NeCLO: Neural Convolutional Learning Optimizer for Electromagnetics

Yanxin Zhang [1] **Zaifeng YANG** [1] **Xinyu Yang** [1] **Yueming Lyu** [1]

[1]*Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR)*. Correspondence to: Yueming Lyu Lyu_Yueming@a-star.edu.sg.

1. Introduction

The finite-difference time-domain (FDTD) method is a fundamental numerical framework in computational electromagnetics [1, 2, 3]. However, traditional implementations, characterized by explicit CPU-based execution on structured Yee grids, facing significant scaling challenges as domain sizes increase. Furthermore, the methods scale poorly with domain size [3] due to the Courant–Friedrichs–Lewy (CFL) condition based on the smallest mesh size and lack the differentiability required for modern model-based inverse design [4]. While deep learning surrogates, such as Recurrent CNNs [5, 6] or Physics-Informed Neural Networks [7, 8, 9, 10, 11], and Neural Operator Surrogates [12, 13, 14, 15, 16, 17, 18, 19], promise acceleration, they often compromise physical exactness or fail to generalize across varying boundary conditions. Recently, Graph Neural Networks (GNNs), exemplified by the GEM framework [20], have achieved significant acceleration over CPU-based solvers by exploiting GPU parallelism. However, on regular structured grids, GNNs incur **unnecessary structural complexity**: they rely on explicit edge storage and gather-scatter operations, which break memory contiguity and limit computational throughput compared to dense tensor operations.

In this paper, we propose NeCLO, a high-throughput differentiable solver framework. We rigorously benchmark three architectures: (1) Graph-based (GEM) baselines [21]; and two novel architectures introduced in this work: (2) the Convolution-based NeCLO; and (3) a Pure Tensor Slicing variant. We demonstrate that NeCLO achieves numerical fidelity indistinguishable from standard FDTD. Furthermore, by exploiting memory contiguity, both our proposed methods significantly outperform GNNs in throughput while retaining full differentiability for inverse optimization.

2. Methodology

In this section, we analyze the evolution of differentiable solvers by comparing three distinct architectures: Graph-based (GEM), Convolutional (NeCLO), and the Pure Tensor Slicing method.

2.1 Graph-Based Approach (GEM)

We adopt the architecture by Bakirtzis et al. [20] as our geometric baseline. GEM maps the discrete Yee grid to a directed graph and formulates the electromagnetic field update as a Message Passing Neural Network (MPNN) operation. While flexible for irregular meshes, this approach incurs significant memory overhead on regular grids due to non-contiguous access patterns.

2.2 Convolutional Approaches (NeCLO)

To align with the hardware optimization of GPUs, we treat the simulation domain as a 3D voxel grid and approximate spatial derivatives using Convolutional Neural Networks (CNNs).

Convolutional Approaches (NeCLO). To exploit GPU hardware optimizations, we treat the simulation domain as a 3D voxel grid. We evaluate two variants:

- **NeCLO v1 (Small Kernels):** Utilizes standard finite-difference kernels to strictly mimic Maxwell's update equations.
- **NeCLO v2 (Unified Large Kernels):** Compresses the iteration into monolithic `Conv3d` operations to maximize throughput, addressing the Yee grid's staggered nature through specific padding strategies.

Detailed architectural specifications and kernel configurations are provided in AppendixA

2.3 Differentiable Parameter Inversion

We extend NeCLO beyond forward simulation to enable Differentiable Parameter Inversion. By unrolling the iterative FDTD updates into a computation graph, we facilitate exact gradient propagation through time via Automatic Differentiation (AD). This formulation allows for the direct optimization of constitutive parameters, such as conductivity σ, by minimizing the discrepancy between simulated and observed fields. Unlike black-box surrogates, this physics-constrained learning ensures that retrieved parameters satisfy Maxwell's equations, enabling accurate inverse design from sparse time-domain measurements without the manual derivation of adjoint state equations.

