OpenParEM3D Theory, Methodology, and Accuracy

Version 1.0

September 2024

Brian Young

Contents

1 Introduction

OpenParEM3D is a full-wave electromagnetic solver using the finite-element method to solve for frequency-dependent S-parameters and fields. The Galerkin procedure is applied to Maxwell's equations to derive the weak form of the wave equation in the electric field $\overline{E}$. The integrals are calculated with calls to the MFEM library [1][2], boundary conditions are applied, and finally a standard $\overline{\overline{A}}\,\overline{x} = \overline{b}$ linear problem is solved for $\overline{E}$. Post-processing produces the magnetic field $\overline{H}$ and S-parameters. The methodology uses fully populated matrices, so OpenParEM3D is not configured for GPU processing.

Boundary conditions include perfect magnetic conductor (PMC), perfect electric conductor (PEC), surface impedance, radiation, and 2D ports. The 2D ports are assumed to be driven by transmission lines or waveguides, generally referred to as wave ports, so OpenParEM2D is used to calculate the 2D fields in setting up the boundary value problem.

This document covers the theory and methodology of how OpenParEM3D builds and solves the fields and S-parameters and provides the results of test cases demonstrating accuracy. For details about how to set up and run OpenParEM3D, see the separate document "OpenParEM3D_Users_Manual.pdf".

2 Theory and Mapping to MFEM

2.1 Wave Equation

Starting at the most fundamental level with Maxwell's equations, we have

$$\nabla \times \overline{E} = -\mathrm{j}\omega\mu\overline{H} \tag{1}$$

and

$$\nabla \times \overline{H} = \overline{J} + \mathrm{j}\omega\epsilon\overline{E}. \tag{2}$$

Let $\overline{J} = \sigma\overline{E}$ and $\epsilon_c = \frac{\sigma}{\mathrm{j}\omega} + \epsilon$, then (2) becomes

$$\nabla \times \overline{H} = \mathrm{j}\omega\epsilon_c\overline{E}. \tag{3}$$

Eliminating $\overline{H}$ from (3) and (1) yields

$$\nabla \times (-\frac{1}{\mathrm{j}\omega\mu}\nabla \times \overline{E}) = \mathrm{j}\omega\epsilon_c\overline{E}. \tag{4}$$

Multiply through by $\mathrm{j}\omega\mu_\circ$ and set $k_\circ^2 = \omega^2\mu_\circ\epsilon_\circ$ to get the wave equation

$$\nabla \times (\frac{1}{\mu_r}\nabla \times \overline{E}) = k_\circ^2\epsilon_{cr}\overline{E}, \tag{5}$$

where $\epsilon_{cr} = \epsilon_c/\epsilon_\circ$ and $\mu_r = \mu/\mu_\circ$.

2.2 Galerkin's Procedure

Multiply (5) by a test field $\overline{T}$ and integrate over the volume, Ω, to get

$$\iiint_\Omega \nabla \times (\frac{1}{\mu_r}\nabla \times \overline{E}) \cdot \overline{T} dV = \iiint_\Omega k_\circ^2\epsilon_{cr}\overline{E} \cdot \overline{T} dV. \tag{6}$$

The surface of Ω is designated by δ. Conventionally, the normal to the volume at the surface is given by $\hat{n}$, which points out of the 3D volume. Apply the vector identity

$$\iiint_\Omega \nabla \times \overline{u} \cdot \overline{v}\, d\Omega = \iiint_\Omega \overline{u} \cdot \nabla \times \overline{v}\, d\Omega - \iint_\delta (\overline{u} \times \hat{n}) \cdot \overline{v}\, dS \tag{7}$$

with $\overline{u} = \frac{1}{\mu_r}\nabla \times \overline{E}$ and $\overline{v} = \overline{T}$ and rearranging, then

$$\iiint_\Omega \frac{1}{\mu_r}\nabla \times \overline{E} \cdot \nabla \times \overline{T}\, dV - k_\circ^2 \iiint_\Omega \epsilon_{cr}\overline{E} \cdot \overline{T} dV - \iint_\delta (\frac{1}{\mu_r}\nabla \times \overline{E} \times \hat{n}) \cdot \overline{T}\, dS = 0. \tag{8}$$

Reorder the cross products involving $\hat{n}$ to get the weak form of the wave equation in $\overline{E}$ as

$$\iiint_\Omega \frac{1}{\mu_r}\nabla\times\overline{E}\cdot\nabla\times\overline{T}\,dV - k_\circ^2\iiint_\Omega \epsilon_{cr}\overline{E}\cdot\overline{T}dV + \iint_\delta \hat{n}\times(\frac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS = 0. \tag{9}$$

In (9) the first two terms are volume integrals over the entire simulation domain, while the third term is a surface integral over the outer surface of the simulation domain and represents the boundary conditions that must be applied to obtain a unique solution. Most of the work in implementing a simulator is in setting up the various components of the surface integrals.

2.3 Boundary Conditions

The third term in (9) captures the boundary conditions, and the entire surface must be treated to produce a unique solution. The boundary can be subdivided into areas requiring different boundary conditions as

$$\begin{aligned}
\iint_\delta \hat{n}\times(\tfrac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS \;=\;& \iint_{\delta_{\mathrm{PMC}}} \hat{n}\times(\tfrac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS && \text{PMC}\\
+\;& \iint_{\delta_{\mathrm{PEC}}} \hat{n}\times(\tfrac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS && \text{PEC}\\
+\;& \textstyle\sum_{i=1}^{M_Z}\iint_{\delta_{Z_{si}}} \hat{n}\times(\tfrac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS && \text{surface impedance}\\
+\;& \textstyle\sum_{i=1}^{M_R}\iint_{\delta_{R_i}} \hat{n}\times(\tfrac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS && \text{radiation}\\
+\;& \textstyle\sum_{i=1}^{M_P}\iint_{\delta_{P_i}} \hat{n}\times(\tfrac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS && \text{ports}
\end{aligned} \tag{10}$$

2.3.1 PMC

The perfect magnetic conductor (PMC) from the first line of (10) is the easiest boundary to implement because simply nothing has to be done. The PMC boundary is the natural boundary that happens when the degrees of freedom (DOF)[1] of the boundary are left to float.

2.3.2 PEC

The perfect electric conductor (PEC) from the second line of (10) is applied by setting the boundary DOFs of PEC boundaries to zero. Nedelec finite elements are used for $\overline{E}$, and these only have tangential components at boundaries. Setting the DOFs to zero at the boundary then forces the tangential component of $\overline{E}$ to zero at the boundary, satisfying the PEC boundary condition.

2.3.3 Surface Impedance

The third line in (10) implements a surface impedance Z_{si} over M_Z sections of the boundary. Z_s relates the tangential electric and magnetic fields on the surface via

$$\overline{E}_t = -Z_s\,\hat{n}\times\overline{H}, \tag{11}$$

where $\hat{n}$ is the unit vector normal to the surface pointing out of the 3D space. The minus sign is necessary since $\overline{E}\times\overline{H}$ points outward for power dissipation, then $-\hat{n}\times\overline{H}$ points in the direction of $\overline{E}$. Substituting $\overline{H}$ from (1) yields

$$\overline{E}_t = \frac{Z_s}{\mathrm{j}\omega\mu}\hat{n}\times\nabla\times\overline{E}. \tag{12}$$

This can be directly substituted into the third line of (10) to find that

$$\sum_{i=1}^{M_Z}\iint_{\delta_{Z_{si}}} \hat{n}\times(\frac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS = \sum_{i=1}^{M_Z}\iint_{\delta_{Z_{si}}} \frac{\mathrm{j}\omega\mu_\circ}{Z_{si}}\overline{E}_t\cdot\overline{T}\,dS. \tag{13}$$

[1] A variable used to implement a finite element is referred to as a degree of freedom (DOF). A higher-order finite element requires more DOFs than a lower-order finite element.

2.3.4 Radiation

The fourth line in (10) implements a radiation boundary condition (RBC) over M_R sections of the boundary. Using a 1st-order RBC, also known as a Sommerfeld RBC, propagation is assumed in the far field to have the dependence $e^{-\mathrm{j}kr}$ in the $\hat{n}$ radial direction, where k is the propagation constant and r is the radial distance from the radiating source, then

$$\hat{n} \times (\frac{1}{\mu_r} \nabla \times \overline{E}) = \frac{\mathrm{j}k}{\mu_r} \overline{E}. \tag{14}$$

This can be easily demonstrated in Cartesian coordinates by working through the math with $\overline{E} = E_o e^{-jkz} \hat{x}$. Plugging (14) back into the 4th line of (10) produces

$$\sum_{i=1}^{M_R} \iint_{\delta_{R_i}} \hat{n} \times (\frac{1}{\mu_r} \nabla \times \overline{E}) \cdot \overline{T} \, dS = \sum_{i=1}^{M_R} \iint_{\delta_{R_i}} \frac{\mathrm{j}k}{\mu_r} \overline{E} \cdot \overline{T} \, dS. \tag{15}$$

Noting the similar forms for (13) and (15), the equivalent "surface impedance" of the RBC can be found by equating the factors $\frac{\mathrm{j}\omega\mu_o}{Z_{si}}$ and $\frac{\mathrm{j}k}{\mu_r}$, then noting that $k = \omega\sqrt{\mu\epsilon}$ and simplifying results in $Z_{si} = \sqrt{\mu/\epsilon}$, which is simply the wave impedance for a plane wave. So in short, the 1st-order RBC is equivalent to setting the boundary to a surface impedance equal to the wave impedance.

To apply the 1st-order RBC, the electromagnetic fields must be in a plane wave configuration at the boundary, with $\overline{E}$ perpendicular to $\overline{H}$ and both perpendicular to the surface. In practical simulation setups with reasonably sized volumes, this condition will not be met with high precision. However, it can be met with engineering precision, so it is practical for typical design work. When the boundary is too close to the radiating structure, plots show a weak but visible standing wave at the boundary.

2.3.5 Port

The fifth line in (10) implements ports, where energy enters and exits the 3D space as modes of transmission lines or waveguides. Since ports are on the surface, they are 2D. It is assumed that the fields at the ports represent transmission line or waveguide solutions with a dependence in the direction of propagation $\hat{n}$ of $e^{-\gamma n}$. This assumption means that the ports can only be applied to planar 2D surfaces on the boundary of the 3D space. Ports defined in this way are generally referred to as *wave ports*.

