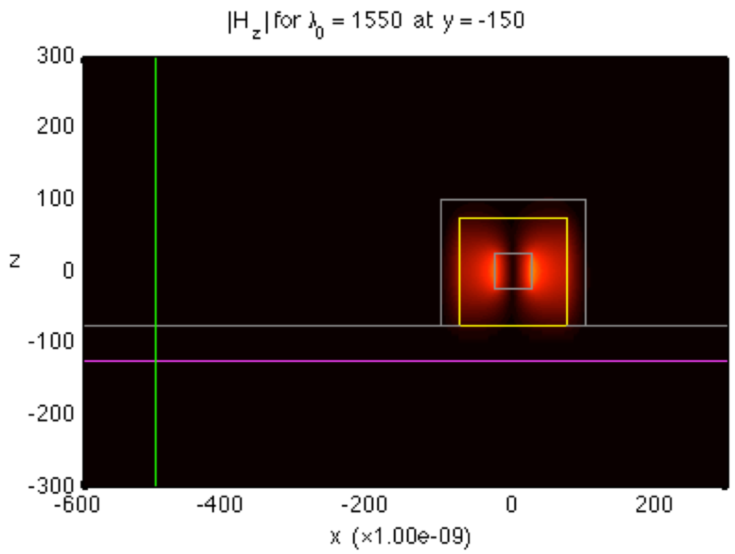
Example 1: lens made of metallic pillars

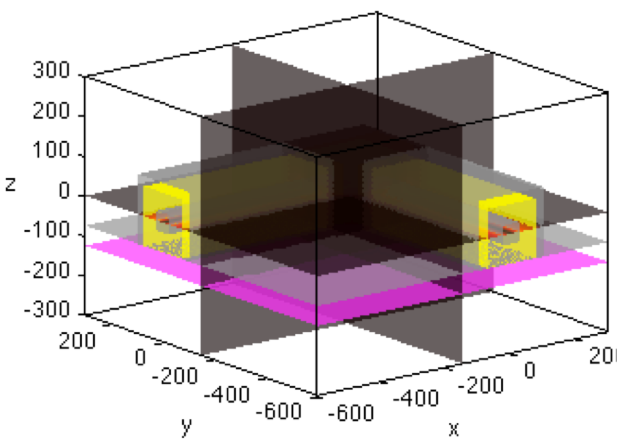


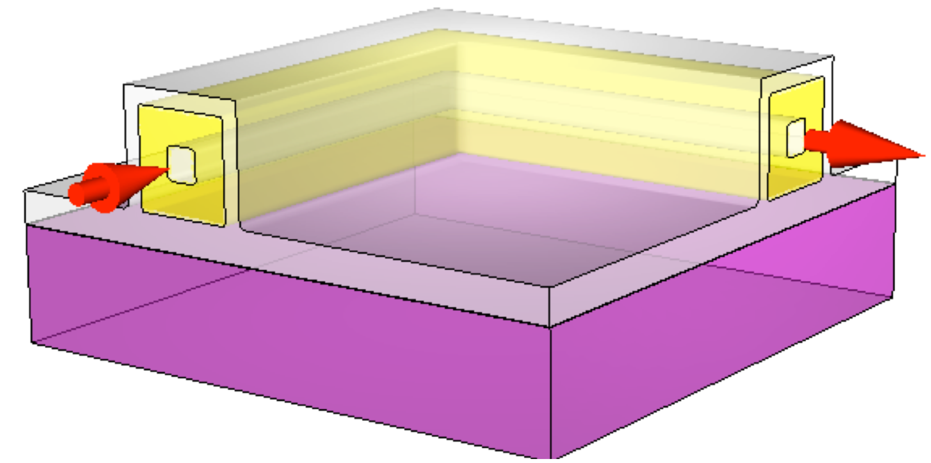


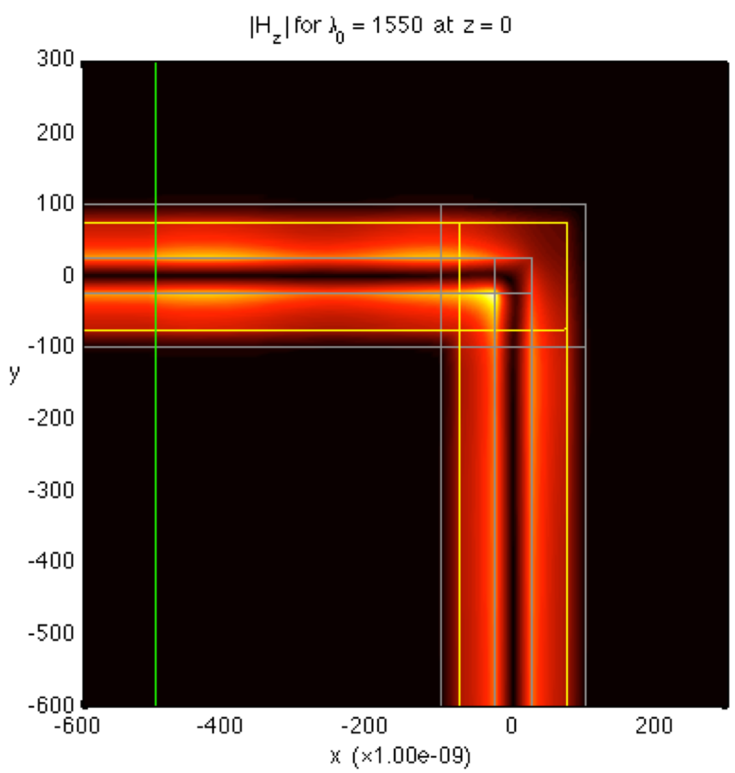
- (wavelength) = 630 nm
- gold: $\varepsilon/\varepsilon_0$ = –10.78 – i 0.79
- Δ = 5 nm
- (# of unknowns) = 20 million

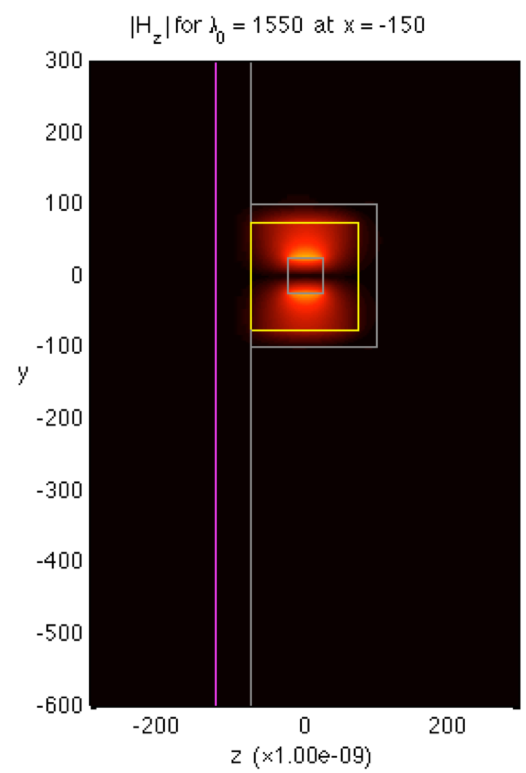
Example 2: 90° bend in metallic coaxial waveguide