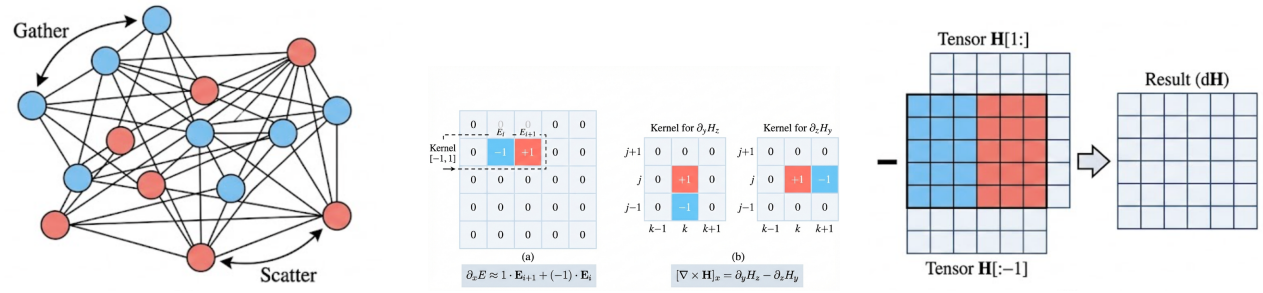
3. Experiments & Results

To validate numerical fidelity, we conducted a 3D resonant cavity simulation on a $20 \times 20 \times 20$ Yee grid over 500 time steps. The simulation parameters were set with grid spacing $\Delta = 2.0$ mm and time step $\Delta t = 3.81$ ps .

3.1 Numerical Accuracy

Table 1: Numerical Accuracy & Status Comparison

Method	Abs. Error	Status	Interpretation
GEM (GNN)	9.0×10^{-15}	Float Limit	Num. Equiv. (Float noise)
NeCLO (Ours)	**0.0**	**Bit-Exact**	**Exact Replica** (Binary Match)



(a) Graph-based (GEM) (b) Convolutional (NeCLO) (c) Pure Tensor Slicing

Fig. 1: Comparison of differentiable solver architectures. (a) **GEM** incur overhead from irregular scatter-gather memory access. (b) Our **NeCLO** utilizes optimized kernels but requires padding and boundary masking. (c) Our **Pure Tensor Slicing** performs global vectorized subtraction, eliminating padding artifacts while maintaining bit-exactness.

Table 2: Numerical Integrity Check in Lossy Media

σ (S/m)	Loss Regime	Max Abs. Error	Limiting Factor
4.0	High	1.17×10^{-10}	Signal Decay (Near Zero)
3.0	Mod.	7.24×10^{-7}	Float32 Resolution
2.0	Mod.	1.32×10^{-6}	Float32 Resolution
1.0	Low	1.42×10^{-6}	Float32 Accumulation

Note: Error bounded by float32 precision limits.

Result Analysis: GEM achieved a maximum absolute error of 9.03×10^{-15} , while both the NeCLO and the Pure Tensor Slicing yield an absolute error of exactly 0.0 in *Float32* precision. This confirms that our tensor formulation is *bit-exact* and mathematically equivalent to the reference finite-difference loops , eliminating any approximation errors introduced by neural network surrogates.

We further validated that NeCLO remains highly effective in a range of lossy materials by sweeping the conductivity σ from 1.0 to 4.0 S/m. As shown in Table 2, the proposed method achieves consistent numerical fidelity with a negligible maximum absolute error ($< 1.5\times10^{-6}$) in all cases, confirming its reliability for complex medium simulations.

3.2 Computational Performance

We benchmarked the computational efficiency on a simulation of a lossy medium ($\sigma > 0$). To rigorously evaluate algorithmic superiority independent of hardware acceleration, all methods were executed on a CPU environment.Table A1 summarizes the results. The traditional FDTD implementation, burdened by explicit loops, requires 37.0 ms per time step. The NeCLO-Conv3D approach reduces this to 0.448 ms ($\sim$ 82$\times$ speedup) by vectorizing the operations. Most notably, Pure Tensor Slicing method achieves a latency of only 0.09 ms per step, delivering a massive $\sim$ 411$\times$ speedup. While these results are CPU-based, deploying the proposed solver on GPUs is expected to yield further orders-of-magnitude improvements in throughput.

3.3 Generalization & Learning

Although the Pure Tensor Slicing approach offers high throughput and precision, it remains inherently memory-bound, strictly constrained by GPU memory bandwidth. Unlike NeCLO, it cannot leverage the hardware acceleration provided by Tensor Cores. Furthermore, during backpropagation, the framework must traverse a massive and fragmented computation graph generated by slicing, resulting in significant Autograd overhead. Consequently, we exclusively employ the NeCLO for the Differentiable Parameter Inversion to maximize computational efficiency. The results demonstrate that the learned conductivity parameter σ converges to the ground truth with an error magnitude of 10^{-5}, indicating superior accuracy (refer to Appendix C Figure A5).

