Georgia Tech College of Computing
School of Computer Science

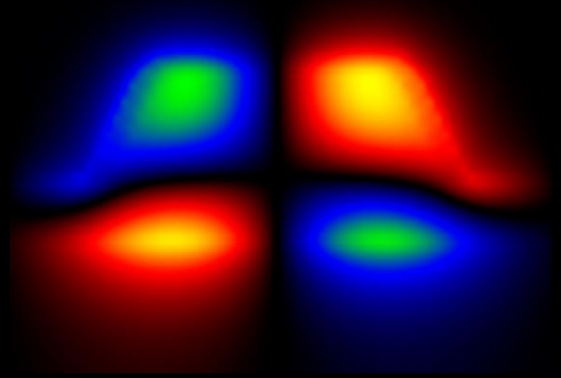
Georgia Tech College of Engineering
School of Electrical and Computer Engineering

GPU-Accelerated Electromagnetic Mode Solvers for High Performance Computing Applications (GAEMSHiPCA)

John Moxley

Joel Slaby

Conner Yurkon



March 16, 2023

Dielectric Waveguide Mode Solver

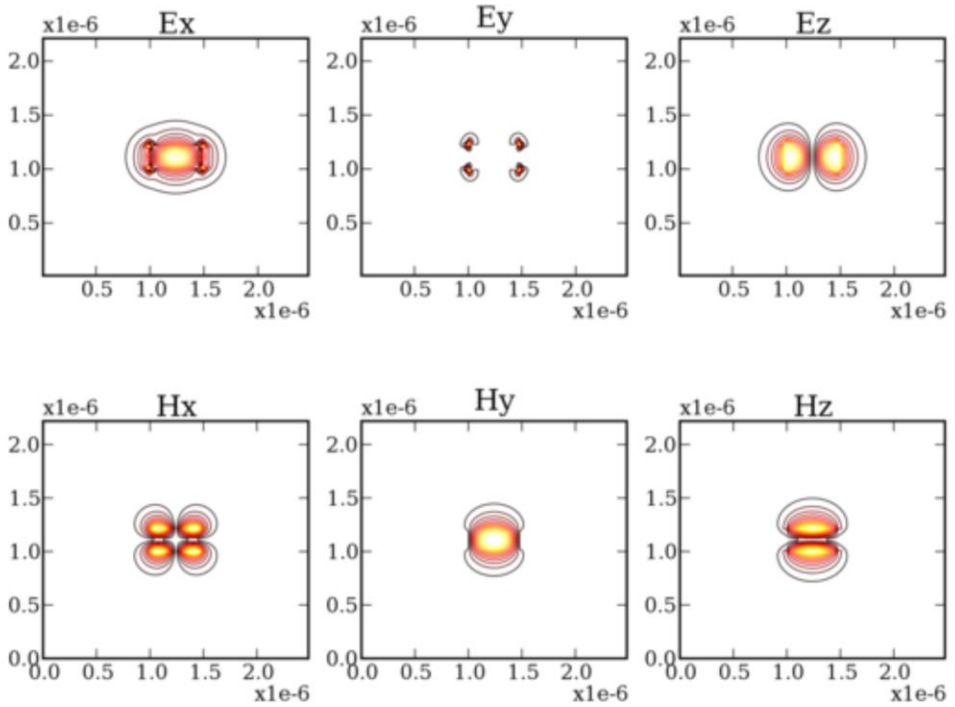
Georgia Institute of Technology

- **Goal:** Find solution to maxwell's equations for a given geometry
  - Iterative eigenvalue problem with sparse matrices

$$(\mu^{-1}\nabla \times \epsilon^{-1}\nabla \times)\mathbf{H} = \omega^2\mathbf{H}$$

- **Current challenges:**
  - Solving for eigenvalues is an expensive calculation
  - Limited to smaller structures at lower resolutions
  - Speed-up can enable:
    - faster device design and optimizations
    - higher resolution calculations (more accurate)

- **Existing work:**
  - A GPU Solver for Sparse Generalized Eigenvalue Problems With Symmetric Complex-Valued Matrices Obtained Using Higher-Order FEM: https://ieeexplore.ieee.org/iel7/6287639/8274985/08468163.pdf
  - This work uses the finite-element method (FEM), which isn't scalable to larger geometries. We will use the finite-difference method (FD)



https://ieeexplore.ieee.org/iel7/6287639/8274985/08468163.pdf

Understanding Light Propagation

- Light is described with both an electric field, E, and a magnetic field, H
  - Each field has three components (x, y, z)
  - Total of 6 field components $(E_x, E_y, E_z, H_x, H_y, H_z)$
  - Each field is a function of *position* and *time*
- How are the fields connected? - **Maxwell equations**
- Time varying magnetic fields induce spatial varying electric fields (and vice versa)
  - Time varying fields can also be considered in frequency domain using a fourier transform