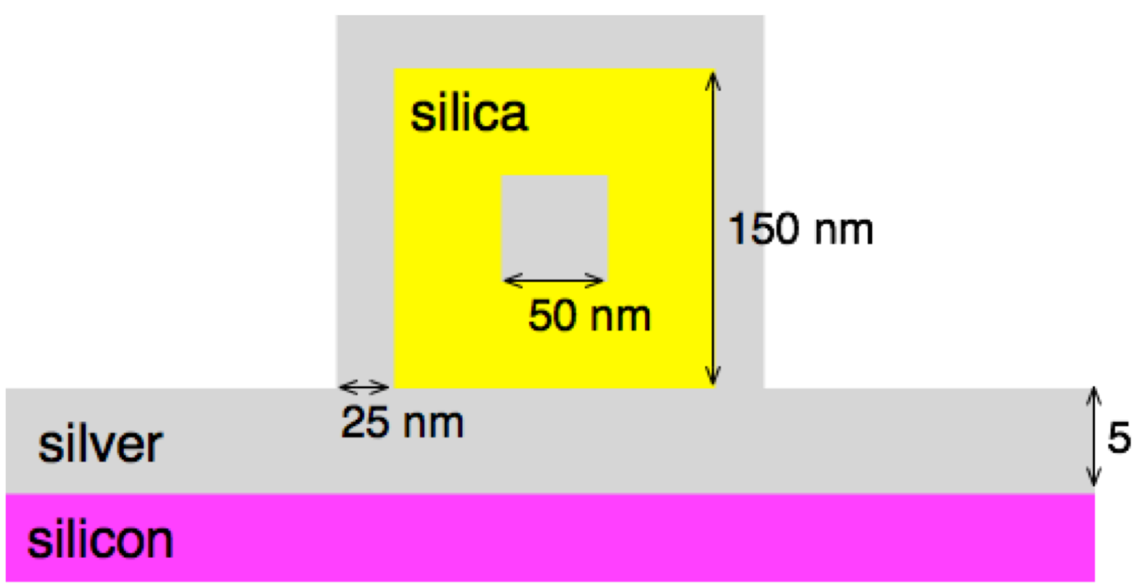




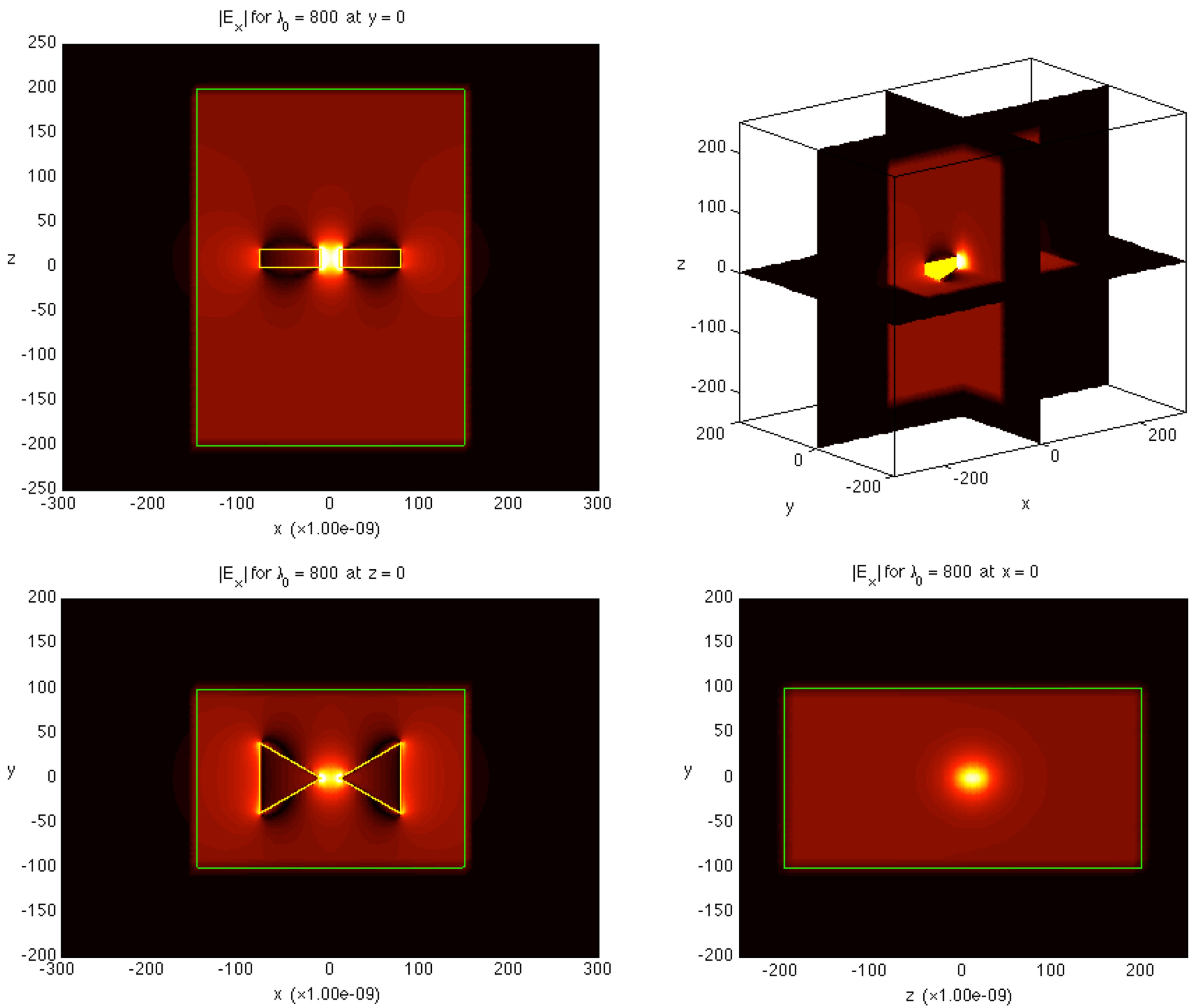








Gold bowtie antenna



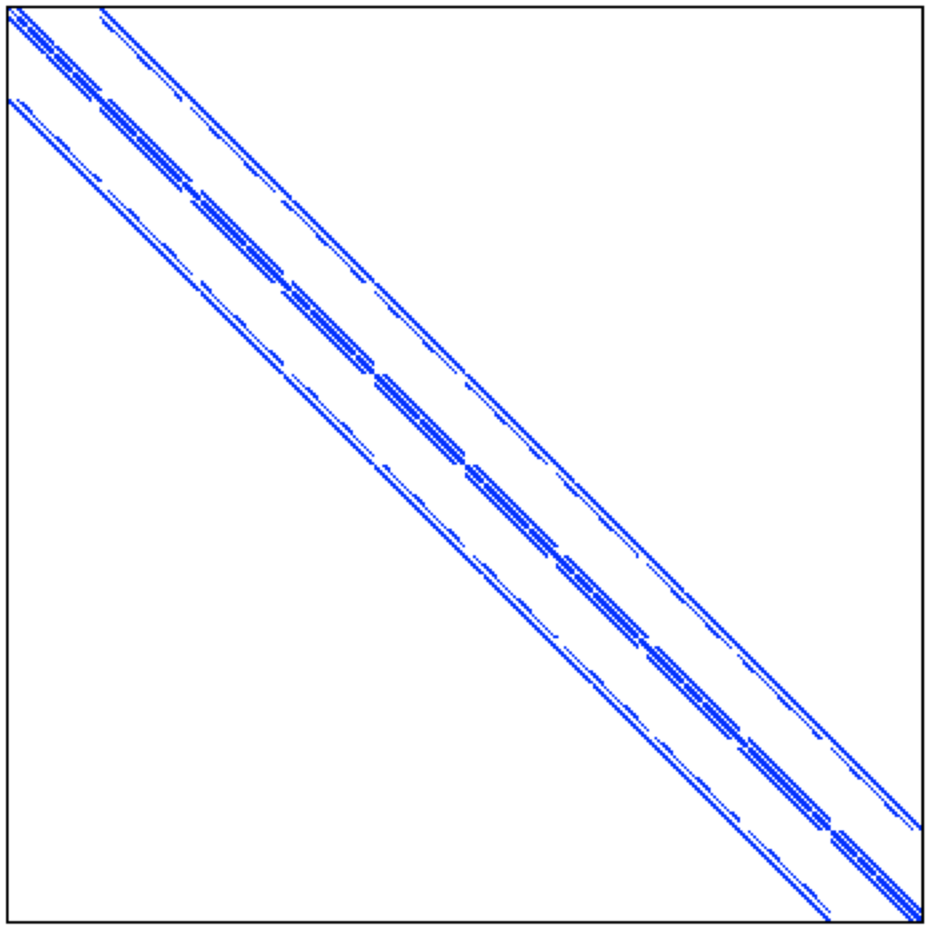
Practical issues in solving $\boldsymbol{A\ x = b}$
(some of my previous research)

There are two kinds of methods to solve $A\,x = b$: direct methods and iterative methods.

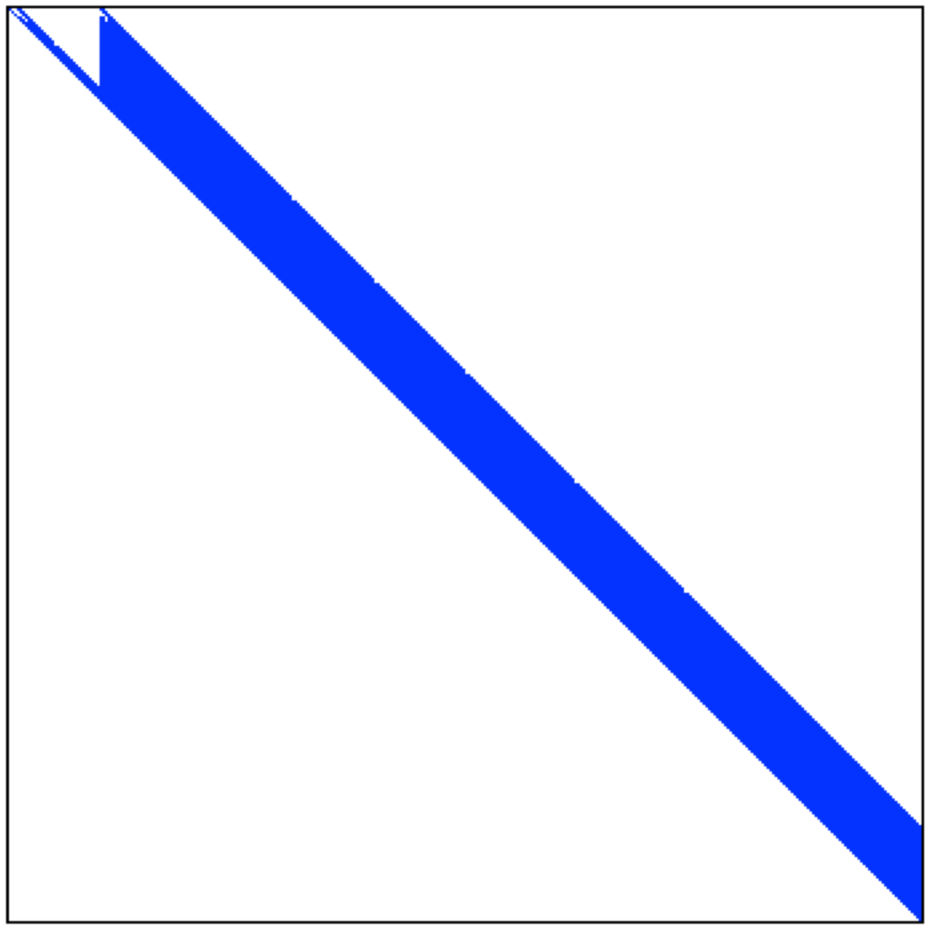
- **Direct methods** ($A = LU \;\Rightarrow\; L\,y = b,\ U\,x = y$)
- **Iterative methods** ($x_0 \to x_1 \to x_2 \to \cdots$)

Direct methods use too much memory for 3D problems.

A



U of $LU = A$



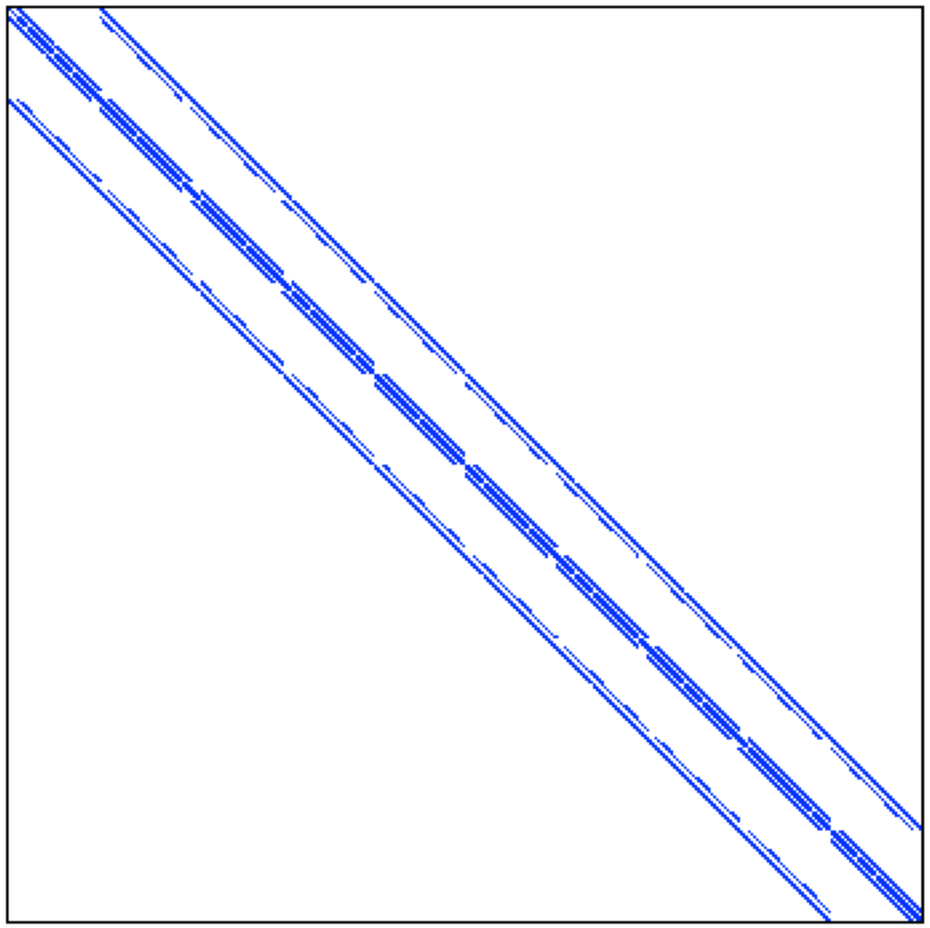
For a 3D grid with $N = 100^3$ grid points

0.6 GB = $O(N)$

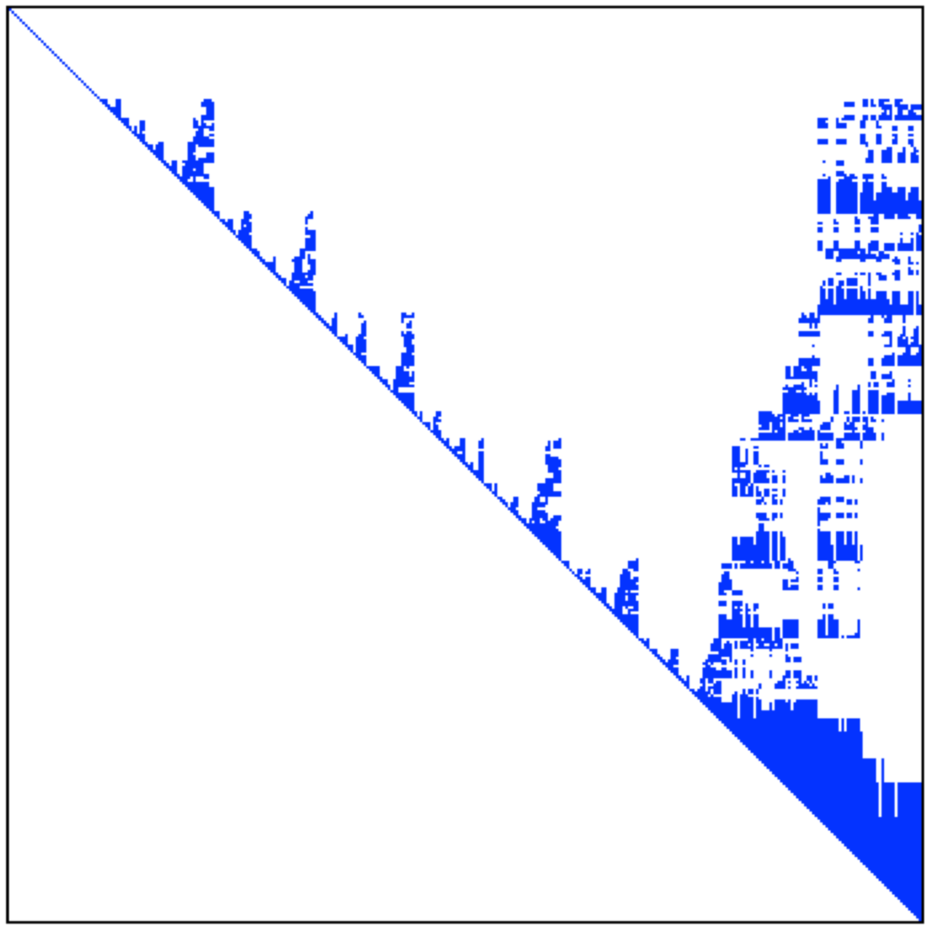
1.5 TB = $O(N^{1.66})$

Direct methods use too much memory for 3D problems.