4. Future Work

Future iterations will incorporate a learnable material density mask $m(\mathbf{x})$ and a dispersion operator f_{disp} directly into the update loop, enabling the inverse design of complex nanophotonic devices. A detailed mathematical formulation of this proposed differentiable framework, including the topology update equations and material interpolation schemes, is provided in Appendix D.

5. Conclusion

In this paper, we introduced NeCLO, a high performance differentiable solver for Maxwell's equations. Through rigorous benchmarking, we demonstrated that our NeCLO along with Tensor Slicing variant significantly outperforms GEM in both throughput and numerical fidelity. NeCLO achieves bit-exact equivalence to standard FDTD while fully leveraging deep learning hardware acceleration. Furthermore, its successful application in inverse parameter retrieval validates NeCLO as a robust foundation for next-generation nanophotonic optimization.

Acknowledgments

This work was supported by the Physics Foundation Model initiative at the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR). Yueming Lyu is supported by Career Development Fund (CDF) of the Agency for Science, Technology and Research (A*STAR) (No: C243512014)

References

[1] A. Taflove, S. C. Hagness, and M. Piket-May. Computational electromagnetics: The finite-difference time-domain method. In *The Electrical Engineering Handbook*, volume 3, pages 629–670. CRC Press, 2005.

[2] Dennis M Sullivan. *Electromagnetic simulation using the FDTD method*. John Wiley & Sons, 2013.

[3] F. Teixeira, C. Sarris, Y. Zhang, et al. Finite-difference time-domain methods. *Nat. Rev. Methods Primers*, 3(1):75, 2023.

[4] T. Shan, J. Zeng, X. Song, et al. Physics-informed supervised residual learning for electromagnetic modeling. *IEEE Trans. Antennas Propag.*, 71(4):3393–3407, 2023.

[5] L. Guo, M. Li, S. Xu, and F. Yang. Study on a recurrent convolutional neural network based FDTD method. In *Proc. Int. Appl. Comput. Electromagn. Soc. Symp.-China (ACES)*, volume 1, pages 1–2, August 2019.

[6] L. Guo, M. Li, S. Xu, F. Yang, and L. Liu. Electromagnetic modeling using an FDTD-equivalent recurrent convolution neural network: Accurate computing on a deep learning framework. *IEEE Antennas Propag. Mag.*, 2021.

[7] D. Zhu, Q. Zhao, Y. Bo, W. Chen, and L. Yang. Application of deep learning in FDTD method. In *Proc. 13th Int. Symp. Antennas, Propag. EM Theor*, pages 1–3, 2021.

[8] Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

[9] Gal G Shaviner, Hemanth Chandravamsi, Shimon Pisnoy, Ziv Chen, and Steven H Frankel. Pinns for solving unsteady maxwell's equations: convergence issues and comparative assessment with compact schemes. *Neural Computing and Applications*, 37(29):24103–24122, 2025.

[10] Joowon Lim and Demetri Psaltis. Maxwellnet: Physics-driven deep neural network training based on maxwell's equations. *Apl Photonics*, 7(1), 2022.

[11] Kuang Luo, Jingshang Zhao, Yingping Wang, Jiayao Li, Junjie Wen, Jiong Liang, Henry Soekmadji, and Shaolin Liao. Physics-informed neural networks for pde problems: a comprehensive review. *Artificial Intelligence Review*, 58(10):323, 2025.

[12] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

[13] Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

[14] Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for pdes on arbitrary domains. *arXiv preprint arXiv:2505.18781*, 2025.

[15] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[16] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[17] Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

[18] Roberto Molinaro, Samuel Lanthaler, Bogdan Raonić, Tobias Rohner, Victor Armegioiu, Stephan Simonis, Dana Grund, Yannick Ramic, Zhong Yi Wan, Fei Sha, et al. Generative ai for fast and accurate statistical computation of fluids. *arXiv preprint arXiv:2409.18359*, 2024.

[19] Chenkai Mao and Jonathan A Fan. Accurate and scalable deep maxwell solvers using multilevel iterative methods. *arXiv preprint arXiv:2509.03622*, 2025.

