Learning preconditioners for electromagnetic simulations

Project category: Physical Sciences

Team: Sattwik Deb Mishra (sdmishra@stanford.edu)

1 Introduction

Electromagnetic (EM) simulators play a vital role in the design of photonic devices for fields like silicon photonics, non-linear photonics, and more recently, quantum photonics. Given any electromagnetic structure (permittivity distribution), EM simulators solve Maxwell's equations numerically for electric and magnetic fields utilising techniques like finite-difference time-domain (FDTD), finite-difference frequency-domain (FDFD), finite element method (FEM) among others. This project will explore how to speed up these simulators by learning from simulation data. From decreasing design-fabrication-testing turnaround times to enabling faster algorithmic [1, 2] design of photonic devices, the benefits of faster EM simulators are aplenty.

For the purposes of this project only FDFD simulations will be considered. Every FDFD simulation of Maxwell's equations can be expressed in terms of a single linear equation to solve corresponding to a partial differential equation evaluated on a discretised grid,

$$[\nabla \times \nabla \times -\frac{\omega^2}{c^2}\varepsilon(\mathbf{x})]\mathbf{E}(\mathbf{x}) = -i\omega\mu_0\mathbf{J}(\mathbf{x}) \rightarrow Ax = b \tag{1}$$

where $\mathbf{E}(\mathbf{x})$ is the electric field and $\mathbf{J}(\mathbf{x})$ is the source current, ω is the frequency at which the device is being simulated, and $\varepsilon(\mathbf{x})$ is the permittivity, i.e., the device structure, as a function of space. Thus, the differential form of Maxwell's equations and the electromagnetic structure of the device to be simulated decide the sparse matrix A, which is hence also called the Maxwell operator matrix. The sources of the EM fields in the problem decide the vector b and x represents the fields to solve for.

The goal of the project is to learn a left preconditioner P that makes this linear equation solve faster. A preconditioner is multiplied to both sides of the equation to obtain $PAx = Pb$. A learned preconditioner P should make this equation better conditioned for iterative numerical methods for solving linear equations.

2 Related work

Data-driven preconditioners have been previously studied in [3] wherein a preconditioner is learned for the pressure Poisson equation for computational fluid dynamics simulations. A CNN is used to model the right preconditioner M^{-1} and the loss function for the simulation is the condition number of the matrix AM^{-1} where A is the sparse matrix obtained from discretisation of the Poisson equation. The use of the CNN model ensures that the preconditioner is agnostic to the size of the simulation. Further, the structure of the CNN used ensures that the learned preconditioner is sparse and hence, computationally inexpensive to apply.

An alternative, matrix-free approach to speeding up iterative solvers could instead be to learn the finite-difference stencil used to discretize the simulation space [4].

3 Dataset and features

Optimization-based inverse design (starting from random initial states) [1] is employed to construct a library of 600 wavelength demultiplexers, which are split into 500 training and 100 test devices. A wavelength demultiplexer is a photonic device that splits different wavelengths of incoming light into separate channels. A single training example $x^{(i)} \equiv (\varepsilon^{(i)}(\mathbf{x}), A^{(i)})$ where $\varepsilon^{(i)}(\mathbf{x})$ is the permittivity distribution of the device, $A^{(i)}$ is the corresponding FDFD matrix, and i indexes the training devices in the library generated. The sparse matrices $A^{(i)}$ are of dimension 46875×46875, $\approx 45\times$ larger than the matrices used for training in [3]. The figure below shows the permittivity distribution of a typical device in the library,

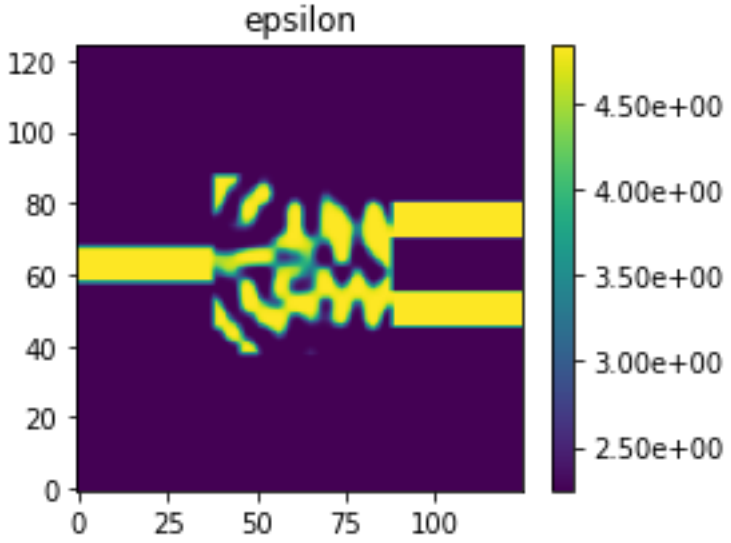

Figure 1: Permittivity distribution of a typical training/test wavelength demultiplexer

4 Methods

We consider a constant, diagonal matrix preconditioner model where the diagonal elements are the weights to be learned. By a 'constant' model, we mean that the preconditioner is not a function of the input permittivity distribution.