A



U of $LU = PAQ$



For a 3D grid with $N = 100^3$ grid points

0.6 GB = $O(N)$

10.5 GB = $O(N^{1.33}) < O(N^{1.66})$

Computation of P, Q: $O(N^2)$

Iterative methods: memory-efficient ⇒ suitable for 3D

- **Only matrix stored is sparse A.**

- **x_m is constructed by adding a linear combination of**

$$r_0 \,,\, A\, r_0 \,,\, \ldots \,,\, A^{m-1}\, r_0$$

to x_0.

- **Do not even need A; only need "action of A on vectors".**
  ⇒ Matrix-free formulation.

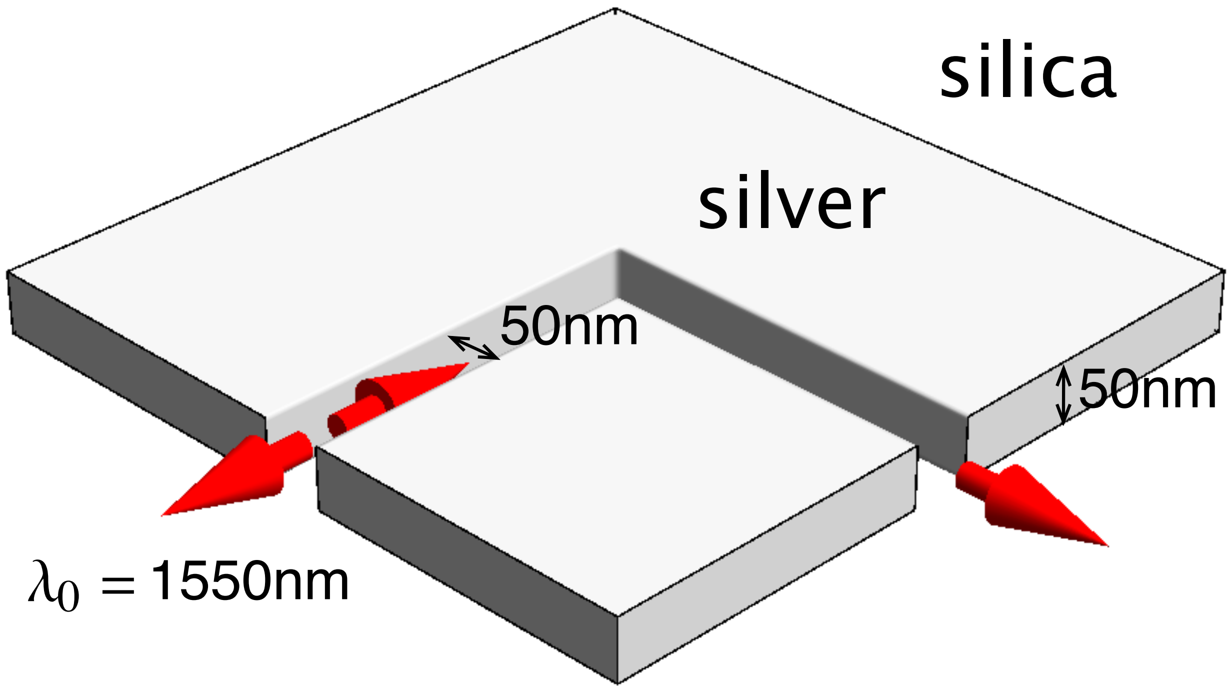
- **Improve solutions until residual vector**

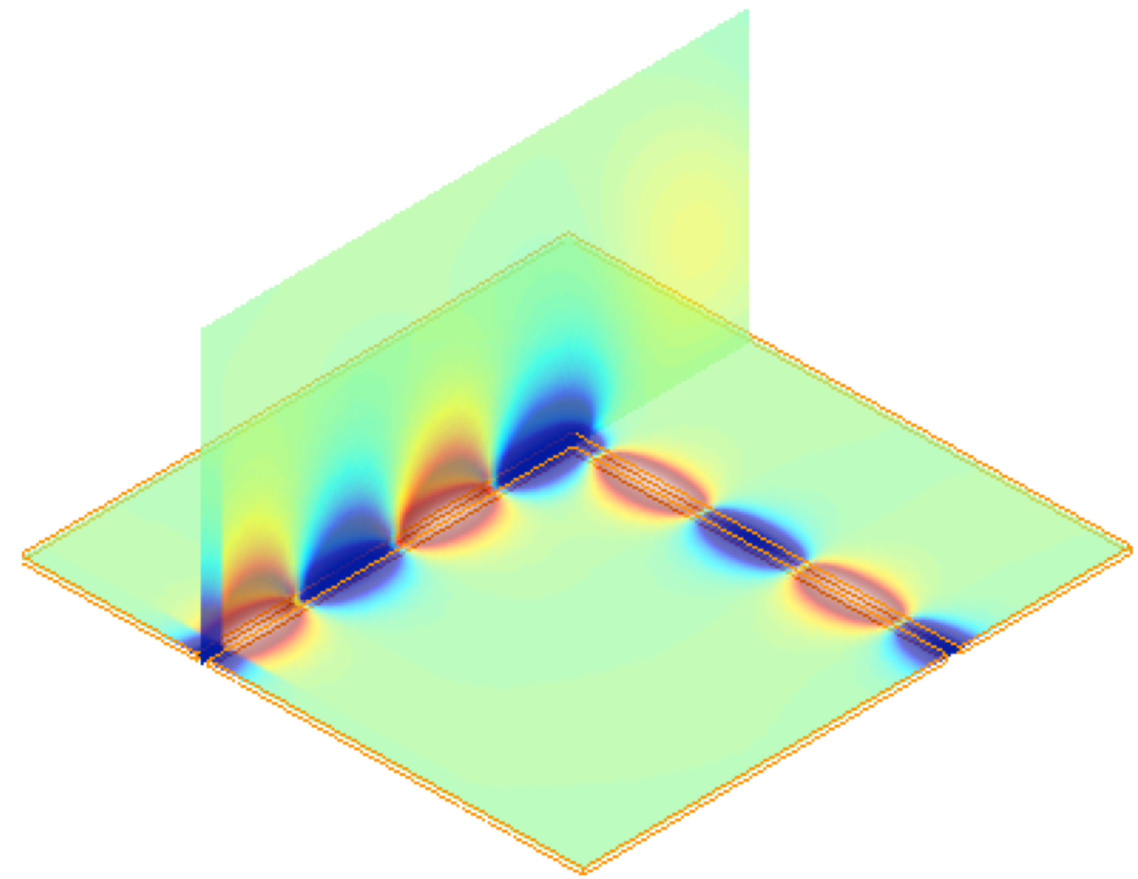
$$r_m = b - A\, x_m$$

  becomes sufficiently small (e.g., $\|r_m\| < 10^{-6}\, \|b\|$).

- **Many iterative methods: BiCG, QMR, GMRES, ...**

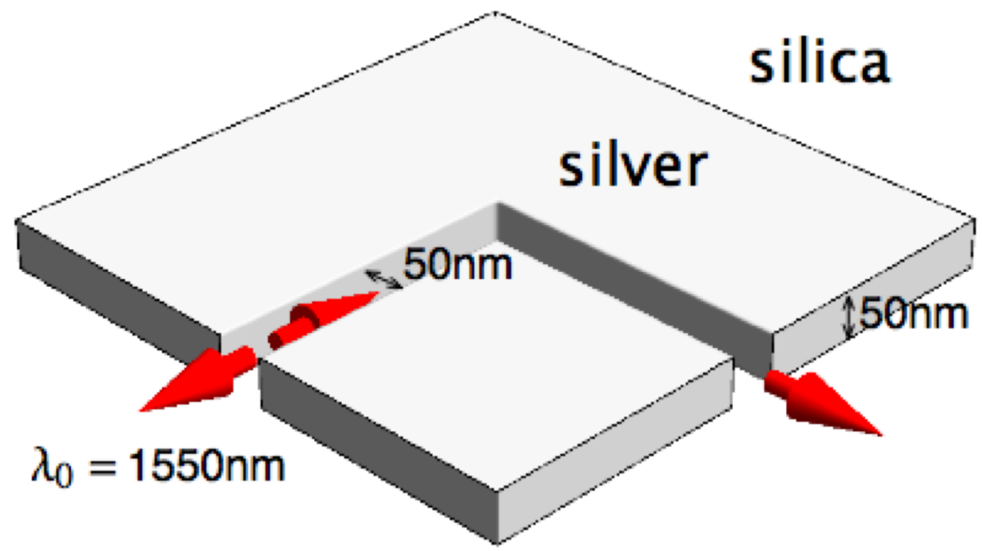
Test problem: 90° bend in metallic slot waveguide



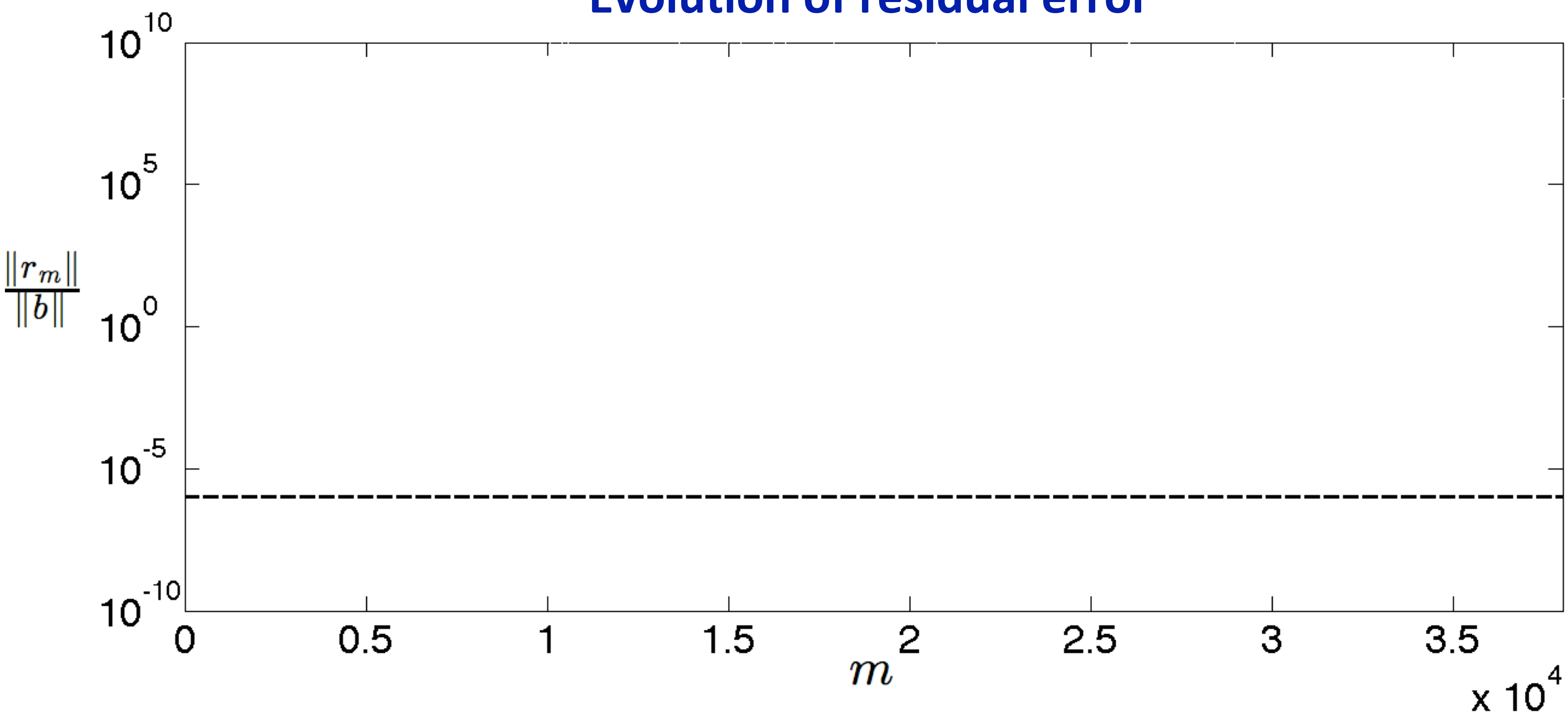


Movie: $$\mathbf{E}(\mathbf{r}, t) = \mathrm{Re}\left\{\mathbf{E}(\mathbf{r})\, e^{i\,\omega\,t}\right\}$$