[20] Stefanos Bakirtzis, Marco Fiore, Jie Zhang, and Ian Wassell. Solving maxwell's equations with non-trainable graph neural network message passing, 2024.

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 1263–1272, June 2017.

Appendix A. Theoretical Foundations

1.1 Maxwell's Curl Equations

The propagation of electromagnetic waves in a 3D domain is governed by Maxwell's curl equations. Consider a linear medium characterized by permittivity ϵ, permeability μ, and electric conductivity σ. The time-domain coupled equations are expressed as:

$$\nabla \times \mathbf{E} = -\mu \frac{\partial \mathbf{H}}{\partial t} - \mathbf{M} \tag{A1}$$

$$\nabla \times \mathbf{H} = \epsilon \frac{\partial \mathbf{E}}{\partial t} + \mathbf{J} \tag{A2}$$

where $\mathbf{J} = \mathbf{J}_{src} + \sigma \mathbf{E}$ represents the total electric current density (combining independent sources $\mathbf{J}_{src}$ and conduction current $\sigma \mathbf{E}$). Similarly, $\mathbf{M}$ denotes the equivalent magnetic current density, typically introduced to model magnetic losses or specific magnetic sources.

1.2 3D FDTD Update Equations

In a standard 3D Yee grid, the vector equations (A1)-(A2) are decomposed into six scalar update equations. Taking the x-component of the magnetic field (H_x) as an example, its temporal evolution depends on the spatial derivatives of the electric field components E_y and E_z:

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu} \left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} - M_x \right) \tag{A3}$$

Discretizing this on the staggered grid yields the standard central difference update rule:

$$H_x^{n+\frac{1}{2}} = H_x^{n-\frac{1}{2}} + \frac{\Delta t}{\mu} \left[\frac{E_y^n|_{k+1} - E_y^n|_k}{\Delta z} - \frac{E_z^n|_{j+1} - E_z^n|_j}{\Delta y} \right] \tag{A4}$$

where spatial indices (i, j, k) are omitted for brevity where context implies local neighbor access.

1.3 NeCLO

For 3D FDTD, the core innovation of NeCLO is reinterpreting these spatial finite-difference operators as fixed-weight volumetric convolution kernels via `Conv3d`.

In the NeCLO architecture, the spatial derivative terms in Eq. (A4) are exacted by convolution operation. For instance, the partial derivative term $\frac{\partial E_z}{\partial y}$ is approximated as:

$$\frac{\partial E_z}{\partial y} \approx \frac{1}{\Delta y} \text{Conv3d}(\mathbf{E}_z, \mathbf{W}_{\partial y}) \tag{A5}$$

Here, $\mathbf{W}_{\partial y}$ is a $3 \times 3 \times 3$ sparse tensor designed to perform the neighbor difference along the y-axis. Mathematically, the weights at the center z-slice ($k = 0$) are:

$$\mathbf{W}_{\partial y}[:, :, 0] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{A6}$$

while weights at $k = -1$ and $k = 1$ are all zeros.

Consequently, the entire update step for the $\mathbf{H}$ field is vectorized on GPUs as a monolithic tensor operation:

$$\mathbf{H}^{n+\frac{1}{2}} \leftarrow \mathbf{H}^{n-\frac{1}{2}} - \frac{\Delta t}{\mu} \cdot \text{Conv3d}(\mathbf{E}^n, \mathbf{K}_{\text{curl}}) \tag{A7}$$

where $\mathbf{K}_{\text{curl}}$ aggregates the kernels for all spatial derivatives required to compute $\nabla \times \mathbf{E}$. This approach eliminates the need for explicit loops over spatial indices (i, j, k), leveraging the highly optimized CUDA implementations of `Conv3d`.

Appendix B. Implementation Code

In this section, we illustrate the evolution of the FDTD implementation. As shown in Listing 1, the legacy CPU-based approach (Method 1) relies on deeply nested loops, which are computationally prohibitive for high-resolution grids. Our proposed NeCLO approach (Method 2) vectorizes this process using 3D convolutions but necessitates computational overhead from padding. Another our proposed Tensor Slicing Method (Method 3) directly maps the stencil operations to memory pointers, achieving maximum throughput without auxiliary operations.