As the ideal preconditioner for A is the inverse A^{-1}, we consider the following objective for learning,

$$J(P) = \frac{1}{N_b} \sum_{i=1}^{N} \left\| PA^{(i)} - \mathbb{1} \right\|^2. \tag{2}$$

where $\mathbb{1}$ is the identity matrix, $\|.\|$ represents the Frobenius norm, and N_b is the batch size. Compared to the loss function based on condition number used in [3], we use this function because it is much more efficient to compute for large matrices and captures the same information as the condition number, i.e., how close the matrix PA is to $\mathbb{1}$. The initial weights are chosen to be all 1, i.e., $P = \mathbb{1}$, to enable comparison to when there is no preconditioning.

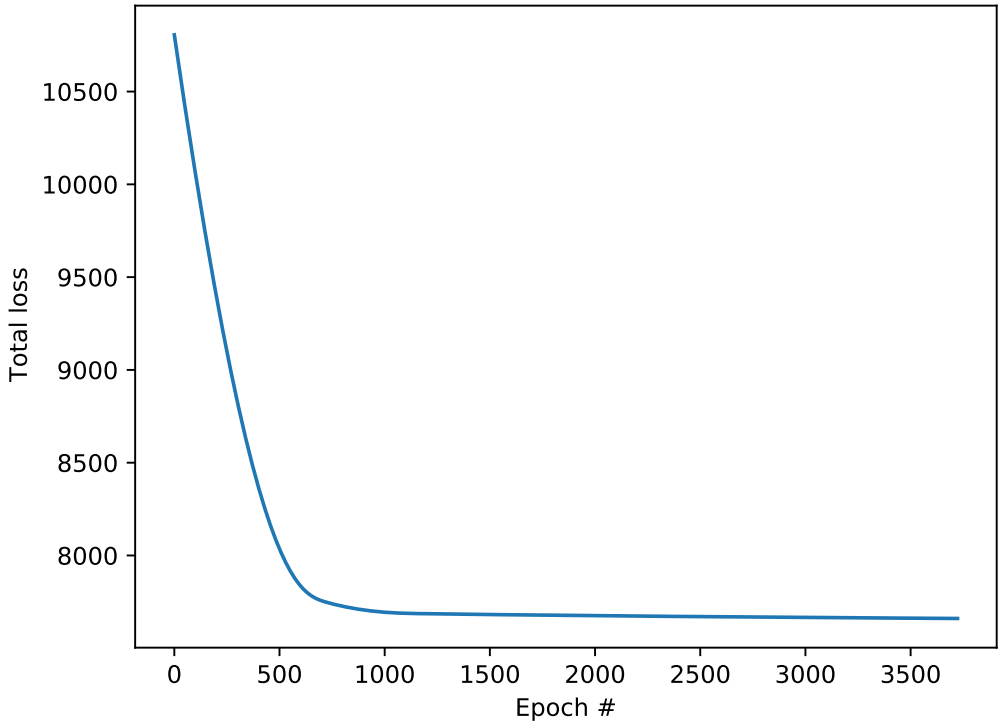

Figure 2: Total loss during training.

5 Results

We train the diagonal preconditioner model using the algorithm *Adam* [5]. Gradient of the loss is computed using the auto-differentiation features of the library JAX [6].

Fig. 2 shows the total loss for $N = 500$ training samples, i.e.

$$J_{\text{tot}}(P) = \frac{1}{N} \sum_{i=1}^{N} \left\| PA^{(i)} - \mathbb{1} \right\|^2 . \tag{3}$$

Clearly, the training converges. After training, we evaluate the loss of each test example with the trained preconditioner and with no preconditioner. The mean and standard deviation of the test loss with the trained preconditioner applied are 15.32 and 4.1×10^{-4}, respectively, whereas the mean and standard deviation of the test loss with no preconditioner are 21.62 and 1.9×10^{-6}. The learned preconditioner clearly improves the conditioning of the test examples too.