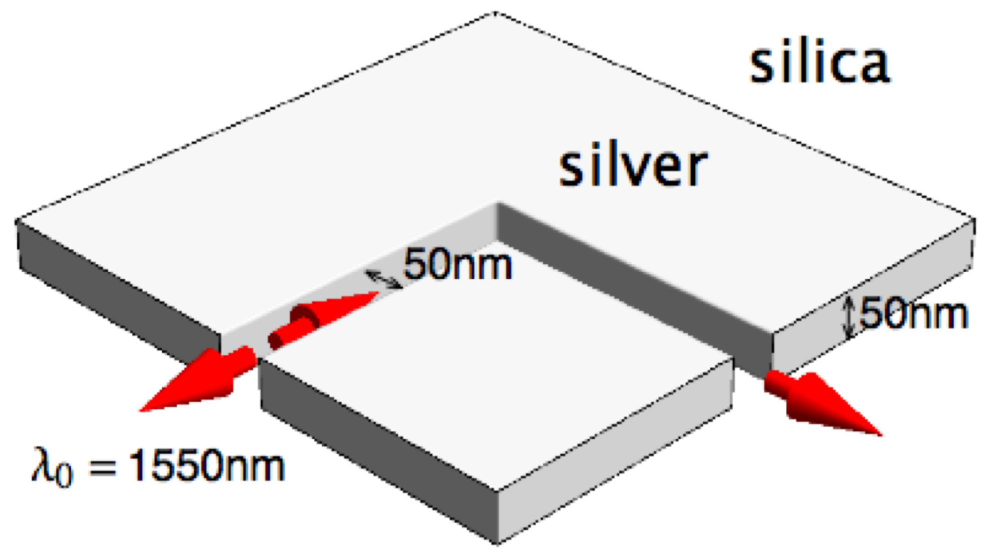
$$N_x \times N_y \times N_z \approx 200 \times 100 \times 200$$
$$N = 3N_xN_yN_z \approx 12 \text{ million}$$



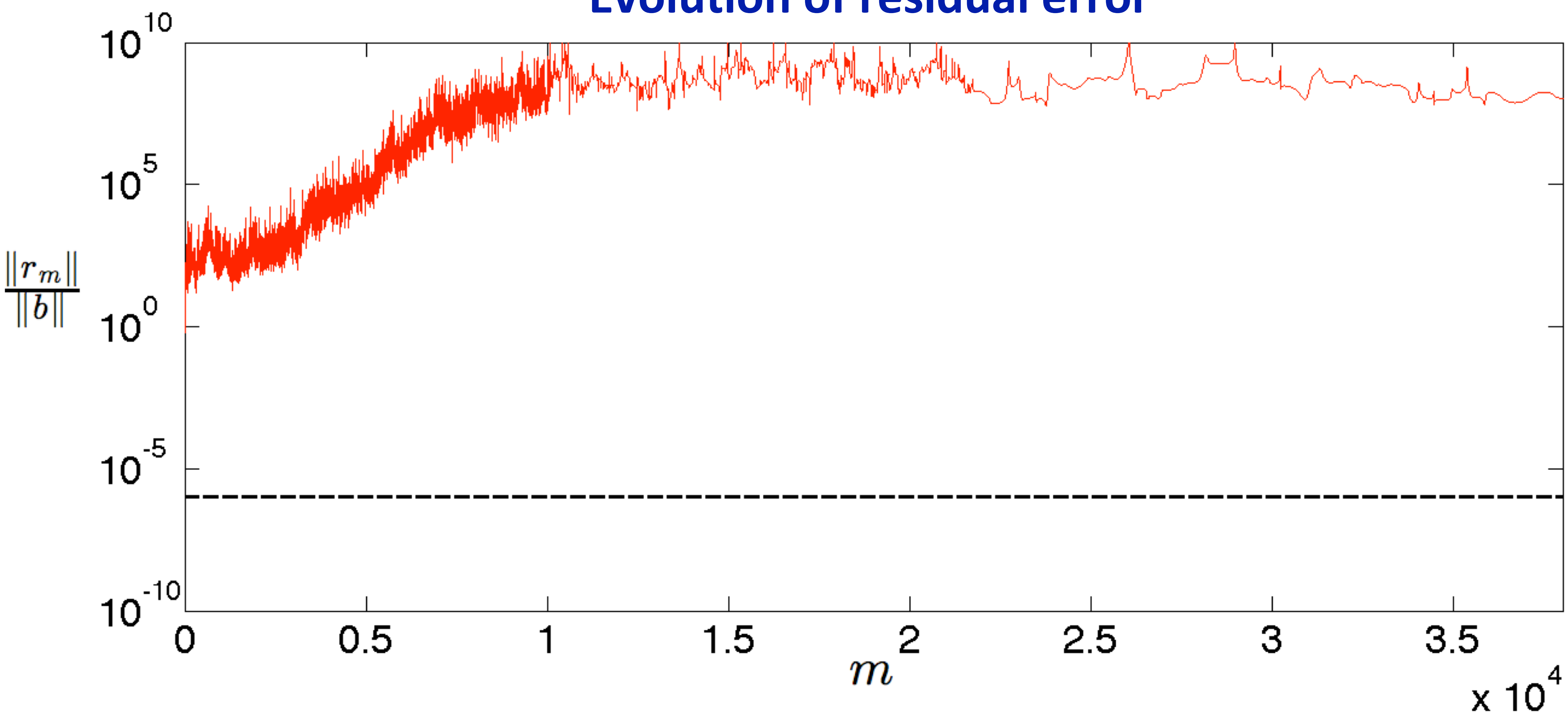
Evolution of residual error



Direct application of BiCG does not work



Evolution of residual error



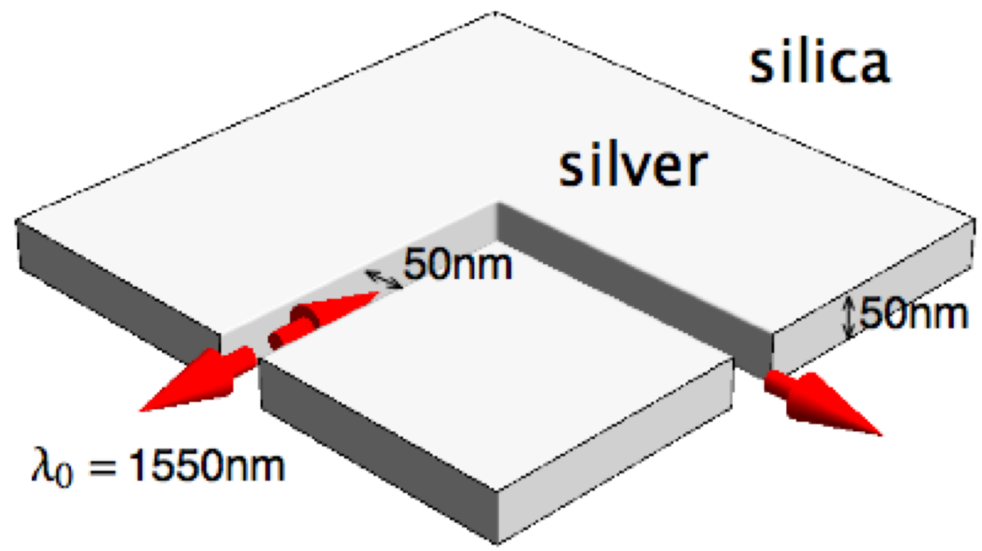
“Preconditioning” accelerates convergence

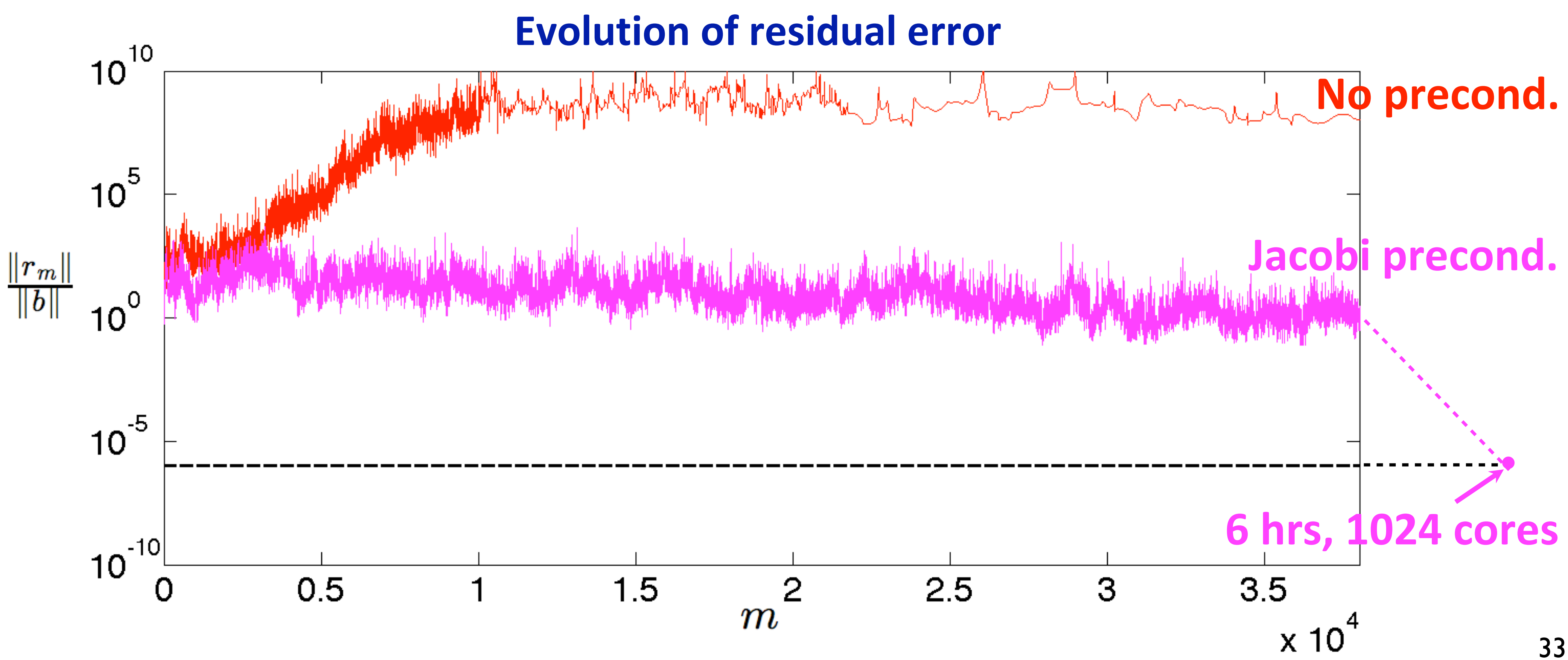
$$A\,x = b \iff \left(P^{-1}\,A\right)x = P^{-1}\,b$$