Listing 1: Comparison of FDTD Implementation Paradigms: From Loops to Slicing

```
# Method 1: Legacy CPU Implementation
for i in range(nx):
    for j in range(ny):
        for k in range(nz):
            # Explicit central difference
            dEz_dy = (Ez[i, j+1, k] - Ez[i, j, k]) / dy
            Hx[i, j, k] -= (dt / mu) * dEz_dy

# ========================================
# Method 2: Conv3d Approach (NeCLO)
# H shape: (Batch, 3, Nx, Ny, Nz)
pad_H = F.pad(H, (1,1, 1,1, 1,1))
curl_H = F.conv3d(pad_H, kernel_curl)
E += C_E * curl_H

# ========================================
# Method 3: Pure Tensor Slicing
# Vectorized Difference (e.g., dHz/dy)
# Slice [1:] represents index 'j+1', Slice [:-1]
    represents index 'j'
dHz_dy = Hz[:, :, 1:, :] - Hz[:, :, :-1, :]

Ex[:, :, :, 1:-1, 1:-1] = (
    CA * Ex[:, :, :, 1:-1, 1:-1] +
    CB * (dHz_dy[:, :, :, 1:-1, :] -
          dHy_dz[:, :, :, :, 1:-1])
)
```

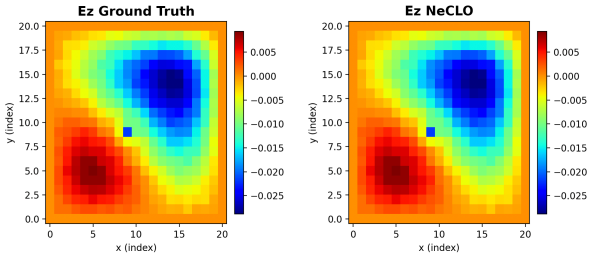
Appendix C. Additional Experimental Results

We provide additional computation throughput comparison and demonstrations of the numerical re-

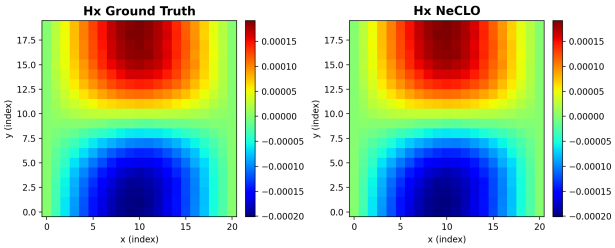
sults of our NeCLO and Tensor Slicing methods. The comparison of computation throughput is shown in the Table A1. The comparisons of our NeCLO and GroundTruth are shown in Figure A3 and Figure A2.

Table A1: CPU Performance Benchmark on Lossy Medium.

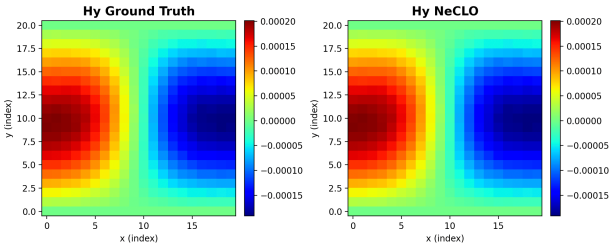
Method	Hardware	Latency (ms/step)	Throughput (Steps/s)	Speedup
Traditional FDTD	CPU	37.00	27	1.0×
NeCLO-Conv3d (Ours)	CPU	0.45	2, 232	~ 82×
Tensor Slicing	CPU	0.09	11, 111	~ 411×



(a) E_z Field Distribution



(b) H_x Field Distribution

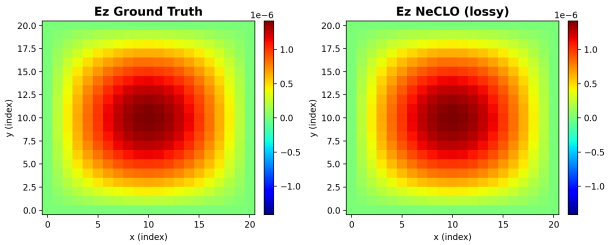


(c) H_y Field Distribution

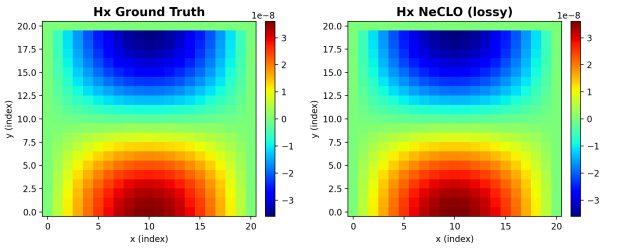
Fig. A1: Visual verification of numerical fidelity. The proposed NeCLO solver (Right) produces electromagnetic field distributions visually indistinguishable from the standard FDTD Ground Truth (Left) across all components (E_z, H_x, H_y).