A distinction must be made between ports drawn on the 3D surface and S-parameter ports. A closed outline drawn on a planar region of the 3D surface is a physical port representing a transmission line or waveguide, which may support one or more propagating modes. For example, a port capturing two symmetric strip transmission lines with the port boundary being set to PEC supports two modes: even and odd. Every mode of a port becomes a column and row in the final S-parameter matrix, and so in this example one port provides two rows and columns. The name used here to describe the row/column of the S-parameter matrix is *S-parameter port* or *S-port*. For the example with two symmetric strip transmission lines, one port leads to 2 S-ports. When there is one mode per port, then the number of ports and S-ports are equal.

To ultimately solve for the S-parameters, a port must be driven with a 2D field configuration while the remaining ports must be terminated with a 2D absorbing boundary condition (2D ABC). So *two* boundary conditions are needed for the ports: driving and 2D ABC.

For the port boundary condition when driving, the 2D solution from OpenParEM2D is simply imposed onto the port by equating the port boundary DOFs to the values computed in OpenParEM2D. The imposed solution may be the dominant or a higher-order mode, depending on the problem setup. Continuing with the symmetric strip transmission line example, a port is driven once for the even mode and a second time for the odd mode.

To construct a 2D ABC, the 5th line of (10) can be re-written under the assumption that a propagating transmission line or waveguide mode is used such that the field dependence along the direction of travel, $\hat{n}$, is $e^{-\gamma n}$, then

$$\hat{n} \times \frac{1}{\mu_r} \nabla \times \overline{E} = -\frac{1}{\mu_r} \frac{\partial \overline{E}_t}{\partial n} + \frac{1}{\mu_r} \nabla_t E_n = \frac{\gamma}{\mu_r} \overline{E}_t + \frac{1}{\mu_r} \nabla_t E_n, \tag{16}$$

The gradient term $\nabla_t E_n$ is not directly supported by Nedelec finite elements in MFEM and requires additional consideration.

OpenParEM3D uses Nedelec finite elements because the divergence for the elements is zero, so the two $\nabla\cdot$ terms of Maxwell's equations are satisfied and spurious (i.e. incorrect) solutions are avoided. A Nedelec finite element contains only tangential vector terms on the faces of the element, so $\nabla_t E_n$ presents a problem in that the boundary does not have a normal component on which to make the calculation. During a simulation, the normal component on the boundary is computed as a natural result of the 3D solution, so the simulations correctly include the normal component. However, the MFEM library does not include a function to directly implement the needed gradient term on the boundary, perhaps due to the need to pull the needed information from multiple finite elements.

To obtain the needed gradient term, the wave equation in 3D given by (5) can be revisited in 2D at the port. In 2D, the dependence in the direction of propagation is $e^{-\gamma n}$, and the electric field and operator can be divided into tangential and normal components as $\overline{E} = \overline{E}_t + E_n\hat{n}$ and $\nabla = \nabla_t - \gamma\hat{n}$. Applying these to (5) and equating the tangential and normal components results in two coupled equations

$$\nabla_t \times \frac{1}{\mu_r}\nabla_t \times \overline{E}_t - \gamma^2\frac{1}{\mu_r}\overline{E}_t - \gamma\frac{1}{\mu_r}\nabla_t E_n = k_\circ^2\epsilon_{cr}\overline{E}_t \tag{17}$$

and

$$\nabla_t \cdot \frac{1}{\mu_r}\nabla_t E_n + \frac{1}{\mu_r}\gamma\nabla_t \cdot \overline{E}_t + k_\circ^2\epsilon_{cr}E_n = 0. \tag{18}$$

The needed terms from (16) can be found from (17) to obtain

$$\frac{\gamma}{\mu_r}\overline{E}_t + \frac{1}{\mu_r}\nabla_t E_n = \frac{1}{\gamma}(\nabla_t \times \frac{1}{\mu_r}\nabla_t \times \overline{E}_t - k_\circ^2\epsilon_{cr}\overline{E}_t), \tag{19}$$

and the terms on the right side of the equality are supported by MFEM since only tangential components are called for. Plugging (19) back into the 5^{th} line of (10) produces

$$\sum_{i=1}^{M_P}\iint_{\delta_{P_i}} \hat{n}\times(\frac{1}{\mu_r}\nabla\times\overline{E})\cdot\overline{T}\,dS = \sum_{i=1}^{M_P}\iint_{\delta_{P_i}}\frac{1}{\gamma}(\nabla_t\times\frac{1}{\mu_r}\nabla_t\times\overline{E}_t - k_\circ^2\epsilon_{cr}\overline{E}_t)\cdot\overline{T}\,dS. \tag{20}$$

There is still the issue of (18), which is not coded into the simulation. The normal components at the boundary derive their behavior from the 3D behavior of the fields near the boundary and are governed by the wave equation (5). The general solution in 3D space then ensures that (18) is satisfied. It would be beneficial to numerically prove this observation, but the needed functions are not provided by the MFEM library for the reasons discussed above. However, the accuracy demonstrations in Sec. 6 include an example with a TM field configuration with significant E_n field strength for which the 2D ABC is shown to be very effective.

There is a very important point to consider and understand with respect to (20): the presence of γ. Only one value for γ can be applied at a given port in its 2D ABC. If there is a single propagating mode on the port, then it is matched and the 2D ABC is very effective. If there is more than one propagating mode on the port, and the modes do not have the same γ, only one mode can be fully absorbed by the 2D ABC. Any other modes will suffer some reflection with a reflection coefficient roughly equal to the ratio of the γs for the modes. For example, coupled stripline will not suffer reflections since the even and odd modes have equal γ, but coupled microstrip will suffer some reflection since the γ for the even and odd modes are not equal. The impact of any reflection from the 2D ABC for mismatched γs from multiple modes may or may not be relevant to engineering applications for a given problem. To avoid active S-parameters, at a multimode port the largest γ is used for the 2D ABC.

2.4 Calculating $\overline{H}$ from $\overline{E}$

Slightly rearranging (1) to find an expression for $\overline{H}$ produces

$$\overline{H} = -\frac{1}{\mathrm{j}\omega\mu}\nabla\times\overline{E}. \tag{21}$$

Applying the Galerkin procedure by multiplying through by a weight $\overline{T}$ and integrating over the volume results in

$$\iiint_\Omega \overline{H} \cdot \overline{T}\, dV = \mathrm{j} \iiint_\Omega \frac{1}{\omega\mu} \nabla \times \overline{E} \cdot \overline{T}\, dV. \tag{22}$$

These volume integrals are directly supported by MFEM. No additional boundary conditions need to be applied in addition to those applied when solving for $\overline{E}$.

2.5 Construction and Solution of the $\overline{\overline{A}}\,\overline{x} = \overline{b}$ Problem

The boundary value problem to be solved to find $\overline{E}$ in the 3D volume and on the boundaries is defined by (9) and (10). To recap, the boundary conditions are given by (13) for impedance, (15) for radiation, and (20) for 2D ABCs at ports. The role of the MFEM library is to provide the function calls to implement each mathematical operation in these equations (volume and boundary) by using finite elements to build matrices enabling the construction of the standard numerical problem $\overline{\overline{A}}\,\overline{x} = \overline{b}$. Additional boundary conditions are applied directly to the $\overline{\overline{A}}\,\overline{x} = \overline{b}$ problem by setting DOFs for PEC boundaries and for 2D port excitations. Ultimately, the linear problem is solved for $\overline{x}$ given $\overline{b}$, where $\overline{x}$ are the DOFs for $\overline{E}$ and $\overline{b}$ are the boundary conditions.

MFEM is fundamentally built around real variables, while OpenParEM3D is a frequency-domain solver requiring complex variables. While there are some wrappers in MFEM to connect real data structures into complex data structures, an issue is that MFEM requires the real variable version of PETSc [3]. It is possible to solve the complex $\overline{\overline{A}}\,\overline{x} = \overline{b}$ problem using real variables, but testing showed that performance is dramatically better using complex variables, which requires the complex version of PETSc. So throughout the code of OpenParEM3D, MFEM is used to construct real matrices for the real and imaginary parts of the math, and then these two real matrices are combined into complex matrices for solution with PETSc compiled for complex variables. There is an impact on dynamic memory usage because the real and complex data structures have to be allocated at the same time before complex construction is complete and the real data structure can be deleted. However, there is not a significant impact on run time from the data translations since the vast majority of simulation time is spent solving the complex $\overline{\overline{A}}\,\overline{x} = \overline{b}$ problem.

The construction and solution of the 3D electromagnetic problem is executed in `fem3D::solve`. Discussion on each step follows.

2.5.1 Building $\overline{\overline{A}}$

With the exception of the application of the PEC boundary, construction of $\overline{\overline{A}}$ occurs in `fem2D::build_A`. Construction involves making appropriate calls to MFEM methods to implement the needed physics in finite elements.

In solving (9), the MFEM mapping of the first two integrals are

$$\begin{array}{lcl} \iiint_\Omega \frac{1}{\mu_r} \nabla \times \overline{E} \cdot \nabla \times \overline{T}\, dV & \rightarrow & \text{CurlCurlIntegrator} \\ k_\circ^2 \iiint_\Omega \epsilon_{cr} \overline{E} \cdot \overline{T} dV & \rightarrow & \text{VectorFEMassIntegrator} \end{array} \tag{23}$$

The matrix is built by MFEM as a `ParMixedBilinearForm` structure that is converted to a `HypreParMatrix` parallel matrix suitable for parallel processing using the Message Passing Interface (MPI) through the `Assemble` and `Finalize` operations. Since the `ParMixedBilinearForm` is rendered into filled-out matrices, GPU processing is ruled out due to the large memory allocation for typical problems.

The remaining part of implementing (9) involves applying the boundary conditions, with ports, surface impedance, and radiation boundary conditions applied in `fem2D::build_A`.

For non-driving ports, port boundary conditions are applied in `Port::addPortIntegrators`. The 2D ABC boundary condition is defined in (20), and it is implemented in `Port::addPortIntegrators` using MFEM CurlCurlIntegrator and VectorFEMassIntegrator on the port boundaries. Note at the top of `Port::addPortIntegrators` how γ is selected for multimode ports.

Impedance and radiation boundary conditions are applied in `Boundary::addImpedanceIntegrators`. Here, the method name is appropriate because both (13) and (15) are implemented with an MFEM VectorFEMassIntegrator applied over the appropriate boundaries.