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}$$

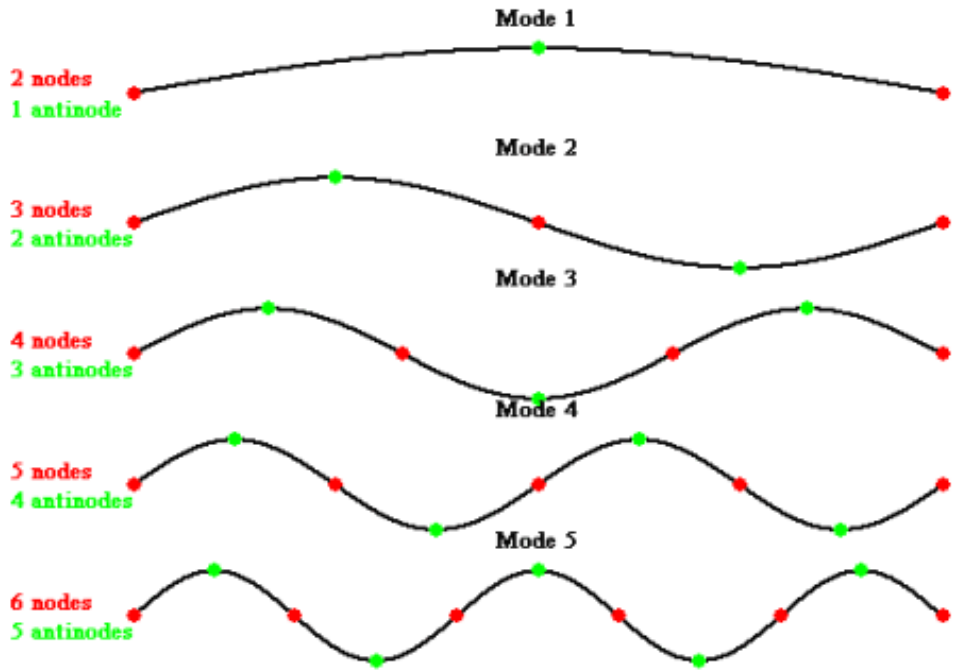
$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{j} + \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}$$

Understanding Waveguide Modes

- Traditional wave equation is of the form: $\frac{d^2u}{dt^2} = c^2\frac{d^2u}{dx^2}$
- Solution $u = A\sin(kx + \omega t)$
  - ω is time dependence
  - k is the spatial dependence (wave vector)
- A **mode** is one of these solutions to the wave equation
  - It is a spatially stable solution
  - Each mode is defined by its wave vector k
- Light is a wave! $\frac{d^2E_x}{dz^2} = \mu\epsilon\frac{d^2E_x}{dt^2}$



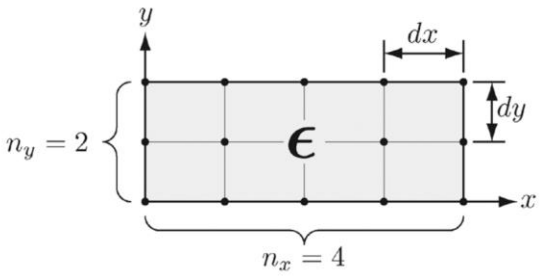
https://blog.soton.ac.uk/soundwaves/standing-waves/2-string-modes/ isvr

- We can combine maxwell's equations to get a similar form for the E and H fields
- Thus, for a given waveguide geometry (ϵ profile) there exists a stable solution to maxwell equations following the form of the wave equation

Mode Solver Problem Breakdown

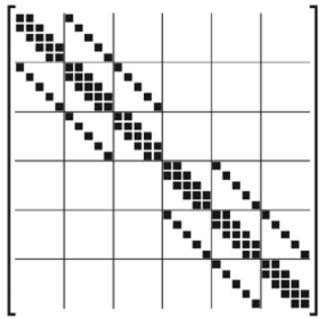
Georgia Institute of Technology

(1) Define your geometry



(2) Create the maxwell operator matrix $\boldsymbol{A}$
sparse matrix

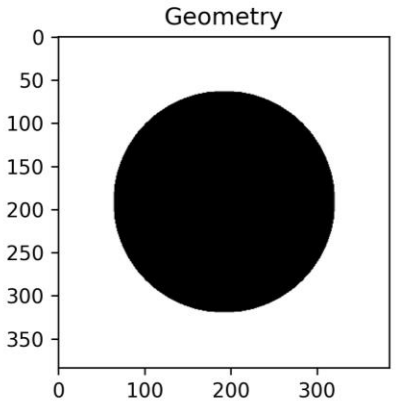
$$\underbrace{(\mu^{-1}\nabla \times \epsilon^{-1}\nabla \times)}_{\boldsymbol{A}}\mathbf{H} = \omega^2\mathbf{H}$$