P **is called a “preconditioner”.**

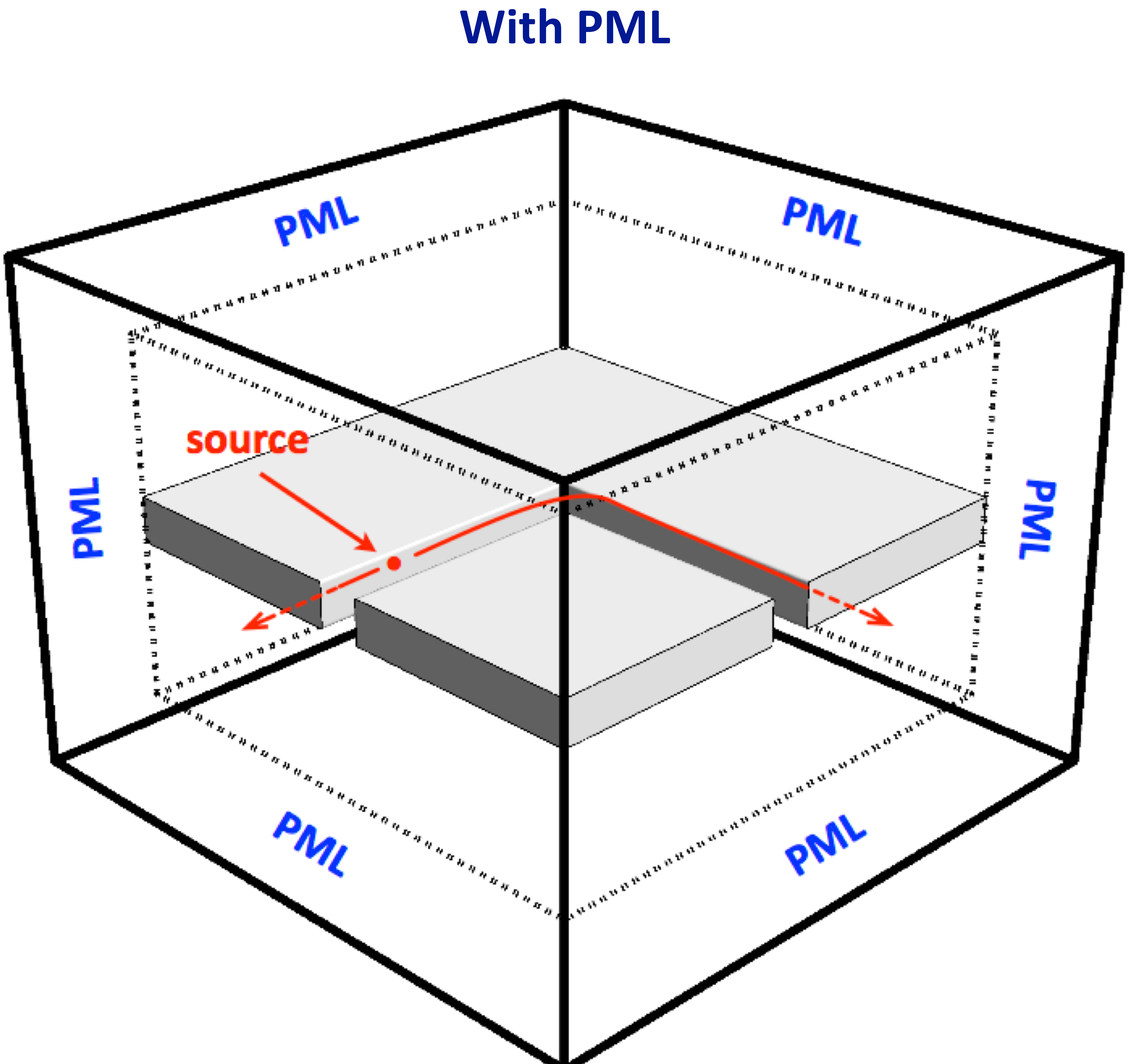
- ***P*** **=** ***A*****: ultimate preconditioner (never used)**
- ***P*** **= diag(*****A*****): Jacobi preconditioner**

Jacobi preconditioner makes convergence faster



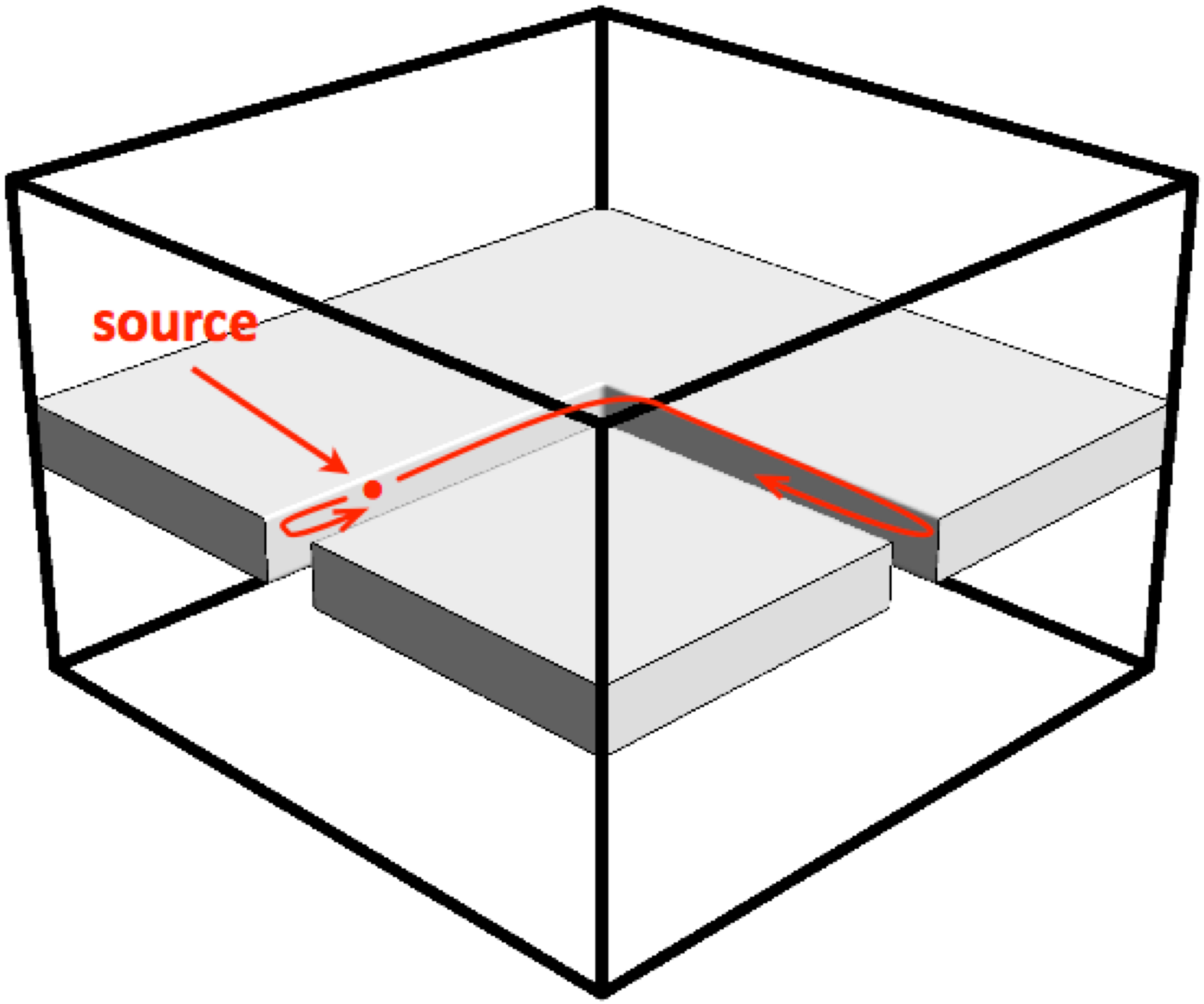


Perfectly matched layer (absorbing boundary cond.)



Perfectly matched layer (absorbing boundary cond.)

Without PML



Two kinds of PML:
uniaxial PML (UPML), stretched-coordinate PML (SC-PML)

Original: $$\nabla \times \mu^{-1} \nabla \times \mathbf{E} - \omega^2 \varepsilon \mathbf{E} = -i \omega \mathbf{J}$$

UPML: $$\nabla \times \boxed{\overline{\overline{\mu}}_s{}^{-1}} \nabla \times \mathbf{E} - \omega^2 \boxed{\overline{\overline{\varepsilon}}_s} \mathbf{E} = -i \omega \mathbf{J} \quad \Longrightarrow A^{\mathrm{u}} x = b$$

$$\overline{\overline{\mu}}_s = \mu \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_z s_x}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix}, \; \overline{\overline{\varepsilon}}_s = \varepsilon \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_z s_x}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix}$$

SC-PML: $$\boxed{\nabla_s} \times \mu^{-1} \boxed{\nabla_s} \times \mathbf{E} - \omega^2 \varepsilon \mathbf{E} = -i \omega \mathbf{J} \quad \Longrightarrow A^{\mathrm{sc}} x = b$$

$$\nabla_s = \hat{\mathbf{x}} \frac{\partial}{s_x \partial x} + \hat{\mathbf{y}} \frac{\partial}{s_y \partial y} + \hat{\mathbf{z}} \frac{\partial}{s_z \partial z}$$

Two kinds of PML: uniaxial PML (UPML), stretched-coordinate PML (SC-PML)

Original: $\nabla \times \mu^{-1} \nabla \times \mathbf{E} - \omega^2 \varepsilon \mathbf{E} = -i \omega \mathbf{J}$

UPML: $\nabla \times \overline{\overline{\mu}}_s^{\,-1} \nabla \times \mathbf{E} - \omega^2 \overline{\overline{\varepsilon}}_s \mathbf{E} = -i \omega \mathbf{J}$

$$\overline{\overline{\mu}}_s = \mu \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_z s_x}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix}, \; \overline{\overline{\varepsilon}}_s = \varepsilon \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_z s_x}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix}$$

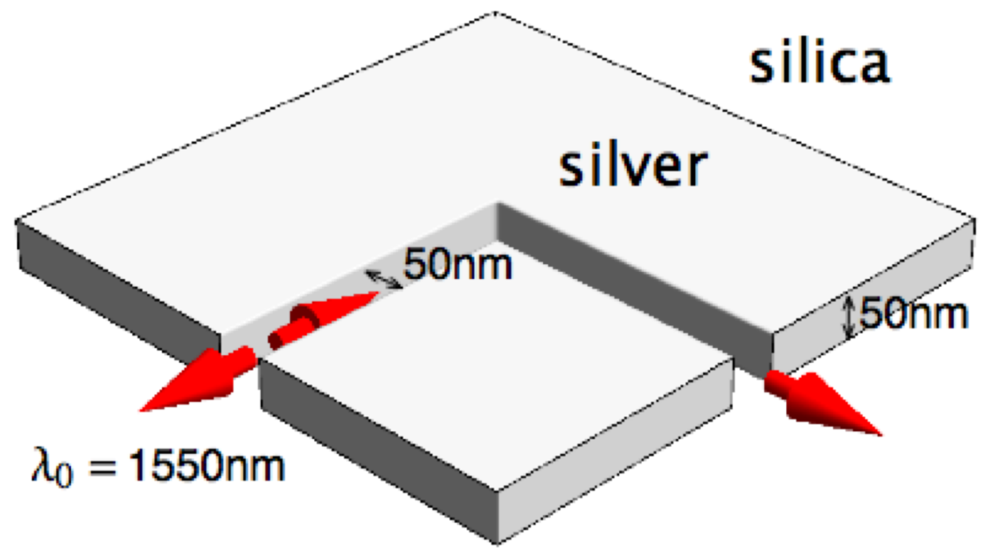
Original eq.
Different materials
⇓
Easy to implement without extra coding