The final step in preparing $\overline{\overline{A}}$ in `fem3D::build_A` is to combine the real and imaginary `HypreParMatrix`s into a PETSc complex `MAT` matrix using in the `hypre_ParCSRMatrixToMat` routine. As the complex `MAT` is constructed, the `HypreParMatrix`s are deleted.

2.5.2 PEC Boundary

After the call to `fem3D::build_A`, the next boundary condition to apply is the PEC boundary. The first step is to identify the PEC boundary DOFs using `fem3D:build_PEC_dofs`, then these are used to enforce the PEC boundary by zeroing the row and column for each PEC DOF in $\overline{\overline{A}}$ while placing the number 1 on the diagonal in `eliminatePEC` from file `solveComplexLinearSystem.c`.

2.5.3 Driving Port Boundary

The final boundary condition to apply is the electric field from the 2D solution from OpenParEM2D at the driving port. There are no MFEM calls associated with setting this boundary condition. With the driving mode on the driving port identified, `Mode::fillX` fills a vector with the known boundary DOFs on the port.

2.5.4 Solution

The driving port DOFs and $\overline{\overline{A}}$ are passed to the function `solveComplexLinearSystem` in file `solveComplexLinearSystem.c`, which builds $\overline{b}$ and solves the completed $\overline{\overline{A}}\,\overline{x} = \overline{b}$ problem using PETSc's KSP infrastructure. Once solved for the 3D electric field DOFS, the DOFs are translated back into MFEM data structures with calls to `fem3D::build_e_re_e_im` and `fem3D::buildEgrids`. Further plotting and post-processing for S-parameters uses the data in the MFEM ParGridFunction data structure.

2.5.5 H Field

OpenParEM3D uses the H-field to separate forward- and reverse- traveling waves at the ports, so it is always required to calculate the magnetic fields. The H-field is derived from the E-field using (22), which is calculated in `fem3D::buildHgrids`. The integrals in (22) are set up in `fem3D::build_P` for the left-hand side using an MFEM VectorFEMassIntegrator, followed by conversion to complex PETSc MAT, and in `fem3D::build_Q` for the right-hand side using an MFEM MixedVectorCurlIntegrator, also followed by conversion to complex PETSc MAT. Since the right-hand side involves $\overline{E}$, which is known at this point, the E-field DOFs are applied to generate $\overline{b}$ for an $\overline{\overline{A}}\,\overline{x} = \overline{b}$ problem, with $\overline{\overline{A}} = \overline{\overline{P}}$, which is solved in `solveHfield` using the PETSc KSP infrastructure.

Like for the electric fields, once the H-field DOFs are known, they are translated back into MFEM data structures with a call to `fem3D::build_h_re_h_im` and construction of ParGridFunction structures which are used for S-parameter calculations and plotting.

3 S-parameter Calculation

The primary engineering outputs from OpenParEM3D are S-parameters, which are post-processed from the computed 3D electric and magnetic fields. Consider a black box with N S-ports as shown in Fig. 1, where the physical ports as discussed in Sec. 2.3.5 are not shown. The S-parameters of the black box relate the outward-traveling waves b_i to the inward-traveling waves a_i using the S-parameter matrix as $\overline{b} = \overline{\overline{S}}\,\overline{a}$, which is shown in expanded form in (24). The goal is to find $\overline{\overline{S}}$ from 3D electromagnetic field calculations.

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1N} \\ S_{21} & S_{22} & \dots & S_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ S_{N1} & S_{N2} & \dots & S_{NN} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} \tag{24}$$

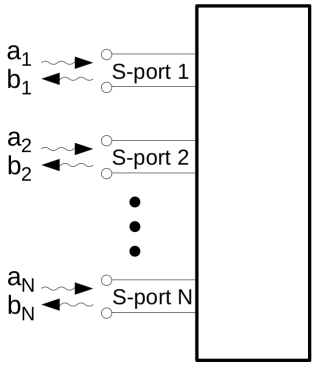


Figure 1: S-parameter black box with N S-parameter (S-port) ports.

Note that b_i and a_i are traveling waves on a transmission line or waveguide, and each b_i and a_i represents one S-port. Each traveling wave is a mode of the transmission line or waveguide. There are one or more modes, hence S-ports, per physical port.

3.1 b_i and a_i

To begin to solve for $\overline{\overline{S}}$, b_i and a_i must be defined in terms of quantities available from the electromagnetic simulation. The fundamental definition for b_i in terms of voltage is

$$b_i = \frac{V_i^+}{\sqrt{Z_{oi}}}, \tag{25}$$

where the + direction is outward from the 3D space. The voltage is available from the tangential electric field on the 2D physical port using the basic definition of voltage to get

$$V_i^+ = -\int_{\ell_i} \overline{E}_{ti}^+ \cdot \overline{d\ell}, \tag{26}$$

where ℓ_i is the voltage integration path on the i^{th} S-port. $\overline{E}_{ti}^+$ is available as the weighted field from the 2D port simulations such that

$$\overline{E}_{ti}^+ = \mathrm{C}_i^+ \overline{E}_{tmi}^+, \tag{27}$$

where C_i^+ is a weight that must be determined and $\overline{E}_{tmi}^+$ is the modal tangential electric field known from the 2D port simulation. Plugging (27) into (26) produces

$$V_i^+ = -\int_{\ell_i} \mathrm{C}_i^+ \overline{E}_{tmi}^+ \cdot \overline{d\ell} = \mathrm{C}_i^+ V_{mi}^+, \tag{28}$$

where $V_{mi}^+ = -\int_{\ell_i} \overline{E}_{tmi}^+ \cdot \overline{d\ell}$ is the voltage calculated from the 2D port simulation. Finally, (25) becomes

$$b_i = \frac{\mathrm{C}_i^+ V_{mi}^+}{\sqrt{Z_{oi}}}. \tag{29}$$

A similar sequence of operations produces a relationship for a_i as

$$a_i = \frac{\mathrm{C}_i^- V_{mi}^-}{\sqrt{Z_{oi}}}. \tag{30}$$

3.2 C_i^+ and C_i^-

The unknowns in obtaining b_i and a_i are C_i^+ and C_i^-, so these must be found from the 3D electromagnetic solution. At any physical port, the total electric field is equal to the sum of the weighted modal fields at that port, giving

$$\overline{E}_t = \sum_{i=1}^{N} \mathrm{C}_i^+ \overline{E}_{tmi}^+ + \sum_{i=1}^{N} \mathrm{C}_i^- \overline{E}_{tmi}^-, \tag{31}$$

where N is the number of modes at the physical port. For modal waves traveling in the + and − directions, the electric field is the same, so $\overline{E}_{tmi}^+ = \overline{E}_{tmi}^-$, and (31) simplifies to

$$\overline{E}_t = \sum_{i=1}^{N} (\mathrm{C}_i^+ + \mathrm{C}_i^-) \overline{E}_{tmi}^+. \tag{32}$$

Dot producting through by $\overline{E}_{tmk}^{+*}$, where $*$ is the complex conjugate, and integrating over the surface of the port leads to

$$\iint_{S_i} \overline{E}_{tmk}^{+*} \cdot \overline{E}_t \, dS = \sum_{i=1}^{N} (\mathrm{C}_i^+ + \mathrm{C}_i^-) \iint_{S_i} \overline{E}_{tmk}^{+*} \cdot \overline{E}_{tmi}^+ \, dS. \tag{33}$$

Modal fields are orthogonal, so $\iint_{S_i} \overline{E}_{tmk}^{+*} \cdot \overline{E}_{tmi}^+ \, dS = 0$ for $k \neq i$, and (33) simplifies to

$$\iint_{S_i} \overline{E}_{tmi}^{+*} \cdot \overline{E}_t \, dS = (\mathrm{C}_i^+ + \mathrm{C}_i^-) \iint_{S_i} \overline{E}_{tmi}^{+*} \cdot \overline{E}_{tmi}^+ \, dS \tag{34}$$

Equation (34) provides one equation in two unknowns, so additional information is required. Using the magnetic field, the similar starting point to (31) is

$$\overline{H}_t = \sum_{i=1}^{N} \mathrm{C}_i^+ \overline{H}_{tmi}^+ - \sum_{i=1}^{N} \mathrm{C}_i^- \overline{H}_{tmi}^-, \tag{35}$$

where the C_i^+ and C_i^- carry over since the electric and magnetic fields are components of the same mode. There is a change in sign for the reverse-traveling wave so that the power flow is in the correct direction. Following the same line of derivation for the magnetic field as for the electric field, then

$$\iint_{S_i} \overline{H}_{tmi}^{+*} \cdot \overline{H}_t \, dS = (\mathrm{C}_i^+ - \mathrm{C}_i^-) \iint_{S_i} \overline{H}_{tmi}^{+*} \cdot \overline{H}_{tmi}^+ \, dS, \tag{36}$$

which provides a second equation for the two unknowns C_i^+ and C_i^-.

With the 3D solution of $\overline{E}$ and $\overline{H}$, (34) and (36) can be solved to find C_i^+ and C_i^-. The necessary integrations are supported by the MFEM library, with the calculations performed in the method `Mode::calculateSplits`.

3.3 Solving for $\overline{\overline{S}}$

To simplify the discussion, consider a 2-port problem with a 2×2 S-parameter matrix. Starting with

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \tag{37}$$

b_i and a_i can be substituted using (29) and (30) to get

$$\begin{bmatrix} \frac{\mathrm{C}_1^+ V_{m1}^+}{\sqrt{Z_{o1}}} \\ \frac{\mathrm{C}_2^+ V_{m2}^+}{\sqrt{Z_{o2}}} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} \frac{\mathrm{C}_1^- V_{m1}^-}{\sqrt{Z_{o1}}} \\ \frac{\mathrm{C}_2^- V_{m2}^-}{\sqrt{Z_{o2}}} \end{bmatrix}. \tag{38}$$

Rearranging (38) to form a linear equation with the matrix values of $\overline{\overline{S}}$ as unknowns yields

$$\begin{bmatrix} \frac{C_1^- V_{m1}^-}{\sqrt{Z_{o1}}}[1] & \frac{C_2^- V_{m2}^-}{\sqrt{Z_{o2}}}[1] & 0 & 0 \\ 0 & 0 & \frac{C_1^- V_{m1}^-}{\sqrt{Z_{o1}}}[1] & \frac{C_2^- V_{m2}^-}{\sqrt{Z_{o2}}}[1] \\ \frac{C_1^- V_{m1}^-}{\sqrt{Z_{o1}}}[2] & \frac{C_2^- V_{m2}^-}{\sqrt{Z_{o2}}}[2] & 0 & 0 \\ 0 & 0 & \frac{C_1^- V_{m1}^-}{\sqrt{Z_{o1}}}[2] & \frac{C_2^- V_{m2}^-}{\sqrt{Z_{o2}}}[2] \end{bmatrix} \begin{bmatrix} S_{11} \\ S_{12} \\ S_{21} \\ S_{22} \end{bmatrix} = \begin{bmatrix} \frac{C_1^+ V_{m1}^+}{\sqrt{Z_{o1}}}[1] \\ \frac{C_2^+ V_{m2}^+}{\sqrt{Z_{o2}}}[1] \\ \frac{C_1^+ V_{m1}^+}{\sqrt{Z_{o1}}}[2] \\ \frac{C_2^+ V_{m2}^+}{\sqrt{Z_{o2}}}[2] \end{bmatrix}. \tag{39}$$

Here, [1] indicates a first simulation driving S-port 1 with a 2D ABC at S-port 2, while [2] indicates a second simulation driving S-port 2 with a 2D ABC at S-port 1. Solving the linear $\overline{\overline{A}}\,\overline{x} = \overline{b}$ problem defined by (39) produces the needed solution for the unknown S-parameters.