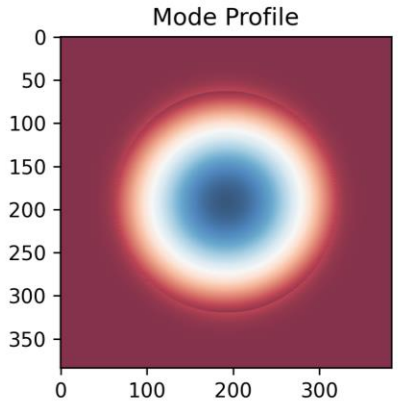


(3) Calculate the eigenvalues
This is where HPC comes into play

$$\begin{bmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{bmatrix} \begin{bmatrix} H_x \\ H_y \end{bmatrix} = \beta^2 \begin{bmatrix} H_x \\ H_y \end{bmatrix}$$

(4) Compute the mode fields (eigenvectors)

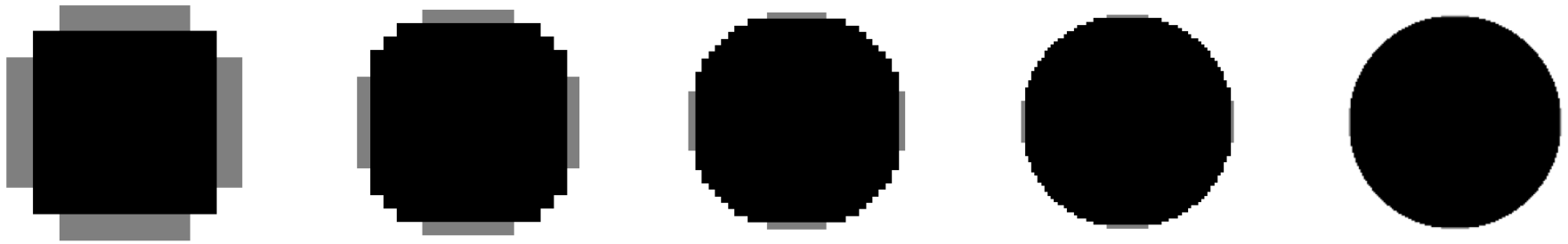




A. B. Fallahkhair, K. S. Li and T. E. Murphy, "Vector Finite Difference Modesolver for Anisotropic Dielectric Waveguides", J. Lightw ave Technol. 26(11), 1423-1431, (2008).

Performance Metrics

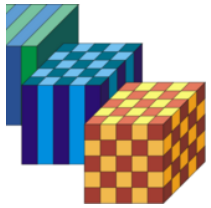
- **Speed vs Resolution**
  - For a given resolution, how long does the algorithm take to complete?
  - Focusing on GPU speedup—fix the number of processors, memory, etc.
  - Compare the accelerated version to the baseline open-source versions
- **Performance of the Resolution (convergence)**
  - At what rate does the resolution increase as we allow for more execution time?
  - This needs to be at least $O(\Delta n^2)$
  - With the speedup, we'll be able to more accurately calculate the convergence rate, because we'll be able to simulate at higher resolutions



Baseline Metrics

- Current state-of-the-art mode solving represented by many open-source repositories

- **Mpb**
  - https://github.com/NanoComp/mpb
  - Implemented in C
  - Includes implementation with distributed memory using MPI



- **Empy**
  - Implemented in python
  - Fully vectorial finite difference method
  - https://github.com/lbolla/EMpy

EMpy

Electromagnetic Python

- **Modesolver**
  - Implemented in python
  - https://github.com/jtambasco/modesolverpy

Solution: GPU Accelerated Mode Solver

Georgia Institute of Technology

- **Goal:** Accelerate a presently implemented mode solver on a GPU

- **Design:**
  - Start with existing solution (serial), open-source such as *empy*
  - Convert the eigenmode problem to be GPU compatible
    - Will require parallelizable pre-conditioner
    - Possible algorithms include locally block preconditioned conjugate gradient (LOBPCG)
  - Validate accuracy and determine GPU speed-up

- **Challenges:**
  - Solving an eigenmode problem requires an iterative solution
  - Will have to take advantage of matrix sparsity to reduce gpu memory requirements

