OpenParEM2D
Methodology

Overview

- OpenParEM2D implements the full-wave 2D method described in Lee's paper:
  - Jin-Fa Lee, “Finite Element Analysis of Lossy Dielectric Waveguides”, *IEEE Tran. Microwave Theory and Techniques*, vol. 42, no. 6, June 1994, pp. 1025-1031.
- The finite elements are implemented using MFEM (https://mfem.org/).
- The generalized eigenvalue problem is solved using SLEPc (https://slepc.upv.es/).
- SLEPc is also used to solve standard Ax=b linear problems.

Electric Fields

- Equation (12) from Lee is an eigenvalue problem in terms of the electric fields and the complex propagation constant:

$$\sum_{\Omega_e}\left(\begin{bmatrix}\frac{1}{\mu_r}[S_t]_e - k_o^2\bar{\epsilon}_r[T_t]_e & 0\\ 0 & 0\end{bmatrix}\right)\begin{bmatrix}\bar{E}_t\\ \bar{E}_z\end{bmatrix} = \gamma^2\sum_{\Omega_e}\left(\begin{bmatrix}\frac{1}{\mu_r}[T_t]_e & \frac{1}{\mu_r}[G]_e\\ \frac{1}{\mu_r}[G]_e^T & \frac{1}{\mu_r}[S_z]_e - k_o^2\bar{\epsilon}_r[T_z]_e\end{bmatrix}\right)\begin{bmatrix}\bar{E}_t\\ \bar{E}_z\end{bmatrix} \quad (12)$$

where T_t, G, S_z, T_z, and S_t are defined by equation (13) as

$$\begin{aligned}
\underline{W_t}[T_t]_e\bar{E}_t &= \int_{\Omega_e}\vec{w}_t\cdot\vec{e}_t\,d\Omega\\
\underline{W_t}[G]_e\bar{E}_z &= \int_{\Omega_e}\vec{w}_t\cdot\nabla_t e_z\,d\Omega\\
\underline{W_z}[S_z]_e\bar{E}_z &= \int_{\Omega_e}\nabla_t w_z\cdot\nabla_t e_z\,d\Omega\\
\underline{W_z}[T_z]_e\bar{E}_z &= \int_{\Omega_e}w_z e_z\,d\Omega\\
\underline{W_t}[S_t]_e\bar{E}_t &= \int_{\Omega_e}(\nabla_t\times\vec{w}_t)\cdot(\nabla_t\times\vec{e}_t)\,d\Omega
\end{aligned} \quad (13)$$

- The process is to build (12) from (13), then solve the eigenvalue problem for γ and the electric field.
- The integrals in (13) are supported by library calls in MFEM.
  - E_t is computed on a vector 2D finite element space using Nedelec finite elements. These curl-conforming elements prevent the calculation of spurious modes.
  - E_z is computed on a scalar 2D finite element space using L2 finite elements.
- MFEM is built around real numbers in the time domain. Engineering electromagnetics is generally done in the frequency domain using complex numbers. To use MFEM to build a frequency-domain solver, the real and imaginary parts are separated, then the eigenvalue problem is built up as a complex problem that is solved by a complex solver in SLEPc.

- The first step is to translate (13) into MFEM function calls on the two finite element spaces. The functions used are

$$
\begin{aligned}
\underline{W_t}\left[T_t\right]_e\overline{\mathrm{E}}_t &= \int_{\Omega_e}\vec{w}_t\cdot\vec{e}_t\,d\Omega && \text{VectorFEMassIntegrator}\\
\underline{W_t}\left[G\right]_e\overline{\mathrm{E}}_z &= \int_{\Omega_e}\vec{w}_t\cdot\nabla_t e_z\,d\Omega && \text{MixedVectorGradientIntegrator}\\
\underline{W_z}\left[S_z\right]_e\overline{\mathrm{E}}_z &= \int_{\Omega_e}\nabla_t w_z\cdot\nabla_t e_z\,d\Omega && \text{DiffusionIntegrator}\\
\underline{W_z}\left[T_z\right]_e\overline{\mathrm{E}}_z &= \int_{\Omega_e} w_z e_z\,d\Omega && \text{MassIntegrator}\\
\underline{W_t}\left[S_t\right]_e\overline{\mathrm{E}}_t &= \int_{\Omega_e}\left(\nabla_t\times\vec{w}_t\right)\cdot\left(\nabla_t\times\vec{e}_t\right)d\Omega && \text{CurlCurlIntegrator}
\end{aligned}
\qquad (14)
$$

- In addition, (14) is applied over the finite elements, and the complex dielectric constant and/or the real permeability can change from element to element. So multiple instantiations of the integrators are required to cover the dependencies on material variations across the 2D cross section plus real and imaginary parts.
  - Note again that it is assumed that the dielectric constants can be complex to include dielectric losses, while the permeability is assumed to be real. Both can vary over the cross section.
- The integrators are set up in fem2D::fem2D in fem2D.cpp, where a total of 9 are required.
- The integrated matrices are saved to disk then imported into eigensolve.c to solve the eigenvalue problem with SLEPc.
  - It is necessary to use a separate program because MFEM is built around a compilation of PETSc using real numbers, and SLEPc [in this usage] is built around a compilation of PETSc using complex numbers.
- Once the eigenvalue problem is solved, the eigenvalues are passed back to the main program in real and imaginary parts and the eigenvectors are saved to disk.
- The main program reads the complex-valued files into real and imaginary data structures and finishes processing with real-valued MFEM functionality.