It is straightforward to generalize (39) for N S-ports. In general, an S-parameter matrix with N S-ports requires N 3D simulations.

3.4 Driving Sets

The formulation in Sec. 3.3 is general and enables options for setting up the 3D simulations for solving S-parameters. To calculate S-parameters, one or more S-ports are driven, 2D ABCs are applied, then the outputs from all ports are simulated and used to calculate $\overline{\overline{S}}$. For a given setup, the collection of driven S-ports are called a `driving set`.

At this time, just one driving set is enabled, the "single" driving set described below. A second driving set called "multiple" is implemented but not fully checked out and may or may not provide valid results. To investigate the "multiple" driving set, it must be selected in `BoundaryDatabase::createDrivingSets` followed by recompilation of OpenParEM3D. A third driving set called "single-ended" is discussed, but it is not coded.

3.4.1 Driving set `single`

The `single` driving set drives each S-port in sequence with 2D ABCs applied to the non-driven ports. This is the setup used in Sec. 3.3. An example 3-port setup showing one port driven with 2D ABCs applied to the remaining two ports is shown in Fig. 2.

The weights of the modes at each port are either 0 or 1. When the weight is 1, the field from the 2D solution is imposed onto the port and drives energy into the 3D space. When the weight is 0, the 2D ABC is in effect. When there are N S-ports, the complete `single` driving set in matrix form looks like

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{40}$$

showing one driven S-port at a time.

3.4.2 Driving set `multiple`

When there are two or more modes on a physical port, it can be convenient to solve the S-parameters driving more than one S-port at a time to generate plots with different field configurations. Consider for example a straight section of a differential pair, where there is an even mode and an odd mode at each physical port. Using the `single` driving set, the port is driving with the even mode for a first 3D solve and then the odd mode for a second 3D solve, then the S-parameters are computed. The fields generated from the 3D solves are available for viewing in ParaView [4] (set `project.save.fields` to `true`), and they show the 3D fields when driving either the even mode or the odd mode. However, what if the 3D fields need to be viewed driving one line or the other of the differential pair as single-ended lines? To obtain that field configuration requires driving the even and odd modes simultaneously. To drive one line, the port must be

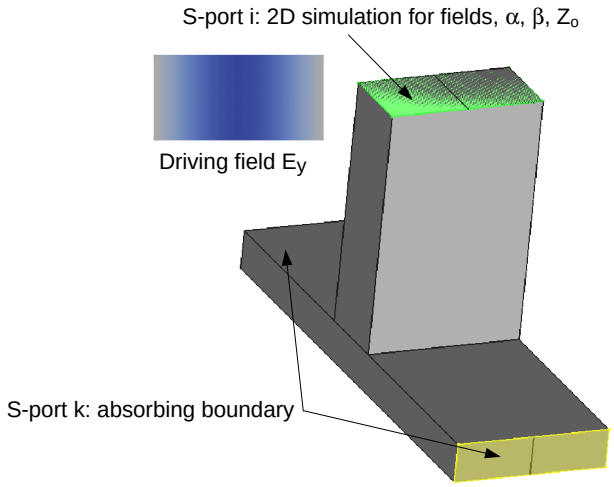


Figure 2: S-parameter S-port setup for waveguide T junction.

driven with the even *plus* the odd mode, while to drive the other line, the even *minus* the odd mode must be driven.

The `multiple` driving set simultaneously drives all modes at a given port. All modes are driven with a weight of 1 with the exception that $N-1$ modes are driven in sequence with a weight of -1. For the straight section of differential pair, the complete `multiple` driving set looks like

$$\begin{bmatrix} +1 & +1 & 0 & 0 \\ +1 & -1 & 0 & 0 \\ 0 & 0 & +1 & +1 \\ 0 & 0 & +1 & -1 \end{bmatrix}, \tag{41}$$

where the columns show the weights for a given S-port and the rows show the 3D simulation. The columns in order are port 1, S-port 1, even mode; port 1, S-port 2, odd mode; port 2, S-port 3, even mode; and, port 2, S-port 4, odd mode. At port 1, the even and odd modes are driven simultaneously in phase in the first simulation then out-of-phase in the second. In the third simulation, the even and odd modes at port 2 are driven in-phase, then they are driven out-of-phase in the fourth simulation.

Note that the final computed S-parameters are still modal S-parameters that must be converted to obtain either single-ended or mixed-mode form. Other than run-to-run numerical differences, the S-parameters produced using driving sets `single` and `multiple` are the same. As described above, the 3D field plots are different.

The `multiple` driving set will produce non-physical S-parameters if a transmission line loops back to the same physical port. In that case, the driving set will drive both ends of the same transmission line. There is no check within OpenParEM3D for this condition, so if there is any question, simply use the `single` driving set.

The `multiple` driving set is coded but not fully checked out. To investigate the "multiple" driving set, it must be selected in `BoundaryDatabase::createDrivingSets` followed by recompilation of OpenParEM3D. One final note is that the `multiple` driving set collapses to the `single` driving set if there is just one mode per port.

3.4.3 Driving set `single-ended`

The `single-ended` driving set is a generalization of the `multiple` driving set, also applicable only when there are two or more modes on a physical port. The difference is that the `multiple` driving set uses fixed weights of 1 and -1, while the `single-ended` driving set uses variable weights taken from the T_v

matrix supplied by OpenParEM2D and stored in the class `Mode`. The fixed weights are sufficient to create single-ended plots for symmetric differential pairs, but for hybrid modes, it is necessary to use T_v. The `single-ended` driving set is not coded.

3.5 Renormalization

The S-parameter matrix is computed unnormalized, meaning that the S-parameters are referenced to the characteristic impedance at each port. When evaluating performance by reviewing and/or plotting S-parameters, unnormalized S-parameters are often preferred since reflections at the ports are not present. It is analogous to taking a measurement with a custom vector network analyzer (VNA) with each port custom matched to the device under test (DUT). When using S-parameters with a circuit simulator, it is generally best practice to renormalize the S-parameters to a single impedance, with 50 Ω being a typical value.

Two different calculations are used to renormalize S-parameters. For modal setups, no combinations or recombinations across ports are required, so a simple calculation to and from the impedance matrix can be used. For line setups, renormalization happens during the process of conversion to single-ended S-parameters.

3.5.1 Modal Setup

For a modal setup, the computed S-parameters are renormalized to a given reference impedance $Z_\circ$, and the results are still modal S-parameters. The renormalization process first converts a $\overline{\overline{S}}$ to $\overline{\overline{Z}}$ using eq. (4.14) from [5]

$$\overline{\overline{Z}} = \overline{\overline{k}} \left(\overline{\overline{I}} + \overline{\overline{S}} \right) \left(\overline{\overline{I}} - \overline{\overline{S}} \right)^{-1} \overline{\overline{k}}, \tag{42}$$

where

$$\overline{\overline{k}} = \begin{bmatrix} \sqrt{Z_{\circ 1}} & 0 & \cdots & 0 \\ 0 & \sqrt{Z_{\circ 2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{Z_{\circ N}} \end{bmatrix}. \tag{43}$$

Then $\overline{\overline{Z}}$ is converted back to $\overline{\overline{S}}$ using the new port impedance $Z_\circ$, typically 50 Ω, using eq. (4.9) in [5]

$$\overline{\overline{S}} = \left(\overline{\overline{Z}} + Z_\circ \overline{\overline{I}} \right)^{-1} \left(\overline{\overline{Z}} - Z_\circ \overline{\overline{I}} \right). \tag{44}$$

The calculation takes place in the method `ResultDatabase::renormalize`.

3.5.2 Line Setup

For a line setup, renormalization occurs during the process of converting the modal S-parameters to single-ended S-parameters using the method in eq. (15) from [6], repeated here as

$$\overline{\overline{S}}_B = \left(\overline{\overline{M}}_C + \overline{\overline{M}}_S \overline{\overline{S}}_A \right) \left(\overline{\overline{M}}_S + \overline{\overline{M}}_C \overline{\overline{S}}_A \right)^{-1} \tag{45}$$

which converts $\overline{\overline{S}}_A$ to $\overline{\overline{S}}_B$ given $\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$. In OpenParEM3D, $\overline{\overline{S}}_A$ is the S-parameter matrix calculated using modal fields, $\overline{\overline{S}}_B$ is the S-parameter matrix for single-ended S-ports, and $\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$ are conversion matrices that must be supplied. The key to the conversion is the definition of the matrices $\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$, which are constructed in the method `fem3D::build_Mc_Ms`.