SC-PML: $\nabla_s \times \mu^{-1} \nabla_s \times \mathbf{E} - \omega^2 \varepsilon \mathbf{E} = -i \omega \mathbf{J}$

$$\nabla_s = \hat{\mathbf{x}} \frac{\partial}{s_x \partial x} + \hat{\mathbf{y}} \frac{\partial}{s_y \partial y} + \hat{\mathbf{z}} \frac{\partial}{s_z \partial z}$$

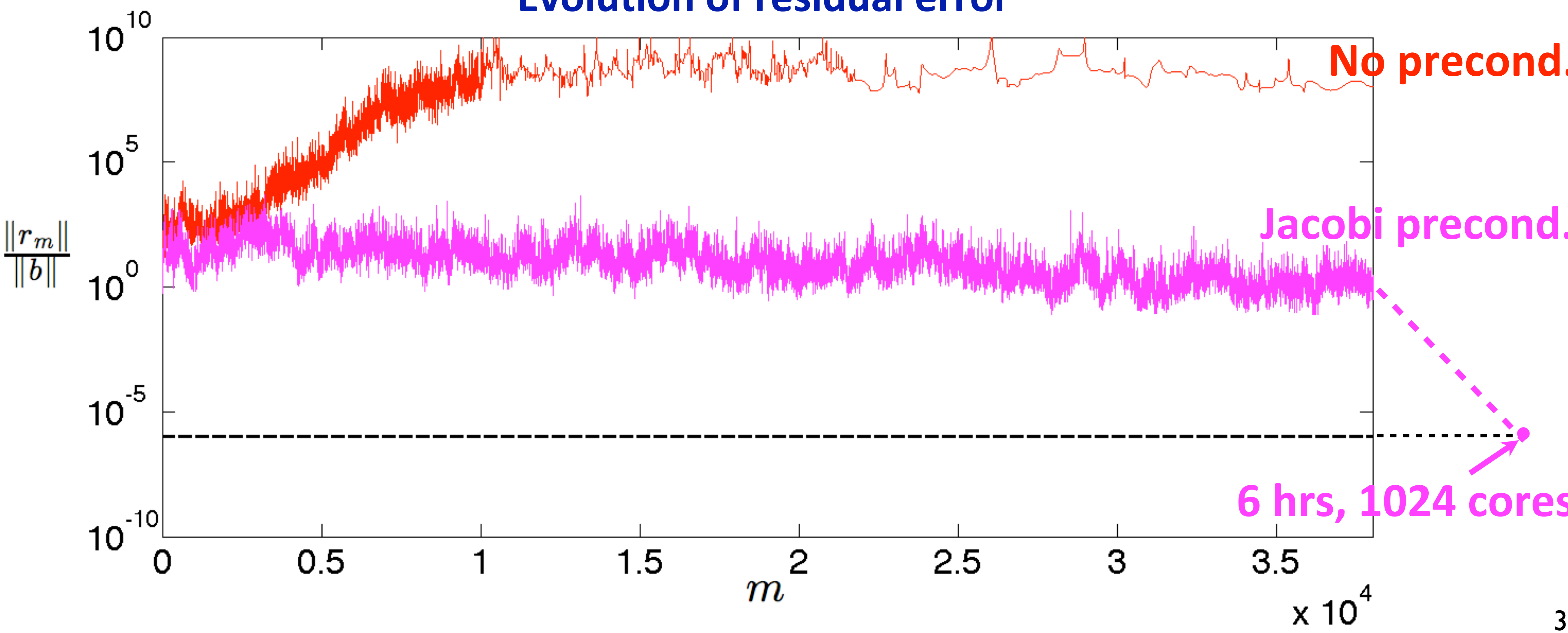
NOT original eq.
⇓
Need extra coding

Jacobi preconditioner makes convergence faster

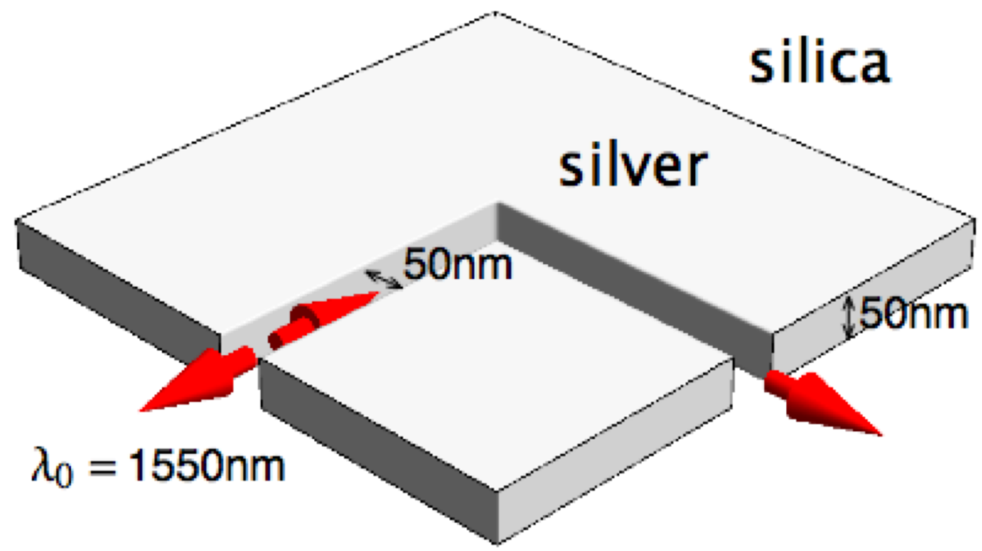


This was with UPML!

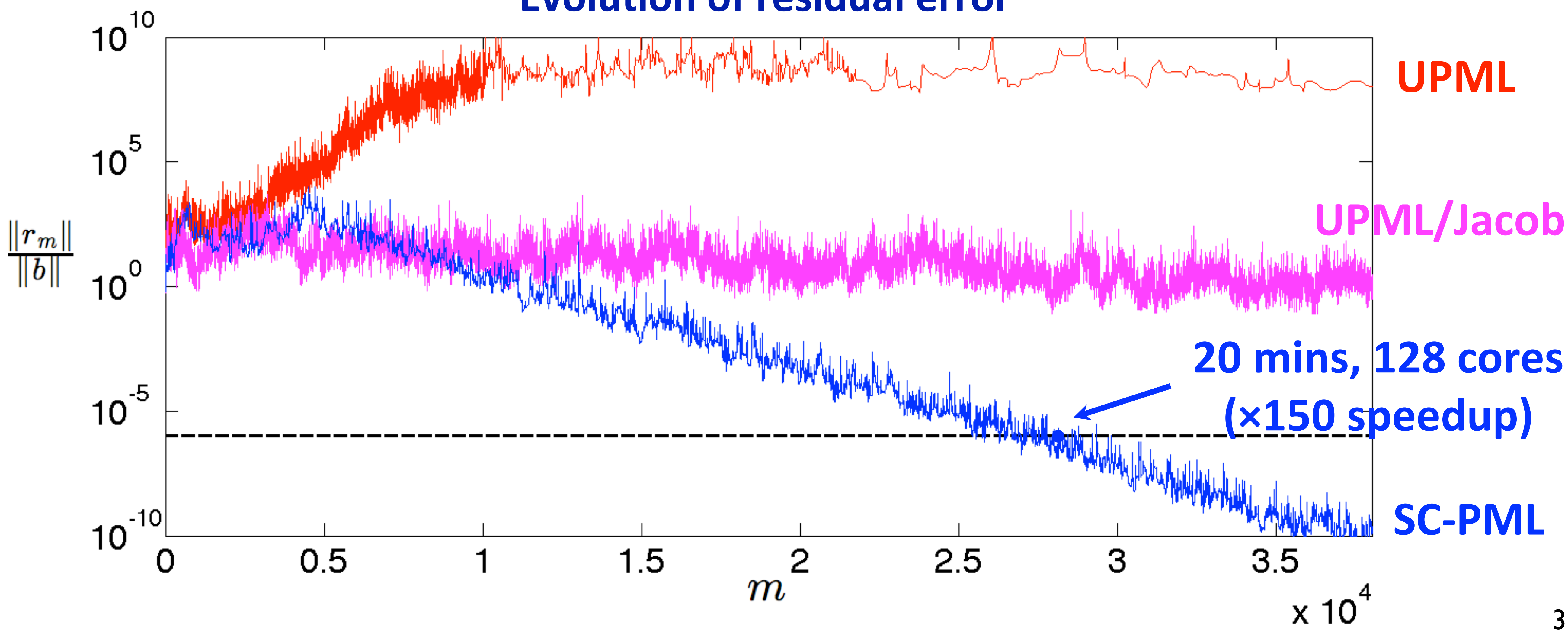
Evolution of residual error



Solution: use SC-PML



Evolution of residual error



Convergence rate depends on *κ*(*A*)

maximum singular value

condition number

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} \geq 1$$

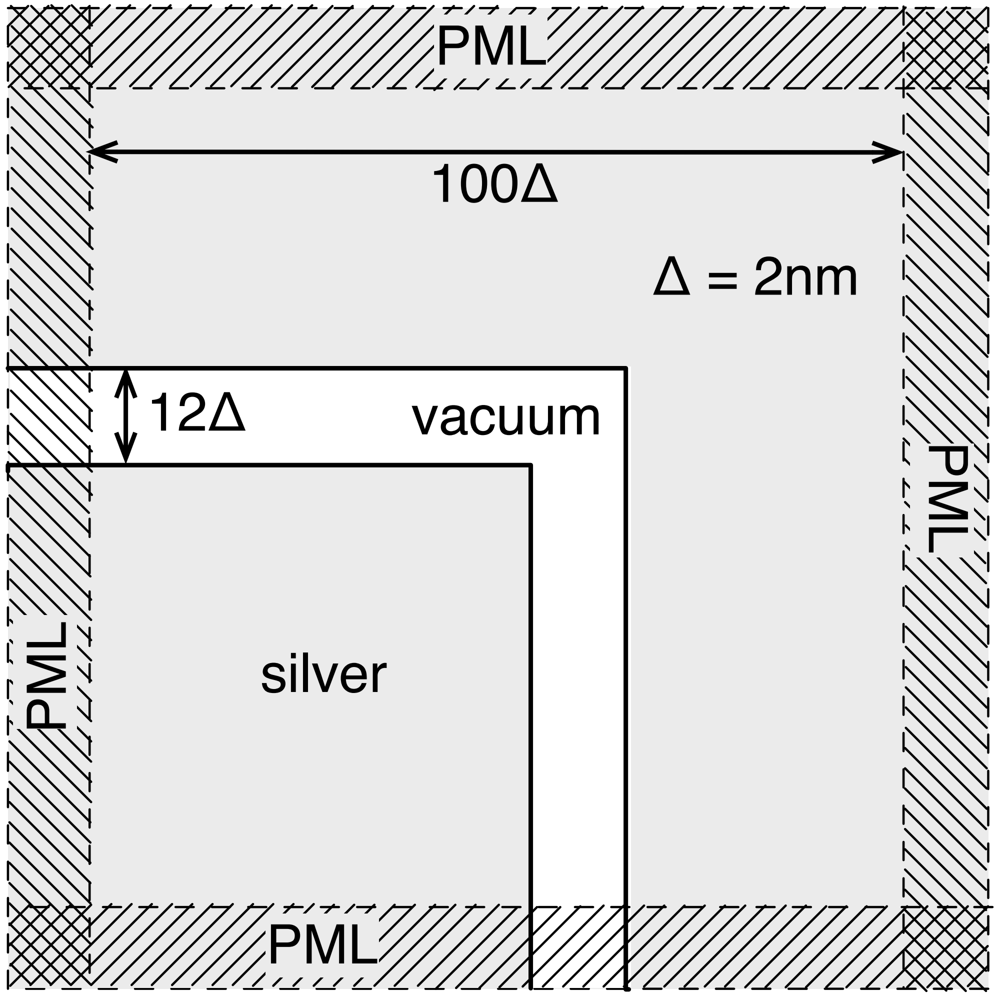
minimum singular value

※ Smaller *κ*(*A*) induces faster convergence.