Appendix D. Extended Discussion and Future Work

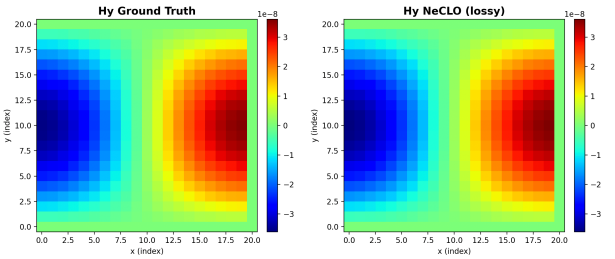
While the current NeCLO solver updates fields in a linear, non-dispersive medium, its fully differentiable nature makes it an ideal candidate for complex inverse design tasks. We propose extending the framework to support **generalized topology optimization** and **dispersive material modeling**.



(a) E_z Distribution ($\sigma = 4.0$ S/m)



(b) H_x Distribution ($\sigma = 4.0$ S/m)



(c) H_y Distribution ($\sigma = 4.0$ S/m)

Fig. A2: Visual verification of numerical fidelity in lossy media ($\sigma = 4.0$ S/m). The proposed NeCLO solver (Right) maintains high consistency with the FDTD Ground Truth (Left) across all field components (E_z, H_x, H_y), confirming robustness in dissipative environments.

4.1 Differentiable Topology Optimization

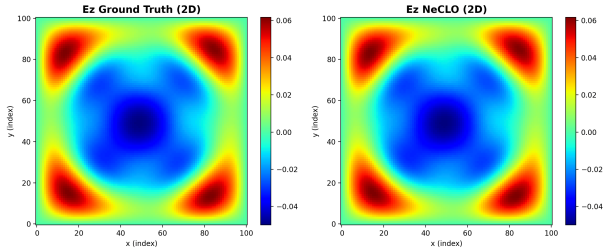
To evolve from simple parameter fitting to full-wave device design, we introduce a learnable material density mask $m(\mathbf{x}) \in [0, 1]$ directly into the update loop. The effective permittivity $\varepsilon_{\text{geo}}(\mathbf{x})$ becomes a differentiable function of the material topology:

$$\varepsilon_{\text{geo}}(\mathbf{x}) = m(\mathbf{x}) \cdot \varepsilon_{\text{mat}} + \big(1 - m(\mathbf{x})\big) \cdot \varepsilon_{\text{air}} \tag{A8}$$

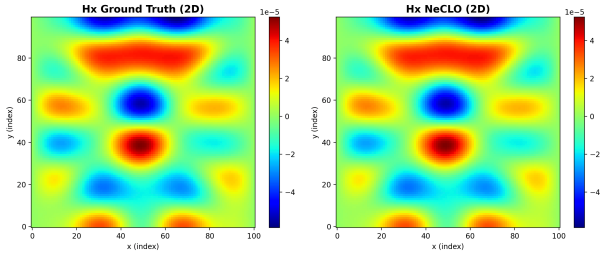
By treating $m(\mathbf{x})$ as a trainable tensor, gradients can be backpropagated from the target field response to the material distribution, allowing the automatic generation of metasurfaces or waveguides.