$\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$ are given by eq. (13) from [6], repeated here as

$$\overline{\overline{M}}_C = \frac{1}{2} \left(\overline{\overline{Z}}_B \right)^{-\frac{1}{2}} \overline{\overline{K}}_v \left(\overline{\overline{Z}}_A \right)^{\frac{1}{2}} - \frac{1}{2} \left(\overline{\overline{Z}}_B \right)^{\frac{1}{2}} \overline{\overline{K}}_i \left(\overline{\overline{Z}}_A \right)^{-\frac{1}{2}} \tag{46}$$

and

$$\overline{\overline{M}}_S = \frac{1}{2} \left(\overline{\overline{Z}}_B \right)^{-\frac{1}{2}} \overline{\overline{K}}_v \left(\overline{\overline{Z}}_A \right)^{\frac{1}{2}} + \frac{1}{2} \left(\overline{\overline{Z}}_B \right)^{\frac{1}{2}} \overline{\overline{K}}_i \left(\overline{\overline{Z}}_A \right)^{-\frac{1}{2}}, \tag{47}$$

where $\overline{\overline{Z}}_A$ is a diagonal matrix holding the S-port modal characteristic impedances, $\overline{\overline{Z}}_B$ is a diagonal matrix holding the reference single-ended characteristic impedance, typically 50 Ω on the entire diagonal, and $\overline{\overline{K}}_v$ and $\overline{\overline{K}}_i$ are dense matrices linking the modal voltages and currents to the single-ended voltages and currents.

OpenParEM2D provides the modal characteristic impedances for $\overline{\overline{Z}}_A$ and the weights for $\overline{\overline{K}}_v$ and $\overline{\overline{K}}_i$. Refer to "OpenParEM2D_Theory_Methodology_Accuracy.pdf" for details of how these are calculated. With the needed matrices filled out, the conversion from modal S-parameters to single-ended S-parameters is made using (45) in the method `ResultDatabase::SparameterConversion`.

3.6 Mixed-Mode Conversion

In systems utilizing differential pairs, it is preferred to plot S-parameters as mixed-mode differential and common-mode signals. The method in [6], eq. (15), is used to convert from single-ended S-parameters to mixed-mode S-parameters, where `DifferentialPair/EndDifferentialPair` blocks indicate differential pairs in the port specification file.

The starting point is a set of single-ended S-parameters (i.e. a line setup with renomalization). To apply (45), $\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$ are built in `fem3D::build_Mc_Ms`, where $\overline{\overline{K}}_v$ and $\overline{\overline{K}}_i$ indicate the differential pairs. For the common-mode voltage, the appropriate row and column entries for the single-ended lines of $\overline{\overline{K}}_v$ are filled with 0.5, while for the common-mode current, $\overline{\overline{K}}_i$ the entries are filled with 1. For the differential-mode voltage, the entries are $+1$ and -1, while for the differential-mode currents, the entries are $+0.5$ and -0.5. Any port remaining as a single-ended line just has the diagonal entry set to 1.

Consider a 2-line symmetric interconnect described by a 4-port single-ended S-parameter matrix, where ports 1 and 2 are at the near end, ports 3 and 4 are at the far end, port 1 is wired to port 3, and port 2 is wired to port 4. To convert the single-ended S-parameter matrix to a mixed-mode S-parameter matrix, $\overline{\overline{K}}_v$ and $\overline{\overline{K}}_i$ are constructed as

$$\overline{\overline{K}}_v = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{48}$$

and

$$\overline{\overline{K}}_i = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0.5 & -0.5 \end{bmatrix}. \tag{49}$$

For each matrix, the first row constructs the common mode for single-ended ports 1 and 2, the second row the differential mode for ports 1 and 2, the third row the common mode for ports 3 and 4, and the fourth row the differential mode for ports 3 and 4.

The single-ended S-parameters are assumed to be normalized to reference impedance $Z_\circ$. The impedance matrices for $\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$ are constructed with $\overline{\overline{K}}_A$ using the single-ended reference impedance and $\overline{\overline{K}}_B$ using $\frac{1}{2}Z_\circ$ for the common-mode impedances and $2Z_\circ$ for the differential mode impedances.

With $\overline{\overline{K}}_v$, $\overline{\overline{K}}_i$, $\overline{\overline{K}}_A$, and $\overline{\overline{K}}_B$ constructed, then $\overline{\overline{M}}_C$ and $\overline{\overline{M}}_S$ can be constructed and finally the mixed-mode S-parameter computed using (45) in the method `ResultDatabase::SparameterConversion`.

Note that OpenParEM3D fundamentally calculates modal S-parameters before optionally making conversions to single-ended S-parameters and then further to mixed-mode S-parameters. For suitable symmetric setups, the modal S-parameters are already in the mixed-mode form. The difference is that the modal S-parameters can be left unnormalized, while the mixed-mode S-parameters are always normalized.

4 Adaptive Mesh Refinement

The adaptive mesh refinement (AMR) methodology is a modified version of the method used in the MFEM Tesla Mini Application, which uses the MFEM method `mfem::L2ZZErrorEstimator`. For use here,

this method is modified to change the preconditioner, check for convergence, and return the error condition instead of the global error. The modified version is OPEM_L2ZZErrorEstimator in the file OPEM_L2ZZErrorEstimator.cpp located in the OpenParEM3D source directory.

Mesh errors are calculated in the method `fem3D::calculateMeshErrors`. An MFEM CurlCurlIntegrator is used to calculate a flux and a smoothed flux, and the difference between the flux and the smoothed flux indicates the error in each mesh element. The CurlCurlIntegrator captures half of the terms in the wave equation in (5), so the full set of physics are not taken into account in calculating the mesh error. A potential area of improvement for AMR in OpenParEM3D is to write a custom integrator that fully captures the wave equation for more accurate error estimates.

For each driven S-port, the 3D $\overline{H}$ is calculated and the mesh errors are calculated for the real part of the field followed by a second calculation for the imaginary part. The magnitude of the complex error per mesh element is merged with errors from prior driven ports so that the adaptive mesh refinement at each iteration takes into account all ports being driven.

Separately handling the real and imaginary parts costs 2× the calculation time, which is a significant part of the overall simulation time in an AMR loop. In fact, as the finite element order increases, the mesh error calculation time becomes comparable to and then exceeds the actual field solve time. A potential area of improvement for AMR in OpenParEM3D is to re-write the error calculation using complex math to target a 2× reduction in error calculation run time. Otherwise, an upgraded mesh error calculation is needed to significantly reduce run times for higher-order finite elements.

Once all ports are driven, the consolidated list of mesh errors are sorted and the mesh elements with the highest errors are targeted for refinement using the MFEM method `mfem::Mesh::GeneralRefinement` in the method `fem3D::refineMesh`. The mesh is refined and convergence criteria are applied to decide whether or not to continue with additional refinement.

Note that AMR uses $\overline{H}$ instead of $\overline{E}$. Since $\overline{H}$ is computed from $\overline{E}$ using (22), errors in $\overline{E}$ are magnified when calculating $\overline{H}$. $\overline{H}$ provide a much better indication of mesh elements needing refinement than $\overline{E}$. Notes in `fem3D::calculateMeshErrors` detail how to have AMR use $\overline{E}$ instead of $\overline{H}$, should that be needed.

4.1 Convergence Testing

Convergence testing relies on the absolute error calculated on $\overline{H}$ using `fem3D::calculateMeshErrors` and on the relative error calculated on the S-parameter matrices $\overline{\overline{S}}_N$ and $\overline{\overline{S}}_{N-1}$ from the N^{th} and $(N-1)^{\text{th}}$ iterations using the error metric

$$\text{error} = \text{max column norm}\left(\overline{\overline{S}}_N^{-1}\left(\overline{\overline{S}}_N - \overline{\overline{S}}_{N-1}\right)\right) \tag{50}$$

in the method `ResultDatabase::calculate_maxRelativeError`. See "OpenParEM3D_Users_Manual.pdf" for a discussion on how to set up convergence using these two metrics.

4.2 AMR Improvement

Measured in terms of run time, the effectiveness of AMR plays an out-sized role in the performance of OpenParEM3D. Mesh refinements at poor locations cost run time without improving accuracy. Efforts to improve run time performance of OpenParEM3D would likely be best directed at improving AMR over other areas. In addition to the potential areas for improvement discussed above, two example sections of code in `fem3D::calculateMeshErrors` are commented out that show how to work with mesh errors to try out ideas on improving AMR.

5 Data Structures and Algorithm Notes

5.1 Data Structures

At the level of `main` in OpenParEM3D.cpp, the primary data structures are defined and listed in Table 1. The bulk of functionality is implemented with the class `BoundaryDatabase`, which contains a list of objects

Table 1: Primary Data Structures

Class	Variable	Function
BoundaryDatabase	boundaryDatabase	Boundary and port definitions except for 2D meshes
FrequencyPlan	frequencyPlan	Frequency plan for refinement sequence and solution frequencies
MeshMaterialList	meshMaterials	Materials used within a mesh
MaterialDatabase	materialDatabase	Material specifications
ResultDatabase	resultDatabase	Computed results
GammaDatabase	gammaDatabase	Port complex propagation constants for use as initial guesses during AMR

from classes `Port` and `Boundary` along with support information such as paths of class `Path` for outlines and integration paths. Objects of class `Port` enable one or modes of class `Mode` to be defined on the port, which ultimately become S-ports.

Computed results are stored using the class `ResultDatabase`, and various post-processing steps are applied within the class to generate S-parameter matrices, renormalized S-parameters, write Touchstone files, etc.

5.2 Boundary and Port Identification and Port Meshes

Boundaries and ports are marked in the 3D mesh in `BoundaryDatabase::markMeshBoundaries`, which applies geometrical checks to see if a boundary mesh element falls within the outline of a boundary or a port, and if so, then marks the boundary mesh element with an attribute linking it to the matching boundary or port.

Attributes are used to identify sections of the mesh on which to apply mathematical operations. For example, `Boundary::addImpedanceIntegrator` implements impedance boundary conditions by using the MFEM method `mfem::BiLinearForm:AddBoundaryIntegrator` restricted to boundary elements marked by a given attribute.

Similarly, `Port::extract2Dmesh` uses attributes assigned to boundary elements forming ports to extract the 2D mesh of the port using `ParSubMesh::CreateFromBoundary`. The 2D meshes are exported to OpenParEM2D to solve transmission lines and waveguides for port fields, complex propagation constants, and characteristic impedances of dominant and optionally higher-order modes.

5.3 Port Fields

S-port fields from OpenParEM2D require considerable infrastructure to ensure that fields are properly oriented. OpenParEM3D defines the normal pointing outwards from the 3D space to be positive, but MFEM may have the normal direction pointing inward or outward. To ensure that fields are properly oriented, the 2D fields are stored in multiple 2D and 3D forms in class `FieldSet` along with a boolean flag in class `Port` called `spin180degrees`, which indicates whether the fields need to be flipped. Comments in class `FieldSet` document the various field storage spaces. The 2D fields in their various states can be viewed by setting `debug.save.port.fields true` in the project setup file [see "OpenParEM3D_Users_Manual.pdf"].