Magnetic Fields

- With the electric fields and complex propagation constants calculated, the magnetic fields can be calculated.
- E_t is known over a finite element space using Nedelec finite elements.
- E_z is known over a finite element space using L2 finite elements.
- The magnetic fields must be similarly broken into H_t and H_z components and solved over the correct spaces with correct elements. H_t is solved with Nedelec elements and H_z is solved with L2 elements.
- Breaking H into the H_t and H_z parts yields two equations

$$j\omega\mu \overline{H}_t = -\nabla_t \times E_z \hat{z} + \gamma \hat{z} \times \overline{E}t$$
$$j\omega\mu H_z = -\nabla_t \times \overline{E}_t$$

- With E_t and E_z known, these two can be solved separately. In matrix form, this leads to two standard linear problems of the form Ax=b.

$$\left[M_t\right]\left[\overline{H}_t\right] = \gamma\left[Z_t\right]\left[\overline{E}_t\right] - \left[C_z\right]\left[\overline{E}_z\right]$$
$$\left[M_z\right]\left[\overline{H}_z\right] = -\left[C_t\right]\left[\overline{E}_t\right]$$

- The matrices are built up with integrators from MFEM in fem2D::fem2D in fem2D.cpp.

$\left[M_t\right]$ VectorFEMassIntegrator
$\left[M_z\right]$ MassIntegrator
$\left[Z_t\right]$ VectorFEMassIntegrator
$\left[C_t\right]$ MixedScalarCurlIntegrator
$\left[C_z\right]$ MixedVectorGradientIntegrator

- The M_t and M_z integrators integrate over μ, which is assumed to be real and possibly vary from element to element.
- The Z_t and C_z integrators include a matrix operation to obtain the cross product with the unit vector in the z direction.
- All of the integrators are real-valued, so just the 5 matrices need to be constructed using real variables. Complex variables are not required until solving the final matrix equation.
- Like for the eigenvalue calculation, the matrices are exported to eigenvalue.c where the standard Ax=b problem is solved twice in SLEPc using complex numbers, once for H_t and once for H_z.
- Once solved, the H-fields are saved to files for import back into the main program into real and imaginary data structures where processing continues using MFEM functions.

Poynting Vector

- With E_t, E_z, H_t, and H_z known, then the Poynting Vector can be calculated.
- For a 2D simulator, the quantity of interest is the power flowing in the z-direction.

$$
\begin{aligned}
P_z &= \frac{1}{2}\overline{E}\times\overline{H}^{*}\cdot\hat{z} \\
&= \frac{1}{2}\overline{E}_t\times\overline{H}_t^{*} \\
&= \frac{1}{2}\left[\left(\overline{E}_t^{\mathrm{Re}}\times\overline{H}_t^{\mathrm{Re}}+\overline{E}_t^{\mathrm{Im}}\times\overline{H}_t^{\mathrm{Im}}\right)+j\left(\overline{E}_t^{\mathrm{Im}}\times\overline{H}_t^{\mathrm{Re}}-\overline{E}_t^{\mathrm{Re}}\times\overline{H}_t^{\mathrm{Im}}\right)\right]
\end{aligned}
$$

- Each of the terms are real and computed with MFEM grid functions for plotting.
- These are then integrated over the cross section to obtain the total complex P_z.
- The code is implemented in Fields::calculatePz in fem2D.cpp.

Losses

- The setup and solution of (12) makes three assumptions regarding losses.
  - All conductors are either perfect electric conductor (PEC) or perfect magnetic conductor (PMC).
  - Dielectrics can have complex permittivity.
  - Dielectrics have real permeability.
- With this setup, the results from the solution of (12) only include dielectric losses.
- To add in conductor losses, the magnitude of the tangential magnetic field on the conductors is multiplied by the surface resistance. These loss estimates are added to the dielectric losses to get the total losses.
- For the conductor losses to be accurate, the conductors must be thick compared to the skin depth. It is up to the user to ensure that this condition is met.
- Since field penetration into the conductors is not included in the solution of (12), then the small effect of the field penetration on the propagation constant is not included.

Adaptive Mesh Refinement

- Adaptive mesh refinement uses the Zienkiewicz-Zhu error estimation method as implemented in MFEM.
  - See Mode::ZZrefineMesh in fem2D.cpp.
- The setup requires an integrator, and ideally, an integrator over the complete electric field would be used.
- Naturally, MFEM does not include such a specific integrator, but it does include a curlcurl integrator, which represents part of the electric field.
- With the curlcurl integrator, the adaptive mesh refinement algorithm calculates mesh errors on part of the electric field.
  - This naturally leads to less accurate error estimates for the mesh elements, potentially leading to less accurate selection of mesh elements for refinement.
- Numerical experiments across a variety of transmission lines and waveguides show that the adaptive mesh refinement is good enough. Mesh refinements occur where they should leading to demonstrable high accuracy.
- Replacing the curlcurl integrator would require the writing of a custom integrator, and this is a challenging task.