4.2 Dispersive Material Modeling

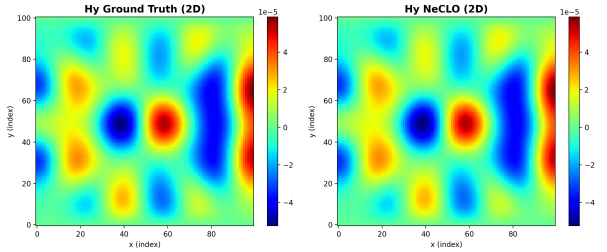
To handle frequency-dependent materials (e.g., metals in the optical regime), the update equation will be augmented with a dispersion function f_{disp}. The generalized update rule is formulated as:



(a) E_z Field Distribution



(b) H_x Field Distribution



(c) H_y Field Distribution

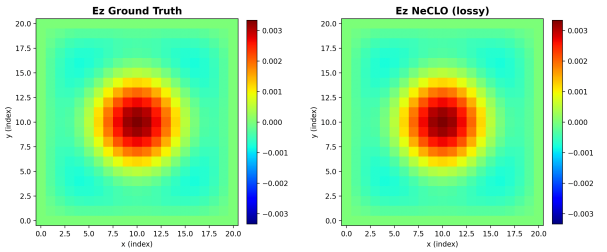
Fig. A3: Visual verification of numerical fidelity (2D TM Mode). The proposed NeCLO solver (Right) produces electromagnetic field distributions visually indistinguishable from the standard FDTD Ground Truth (Left) across all components (E_z, H_x, H_y).

$$E^{t+1}(\mathbf{x}) = E^t(\mathbf{x}) + \frac{\Delta t}{\varepsilon_{\text{geo}}(\mathbf{x})}(\nabla \times H^t)(\mathbf{x}) + f_{\text{disp}}\left(E^{t-k:t}(\mathbf{x}), m(\mathbf{x})\right) \tag{A9}$$

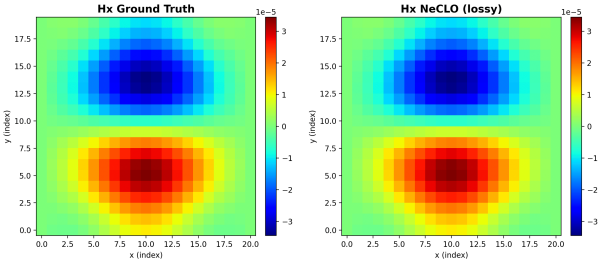
Here, f_{disp} can be implemented as either:

- An **Analytical Operator**: Implementing standard models like Lorentz-Drude or Debye via auxiliary differential equations (ADE).
- A **Neural Operator**: Using a recurrent neural network (RNN) or 1D convolution to learn complex, non-linear material responses from data.

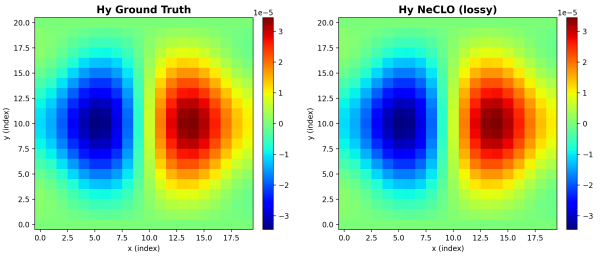
This framework will enable NeCLO to optimize spatial distributions $m(\mathbf{x})$ for specific spectral responses, effectively bridging the gap between fast simulation and deep-learning-based inverse design.



(a) E_z Field Distribution (σ = 4.0 S/m)



(b) H_x Field Distribution (σ = 4.0 S/m)



(c) H_y Field Distribution (σ = 4.0 S/m)

Fig. A4: Visual verification of numerical fidelity in 2D lossy media (σ = 4.0 S/m). The proposed NeCLO solver (Right) produces electromagnetic field distributions visually indistinguishable from the standard FDTD Ground Truth (Left) across all components (E_z, H_x, H_y).

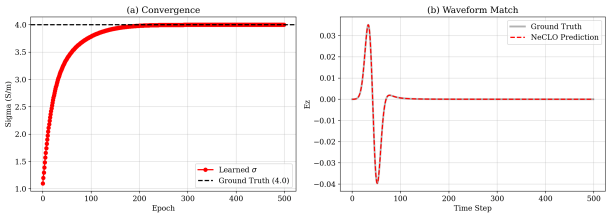


Fig. A5: Differentiable Parameter Inversion Results. (a) The learned conductivity σ converges to the ground truth value (4.0 S/m) within 500 epochs using double-precision training. (b) The time-domain waveform generated by the learned NeCLO model (Red Dashed) perfectly matches the CPU FDTD Ground Truth (Grey Solid), demonstrating that the differentiable solver has correctly identified the physical properties of the lossy medium.