⇒ $\kappa(A^{sc}) \ll \kappa(A^{u})$?

Yes, $\kappa(A^{sc}) \ll \kappa(A^{u})$!

2D Example



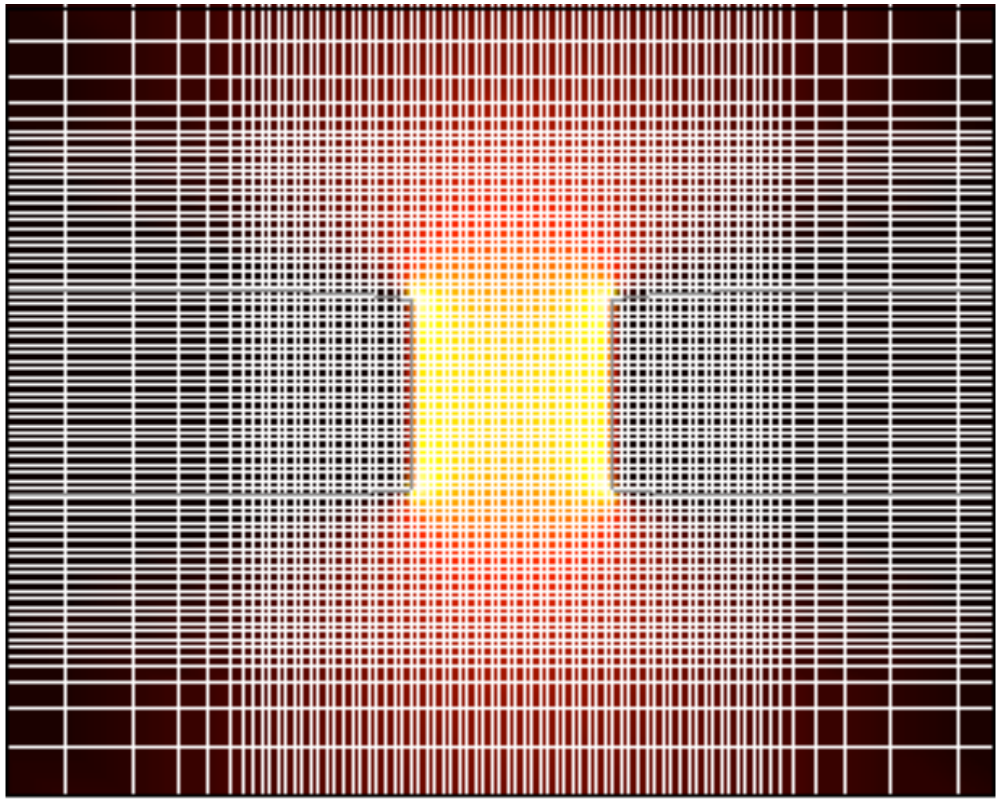
	$\sigma_{max}(A)$	$\sigma_{min}(A)$	$\kappa(A)$
UPML ($A = A^{u}$)	517	2.20×10^{-6}	2.47×10^{8}
SC-PML ($A = A^{sc}$)	2	4.74×10^{-6}	4.22×10^{5}

ratio > 500

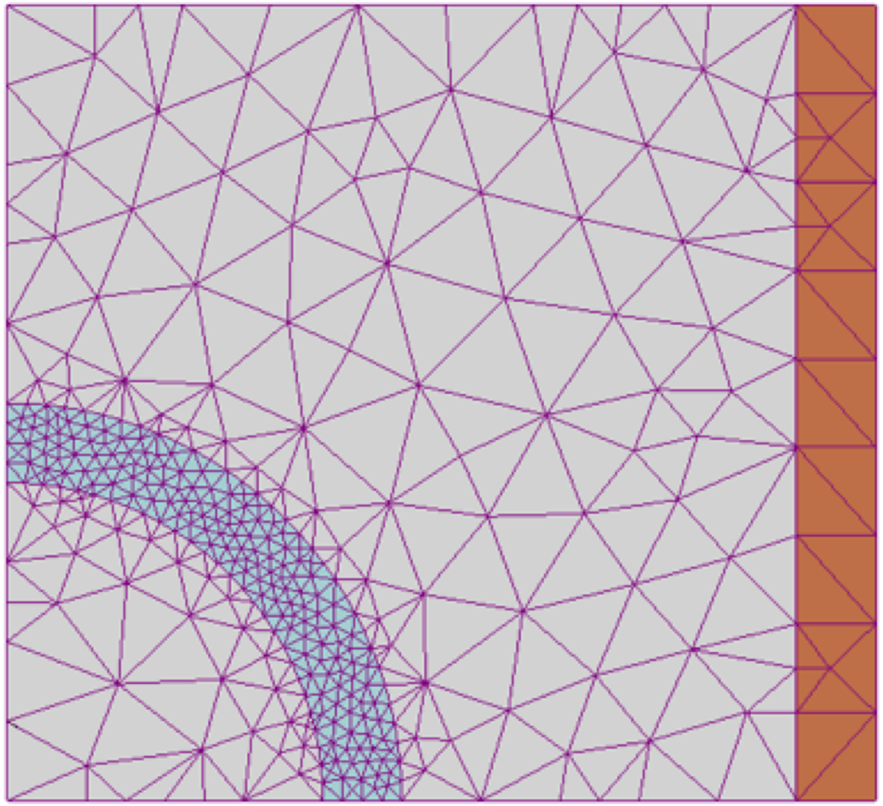
Comparison with FEM

FEM can model curved objects better

FD grid



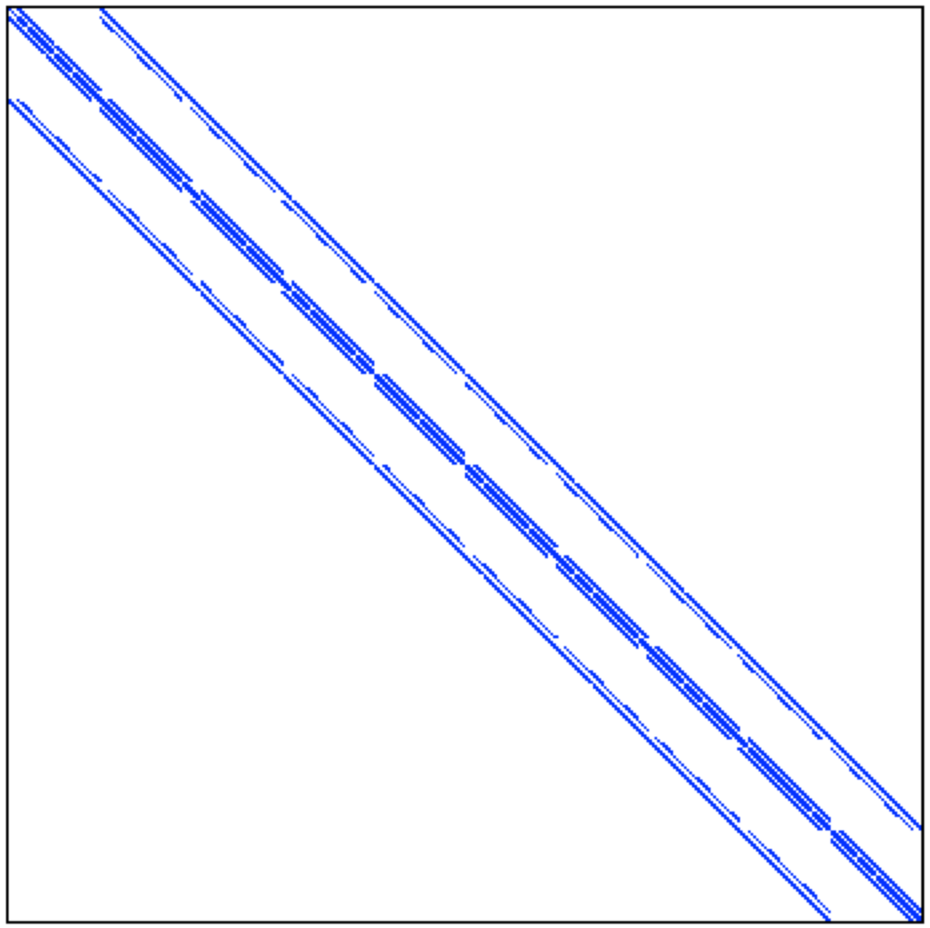
FE mesh



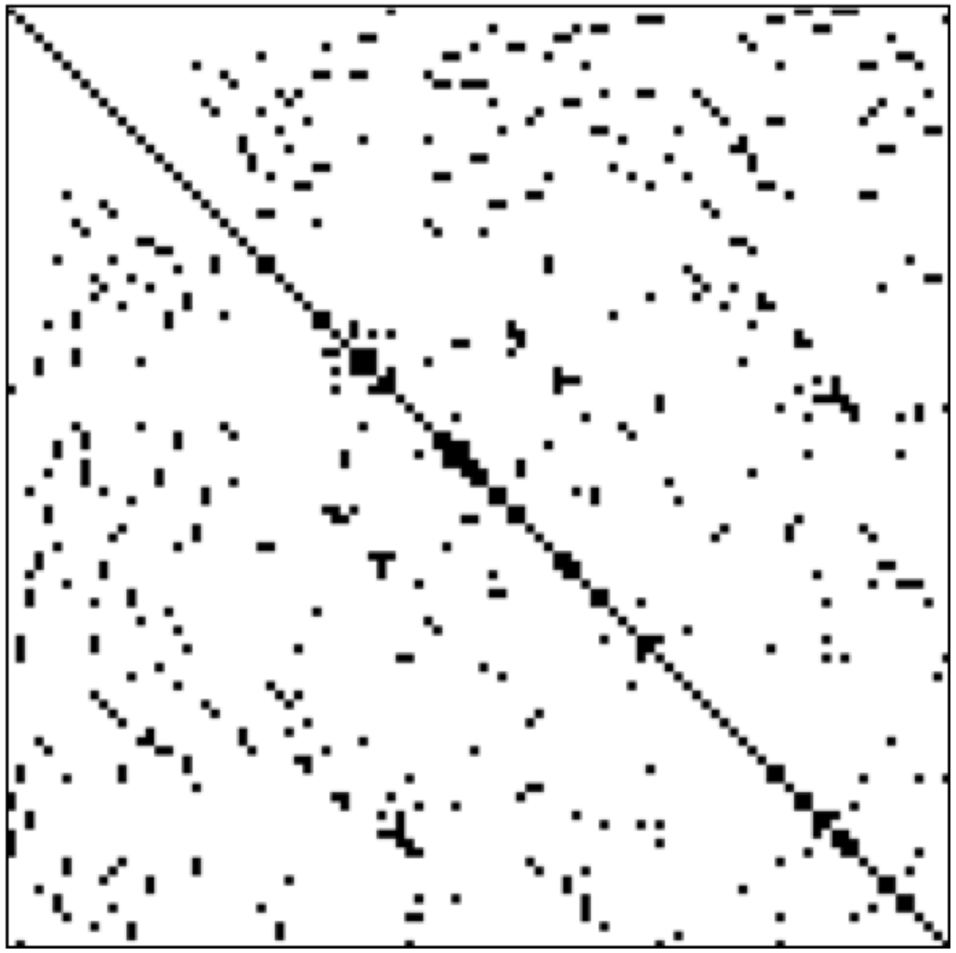
(image from Wikipedia)