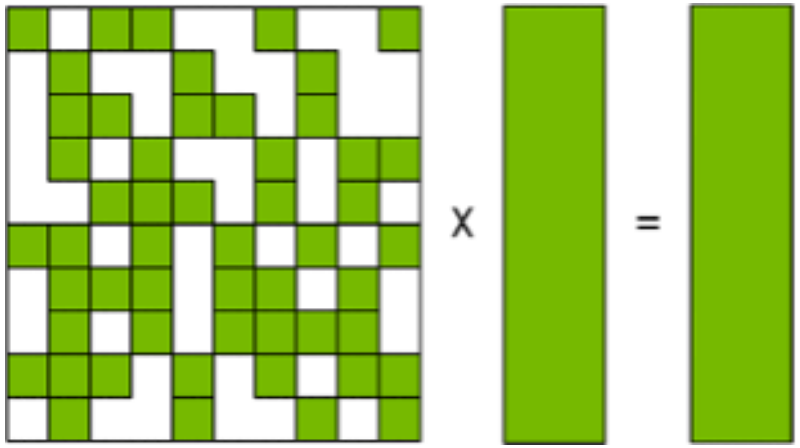
Taking Advantage of Matrix Sparsity on GPU

Georgia Institute of Technology

- We can use the fact that matrices we will be working with are *sparse*, i.e., very few coefficients are nonzero values

- Memory consumption can be reduced (and performance increased) by using special representations of these matrices, storing only the nonzero coefficients

- For portions of the algorithm that involve multiplying matrices, we can take advantage of Sparse-Matrix Dense-Matrix Multiplication (SpMM) on CUDA
  - The cuSPARSE library provides `cusparseSpMM` for this purpose



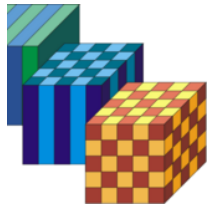
https://developer.nvidia.com/blog/accelerating-matrix-multiplication-with-block-sparse-format-and-nvidia-tensor-cores/

Validation

- Results will be compared against current open-source mode solvers
- Multiple mode solvers will be used in tandem since accuracy also depends on simulation resolution

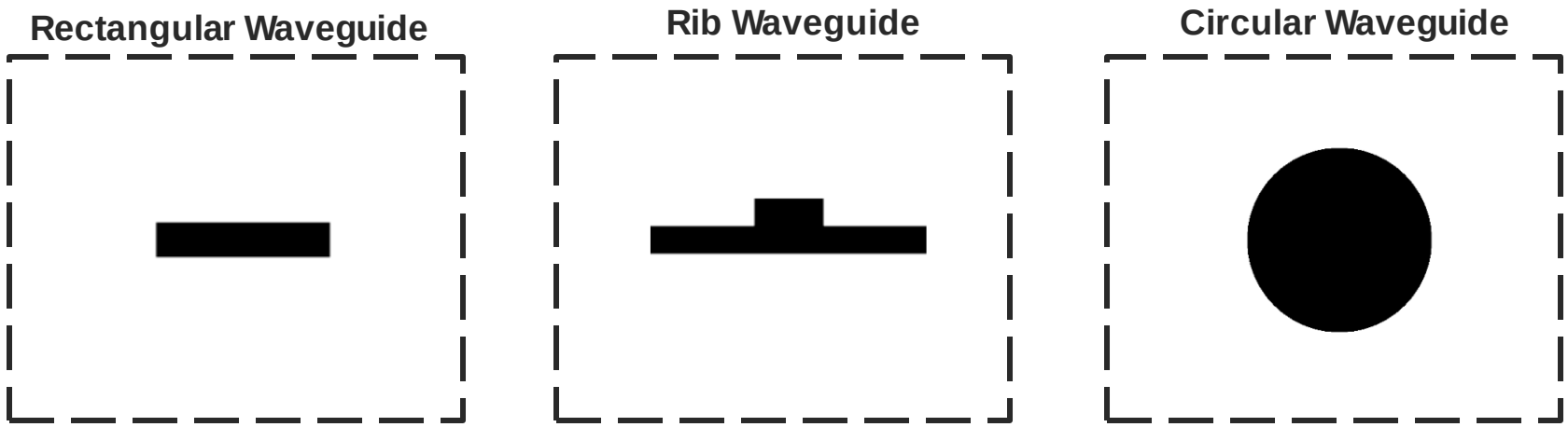
EMpy

Electromagnetic Python



Planned Experiments

- **Datasets:** Select three geometries (primarily isotropic but with consider anisotropic dielectrics)
  - Rectangular waveguide
  - Rib waveguide
  - Circular waveguide



- **Testbed:**
  - PACE Cluster (coc-ice, coc-ice-multi, coc-ice-gpu)
- **Potential plots:**
  - Speed vs Resolution
  - Performance vs Resolution (convergence)