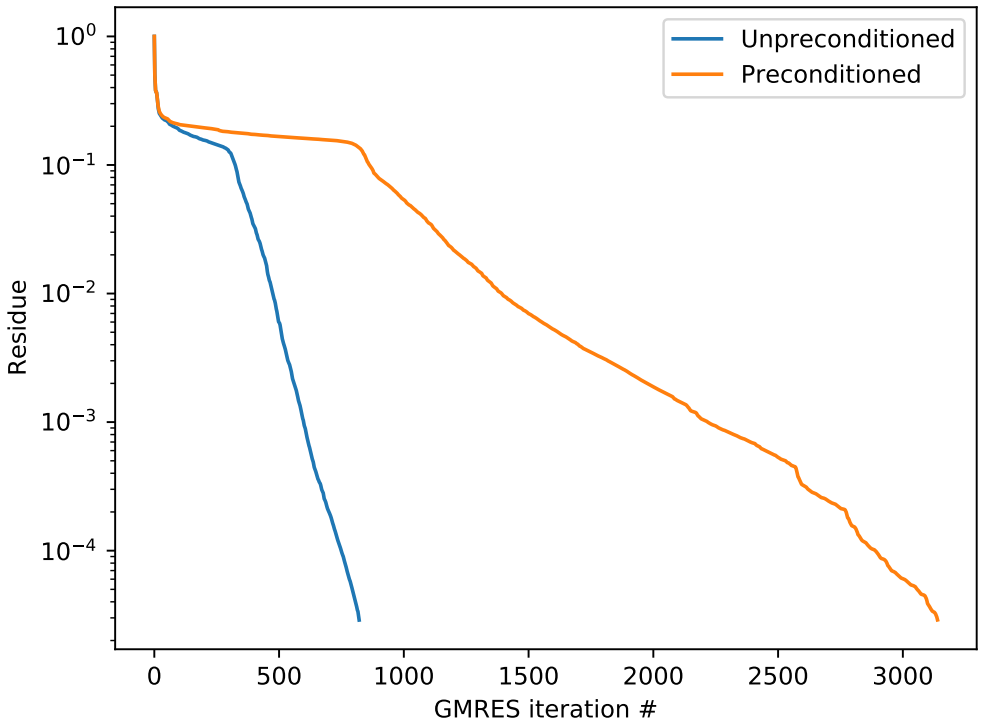

Figure 3: Convergence of iterative solver.

Unfortunately, as shown in Fig. 3, when the convergence of an iterative solver (GMRES, Generalized Minimal Residual method) is studied on an example, the preconditioner worsens the performance of the iterative solver. We suspect this is due to the

nature of the iterative solver where an improvement in conditioning does not necessarily imply an improvement in convergence.

6 Conclusion

In this project, we have applied supervised learning to learn preconditioners for numerical solution of Maxwell's equations. We trained a diagonal preconditioner and found that it improves conditioning of the Maxwell operator matrix but it worsens the performance of a numerical solver (GMRES).

For future work, instead of using the Frobenius norm-based loss function, the residual of GMRES can be used as the loss function, ensuring that the performance of the numerical solver is directly improved. Convergence can also be studied with other solvers. We note that [3] studies preconditioners for positive semi-definite matrices and conjugate gradient solver, which has provably improved convergence with improved condition numbers. In terms of other models, as the Maxwell operator matrices are very large and sparse, if CNNs are to be used as models, they will need to be customised to work with sparse tensors [7].

7 Neural network models

Changing the training from unsupervised to supervised with the following objective function –

$$J(P) = \frac{1}{N_b} \sum_{i=1}^{N_b} \left\| P(\epsilon^{(i)})b - e^{(i)} \right\|^2 \tag{4}$$

where $e^{(i)}$ are the solutions obtained from iterative solver for $A^{(i)}e^{(i)} = b$. The preconditioner P will be modeled as a 2D convolution, realised by a CNN.

8 Contributions

S.D.M. produced the data, trained the model, and wrote the report. S.D.M. thanks Logan Su and Jinhie Skarda for their help with understanding the code base of SPINS [1] and with debugging the data generation code. S.D.M. thanks Rahul Trivedi and Logan Su for the idea of the project.

References

[1] Nanophotonic inverse design with SPINS: Software architecture and practical considerations, Su et al., Applied Physics Reviews 7, 011407 (2020); https://doi.org/10.1063/1.5131263. Open source version available at, https://github.com/stanfordnqp/spins-b.

[2] Data-driven acceleration of photonic simulations, Trivedi et al., Scientific Reports volume 9, Article number: 19728 (2019), https://www.nature.com/articles/s41598-019-56212-5

[3] Deep Learning of Preconditioners for Conjugate Gradient Solvers in Urban Water Related Problems, Sappl et al., arXiv:1906.06925 [cs.LG], https://arxiv.org/abs/1906.06925

[4] Learning data-driven discretizations for partial differential equations, Bar-Sinai et al., PNAS 116, 15344–15349 (2019), https://www.pnas.org/content/116/31/15344

[5] Adam: A Method for Stochastic Optimization, D. P. Kingma and J. Ba, arXiv:1412.6980 [cs] (2017), https://arxiv.org/abs/1412.6980

[6] https://github.com/google/jax

[7] High-dimensional convolutional neural networks for 3D perception, C. Choy, Ph.D. Thesis, https://purl.stanford.edu/fg022dx0979