… at the penalty of making *A* less structured

***A* on FD grid**



***A* on FE mesh**

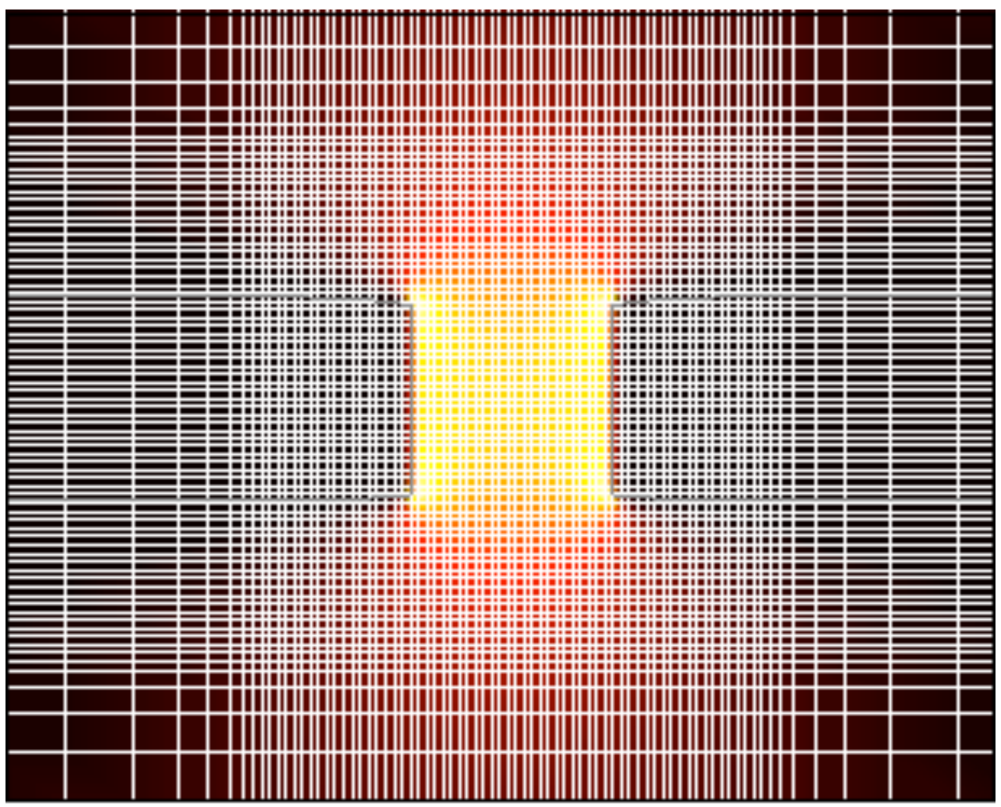


(image from Wikipedia)

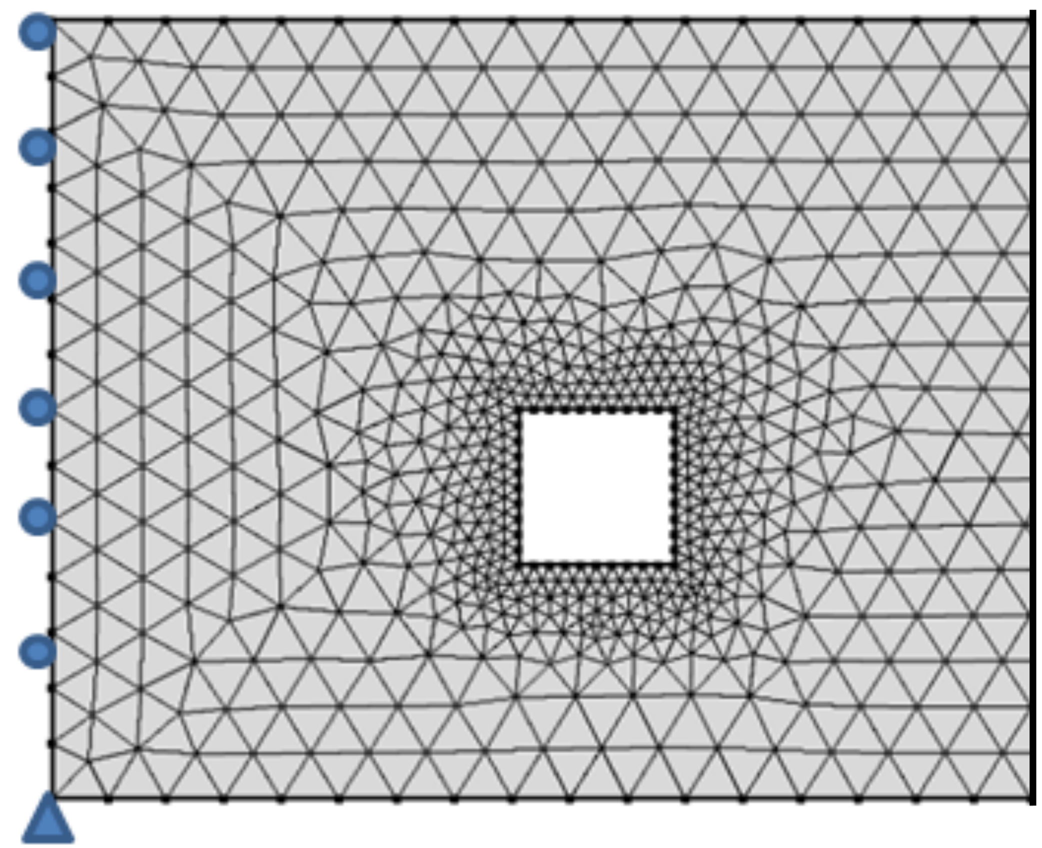
※ Banded *A* is much more efficient to store/apply/factorize.
⇒ FDM is better for large 3D problems?

Still, FEM has much fewer # of unknowns

FD grid



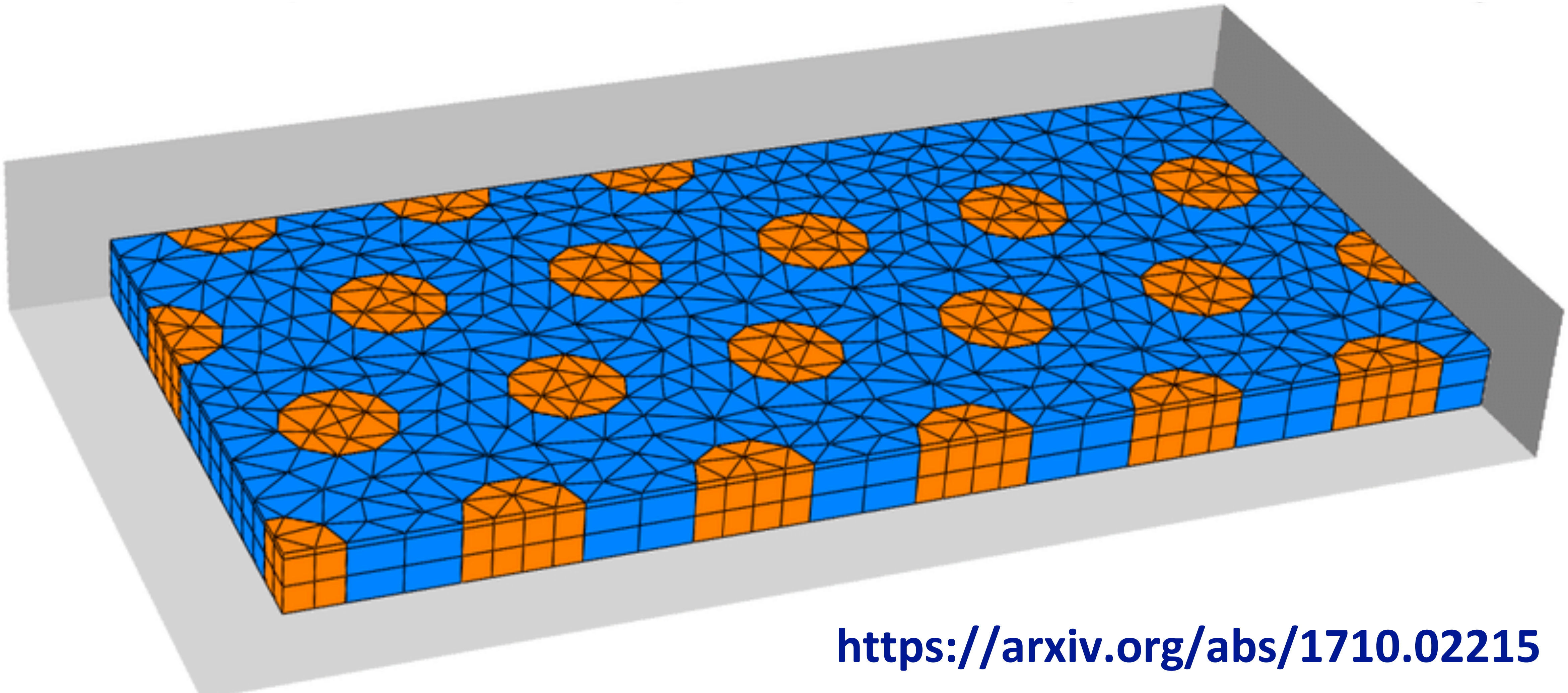
FE mesh



(image from comsol.com)

Even though *A* on FE mesh is unstructured, it is much smaller so more efficient to store/apply/factorize in general.

… but what if scatterers are everywhere?



https://arxiv.org/abs/1710.02215

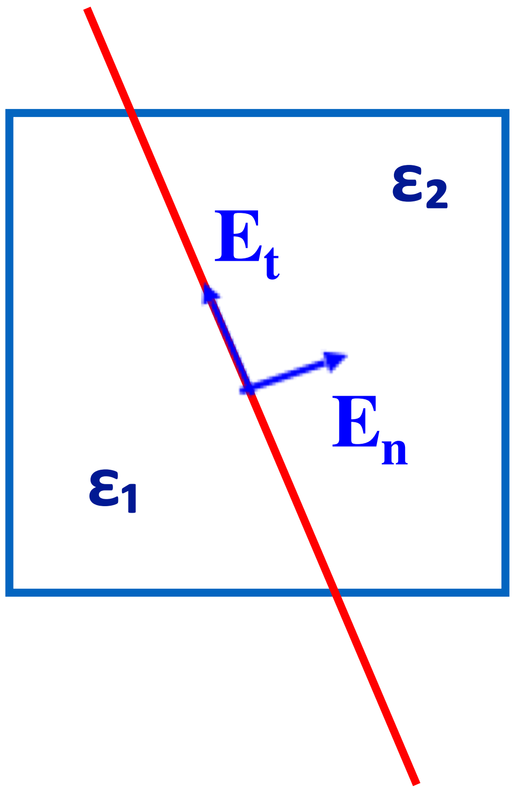
Not much reduction in # of unknowns by using FE mesh!

FDM can also model curved objects!

"Subpixel smoothing"

(Prof. Johnson will discuss this more, if he hasn't):

Assign a single anisotropic ε in a voxel that accurately "averages" ε



(Energy inside voxel)

$$= \frac{1}{2}\left(\mathbf{E}_1 \cdot \mathbf{D}_1\right) V_1 + \frac{1}{2}\left(\mathbf{E}_2 \cdot \mathbf{D}_2\right) V_2$$

$$= \frac{1}{2}\left(\varepsilon_1 E_t^2 + \frac{D_n^2}{\varepsilon_1}\right) V_1 + \frac{1}{2}\left(\varepsilon_2 E_t^2 + \frac{D_n^2}{\varepsilon_2}\right) V_2$$

$$= \frac{1}{2}\left(\frac{V_1\,\varepsilon_1 + V_2\,\varepsilon_2}{V}\right) E_t^2\, V + \frac{1}{2}\left(\frac{V_1 / \varepsilon_1 + V_2 / \varepsilon_2}{V}\right) D_n^2\, V$$

$$\equiv \frac{1}{2}\,\varepsilon_t\, E_t^2\, V + \frac{1}{2}\,\frac{D_n^2}{\varepsilon_n}\, V,$$

- **(Energy inside voxel of two materials)**

= (energy inside voxel of single anisotropic material whose ε is ε_t in *t*-direction and ε_n in *n*-direction)

FDM is much easier to implement than FEM

- Users can easily modify code to add new features (e.g., anisotropy, nonlinearity, new PML)