If viewing the 2D fields in the various configurations using `debug.save.port.fields true`, there is an important point to consider. The original 2D fields exist in two finite element spaces: tangential fields using Nedelec elements and longitudinal fields using H1 elements. When these fields are projected onto the ports in the 3D space, the fields appear slightly corrupted because the 3D space only uses Nedelec finite elements, which only have tangential components on the ports. There is insufficient information for the 3D plot to accurately show the normal components of the 2D fields. In this case, the plot exists to check that the field is imported with the correct orientation and not to verify exact field values. In the 3D solution, the 2D tangential components are applied at the ports, then the full 3D solution ensures that the components normal to the port are correct.

5.4 Parallel Processing with MPI

Parallel processing using MPI is used extensively through calls to MFEM and PETSc, which are both heavily parallelized. Time-consuming number crunching occurs in these libraries, so OpenParEM3D benefits from their expert use of MPI. Otherwise in OpenParEM3D, MPI is sparingly used because of the lack of return on the programming effort. It simply makes no sense to parallelize code that represents a tiny fraction of the overall run time. In code where MPI is used, it is primarily present to simply keep data structures aligned for use with MFEM and PETSc.

When MPI is not coded, then operations are duplicated across all processors. For minor processing, the run-time hit is not significant. For example, S-parameters are calculated across all processors, so duplicating the effort N times on N cores.

There is one exception where MPI is implemented to avoid run-time problems, and that involves disk access. When N cores attempt to read from disk at the exact same time, problems can occur. For example, the reading of the project setup file is parallelized where one core reads and parses the file then sends the results to all other cores. There are no known areas where disk access causes problems with non-parallelized code, but it is possible that a problem area will appear that will need to be parallelized.

6 Accuracy Demonstrations

6.1 Microstrip Bandpass Filter

The microstrip bandpass filter described in [7] is simulated for comparison with the paper's measurement. The project can be found in the OpenParEM3D distribution in `regression/microstrip/filter_study`. The layout is done in FreeCAD [8] following the dimensions from the paper, and the final drawing is shown in Fig. 3. Meshing uses gmsh [9] with all default settings except that the mesh "Element size factor" is set to 0.5 to reduce the number of large elements. The starting mesh before adaptive refinement is shown in Fig. 4.

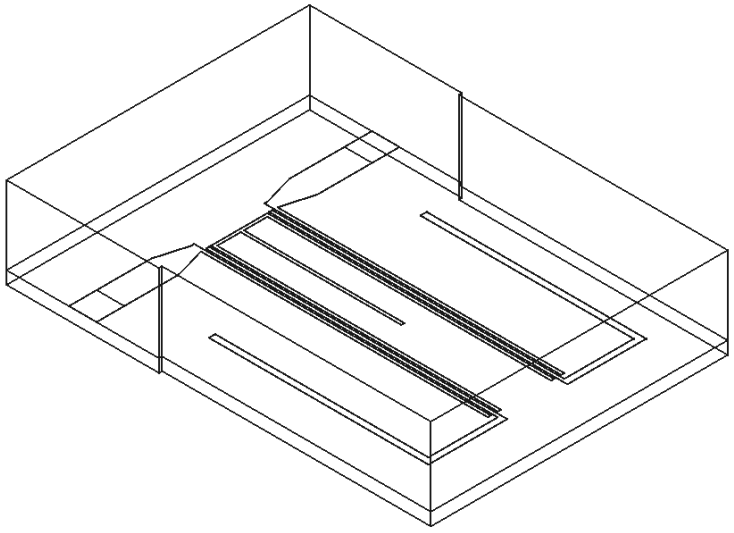


Figure 3: Microstrip filter drawing in FreeCAD.

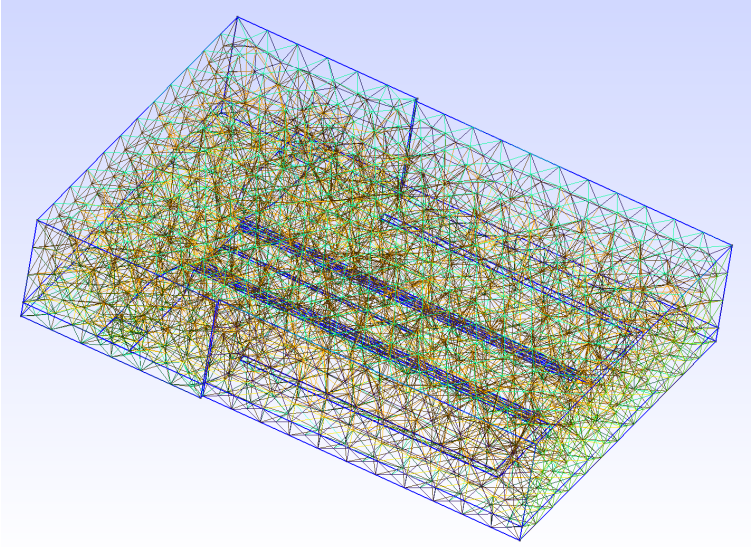


Figure 4: Coarse starting mesh before adaptive refinement.

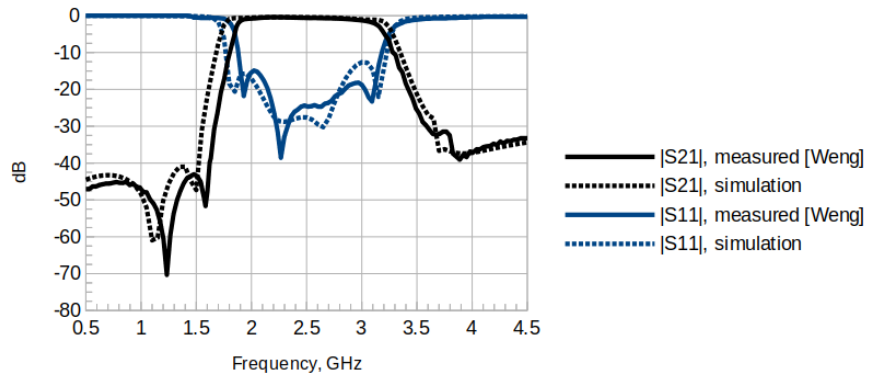
The simulation uses finite elements of order 1 through 3 and adaptive mesh refinement at 4 then 3 GHz using SandH refinement with reltol=0.1 and abstol=0.3e-06 for 1st order, 0.7e-06 for 2nd order, and 1e-06 for 3rd order to achieve similar run times. In short, the simulations are roughly normalized by the resource of time, with the simulations taking 28,733 s, 23,239 s, and 26,563 s for 1st, 2nd, and 3rd order, respectively. Simulation results are compared to the measurements in [7] in Fig. 5 and Fig. 6, and the agreement is quite good.

Comparisons of the bandwidth and center frequencies are shown in Table 2, where agreement between measurement and simulation is good for all orders. All three simulations produce slightly higher bandwidths, with 2nd-order and 3rd-order being very similar, while increasing order shifts the center frequency higher. Since the simulations use as-drawn values for dimensions and datasheet values for materials, it is not knowable which simulation is more accurate since they all fall within the manufacturing tolerances of the measured sample.

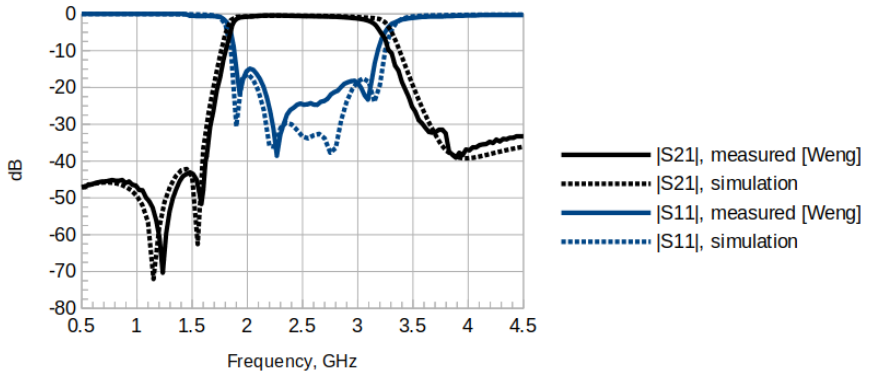
For 3rd-order elements, a slice of the refined mesh at the interface between the substrate and air including the bottom side of the conductor is shown in Fig. 7,

Table 2: Simulation vs. measurement comparisons.

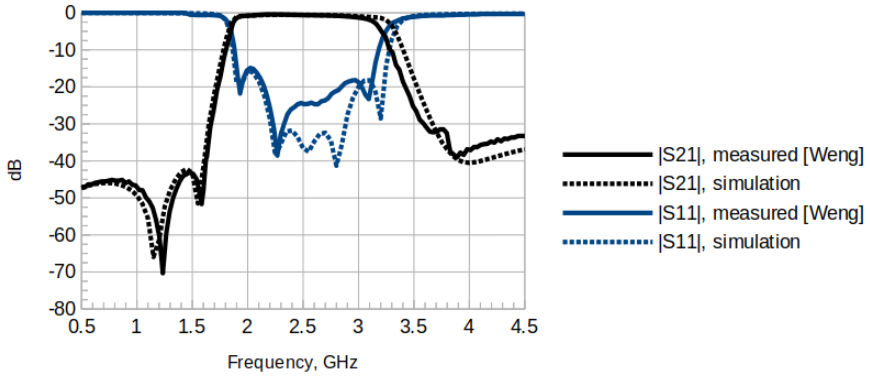
Case	3 dB BW, GHz	f_c, GHz
Measured [7]	1.26	2.51
1st-order	1.50	2.50
2nd-order	1.43	2.55
3rd-order	1.44	2.57



(a) 1st order.

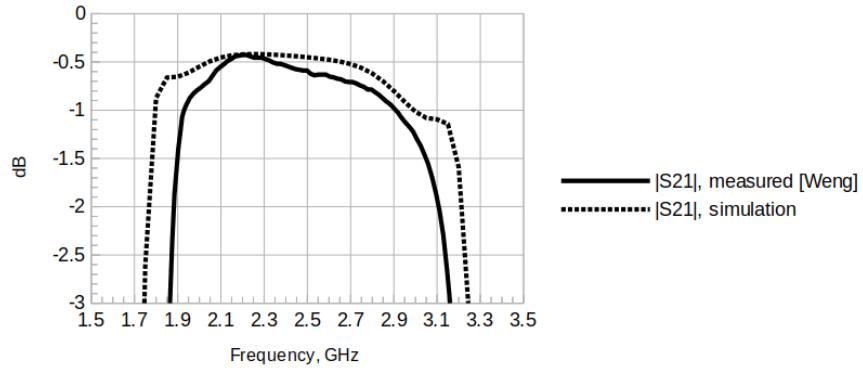


(b) 2nd order.

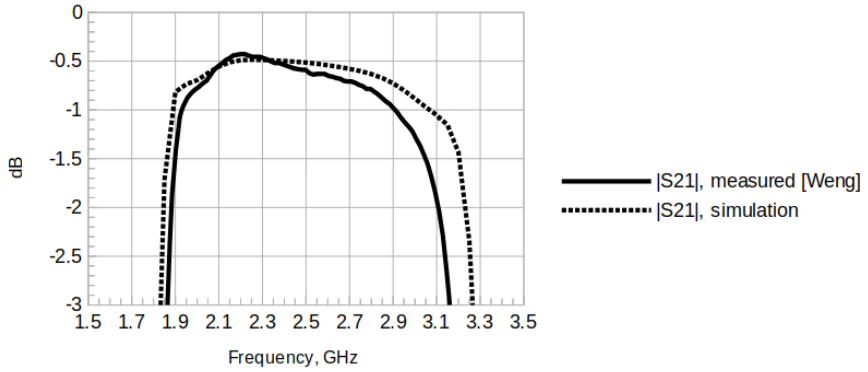


(c) 3rd order.

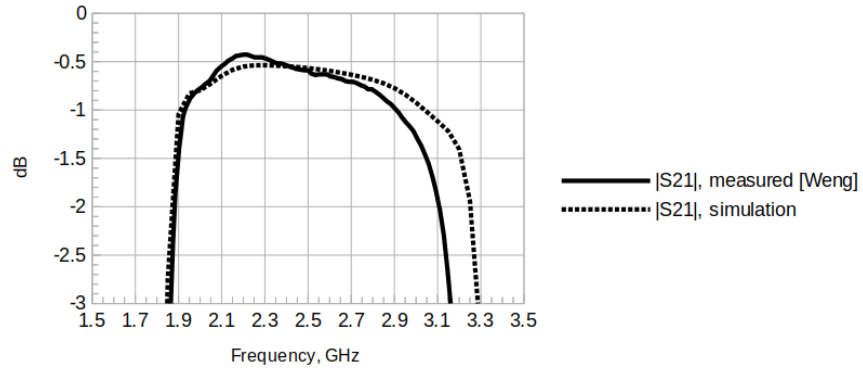
Figure 5: S-parameter simulation results and comparison to experiment from [7].



(a) 1st order.



(b) 2nd order.



(c) 3rd order.

Figure 6: Zoom of S-parameter simulation results and comparison to experiment from [7].

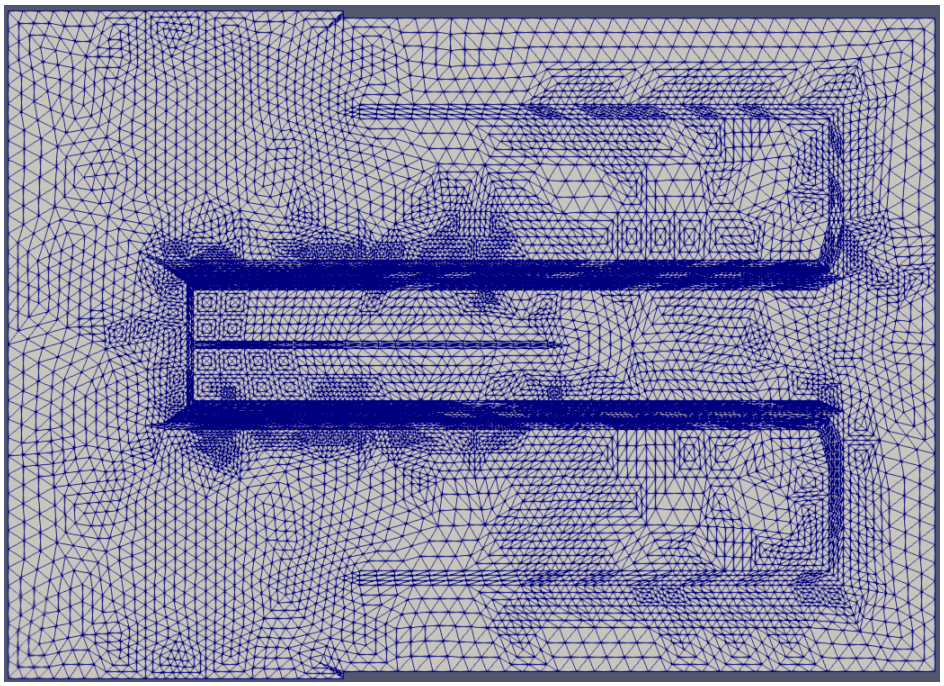


Figure 7: Refined mesh for 3$^{\text{rd}}$-order elements at the substrate/air interface including the bottom side of the metal traces.

where adaptive mesh refinement has added mesh elements to better resolve the traces. The real part of magnitude of the magnetic field at 3 GHz on this mesh is shown in Fig. 8. A more refined plot would result if the abstol convergence parameter were tightened from 1e-06 to 1e-07, but the run time would be greatly increased without a significant change to the S-parameters.

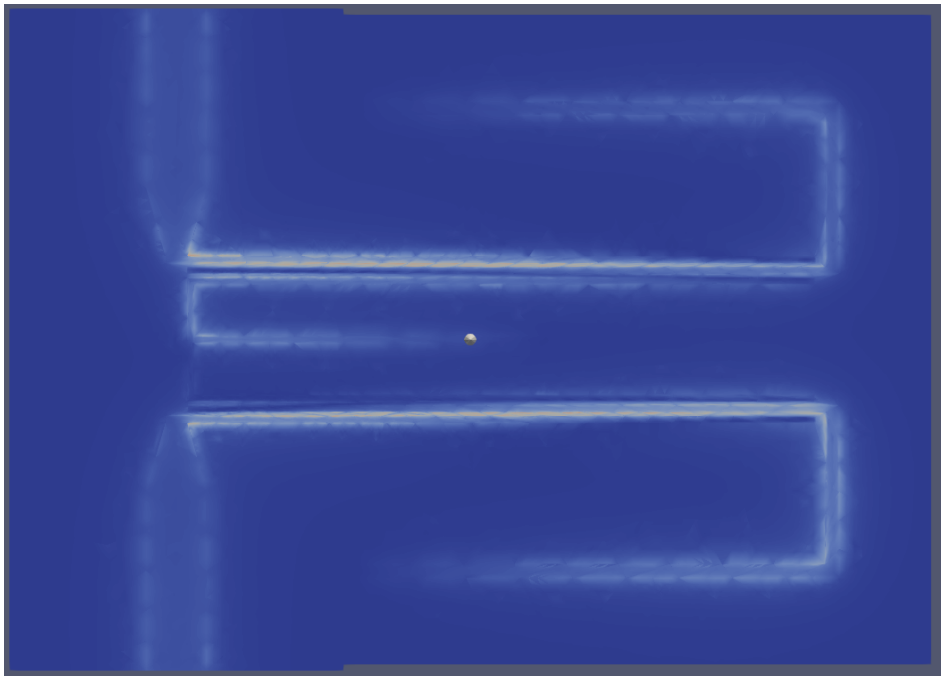
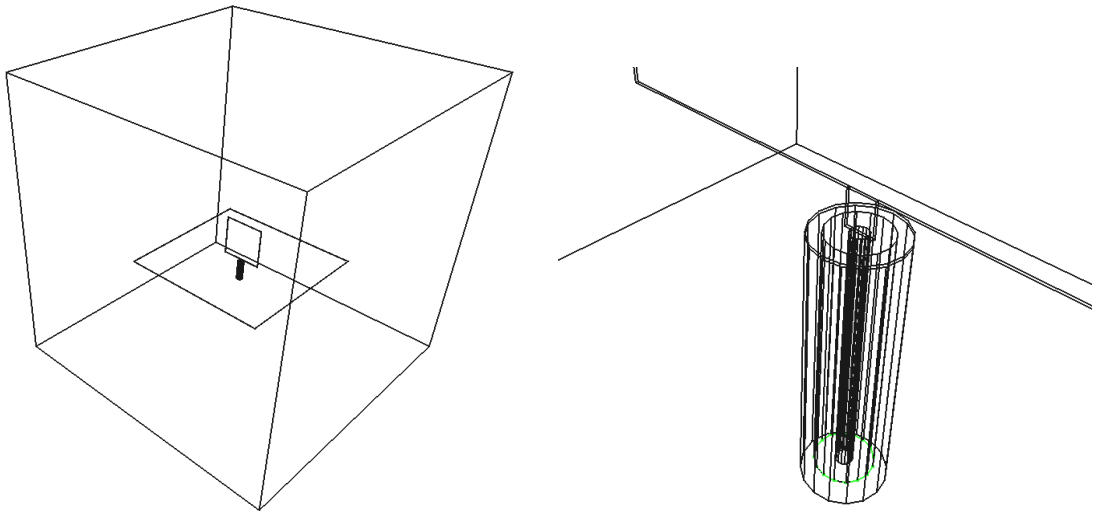


Figure 8: Plot of Re($|\overline{H}|$) at 3 GHz at the substrate/air interface including the bottom side of the metal traces.

6.2 Square Monopole Antenna

The planar monopole antenna described in [10] is simulated for comparison with the paper's measurement. The project can be found in the OpenParEM3D distribution in `regression/antenna/square_monopole_study`. The layout is done in FreeCAD following the dimensions from the paper, and the final drawing is shown in Fig. 9 along with the coaxial feed with the port outlined in green. The paper does not describe how the antenna mounts to the connector, how the connector pin is treated, nor the drilled hole size in the ground plane, so the drawn antenna makes a reasonable guess, but some impact on the results is expected. Meshing uses gmsh with all default settings except that the mesh "Element size factor" is set to 0.5 to reduce the number of large elements.



(a) Square planar monopole drawing in FreeCAD. (b) Zoom of coaxial feed with the port outlined in green.

Figure 9: Square planar monopole drawing in FreeCAD.

The simulation uses finite elements of order 2 with adaptive mesh refinement at 6 then 4 GHz using S refinement with reltol=0.01. The material for the metals is defaulted to brass, and radiation boundary conditions are applied to the outer cube. Simulation results are compared to the measurements in [10] in Fig. 10, and the agreement is good considering the uncertainty of the physical construction at the antenna mounting location.

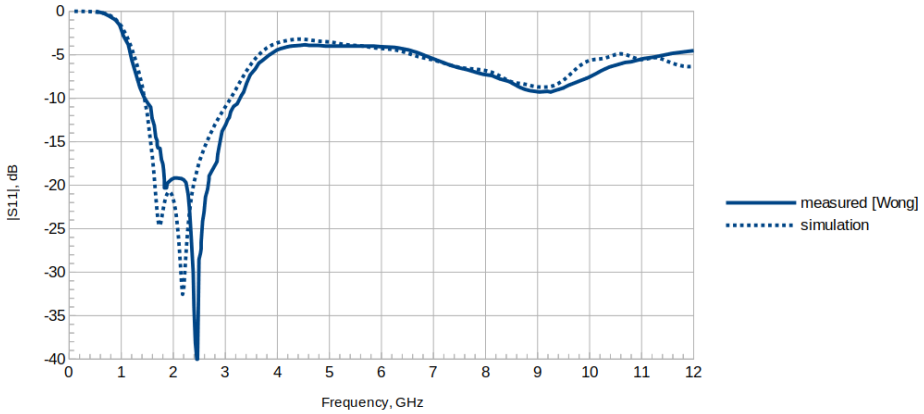


Figure 10: S-parameter simulation results and comparison to experiment from [10].

6.3 Microstrip Bridge

The microstrip bridge described in [11] is simulated for comparison with the paper's simulation. The bridge inserts a break in a microstrip line with the conductor bridging between the two microstrip sections through an area that is uniformly filled with a dielectric taking values of 1, 2.32, 3.78, and 9.8. The project can be found in the OpenParEM3D distribution in `regression/microstrip/bridge_study`. The layout is done in FreeCAD following the dimensions from the paper, and the final drawing is shown in Fig. 11. Meshing uses gmsh with all default settings except that the mesh "Element size factor" is set to 0.5 to reduce the number of large elements.

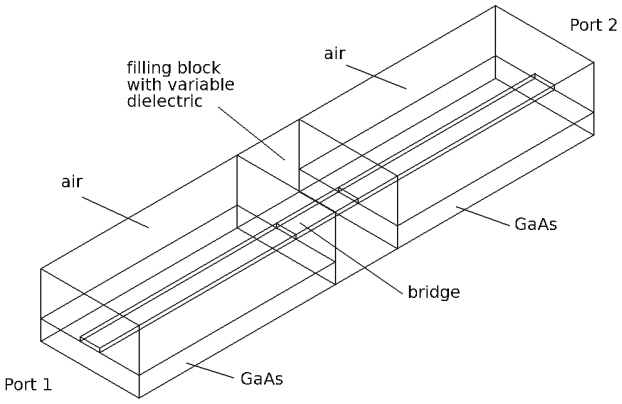


Figure 11: Drawing of a microstrip bridge across a gap with variable dielectric constant filling.

At the highest frequency in the simulation, the longest mesh element in any material is just 0.21λ, enabling 5^{th}-order elements are to be used without adaptive mesh refinement. The mesh used at all frequencies is shown in Fig. 12.

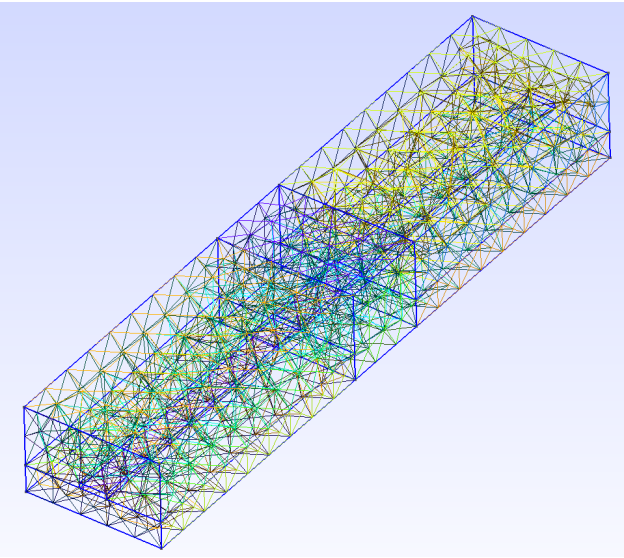


Figure 12: Mesh used for the analysis of the microstrip bridge.

Results are shown in Fig. 13, where the agreement with [11] is excellent. The results here do not show the small level of "waviness" with respect to frequency that reference [11] show. Given the short electrical length of the bridge and its simplicity, there is no physical mechanism to generate the waviness observed in the results of [11], so it is assumed that the waviness an artifact of the simulation in [11].

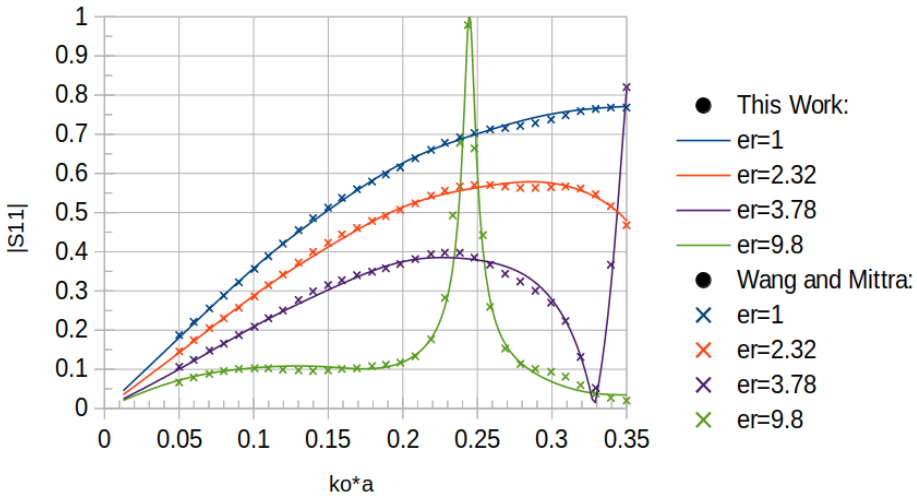


Figure 13: Simulation results with comparison to [11] using 5^{th}-order elements *without* adaptive mesh refinement.

It is interesting to note that the simulation with 5^{th}-order elements supports a 167× range of frequency over a 9.8× range of dielectric constant with the same mesh that has no adaptive mesh refinement. The performance is even achieved for microstrip with finite-thickness metal and sharp edges with no special treatment of the mesh at the edges.

6.4 Slotline Step

The slotline step in width described in [12] is simulated for comparison with the paper's simulation. The project can be found in the OpenParEM3D distribution in `regression/slotline/step_study`. Applying symmetry, half of the structure is drawn in FreeCAD following the dimensions from the paper, and the final drawing is shown in Fig. 14. Meshing uses gmsh with all default settings.

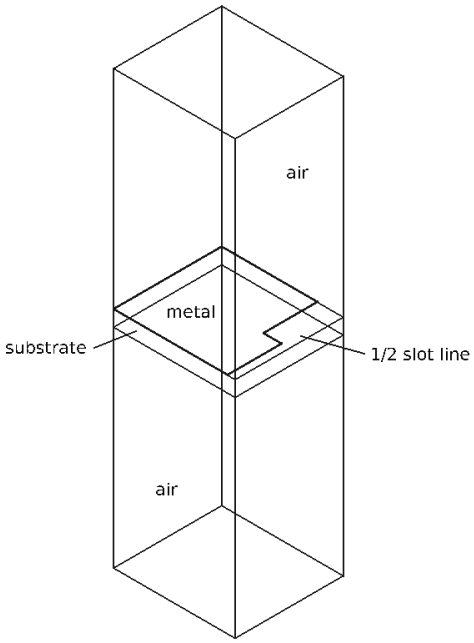


Figure 14: Drawing of 1/2 of a slotline step in width.

The simulation uses 3rd-order elements and adaptive mesh refinement at 35 GHz with convergence on S using a relative tolerance of 0.001. The results and comparison to those of [12] are shown in Fig. 15, where the agreement is excellent. A plot of Re($|\overline{H}|$) on the substrate side of the metal and the slotline gap at 35 GHz when driving the step from the narrow end is shown in Fig. 16.

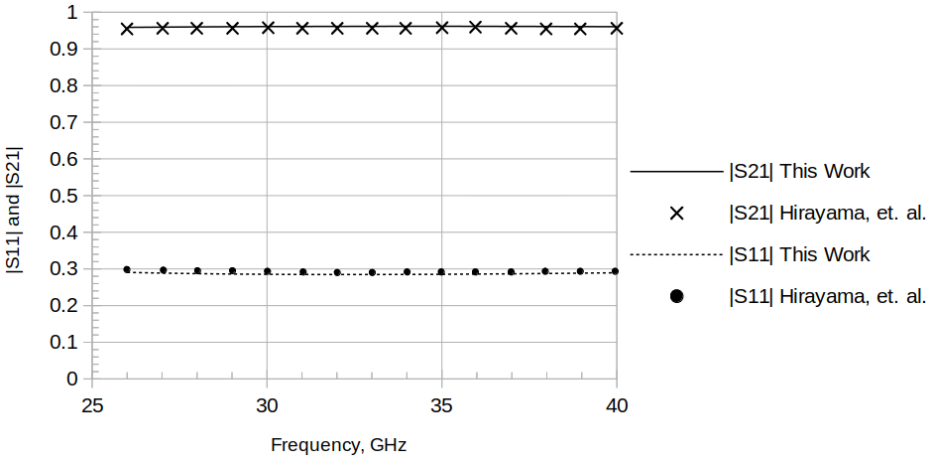


Figure 15: Simulation results with comparison to [12] using 3rd-order elements with adaptive mesh refinement.

Figure 16: Plot of Re($|\overline{H}|$) at 35 GHz driving the narrow end.

6.5 Waveguide T-Junction

The WR75 T-Junction described in [13] is simulated for comparison with the paper's simulation. The project is in the OpenParEM3D distribution in `regression/WR75/T-Junction_study`. The structure is drawn in FreeCAD as shown in Fig. 17. Shown in Fig. 18, the mesh is built using gmsh with all default settings except that the mesh "Element size factor" is set to 0.75.

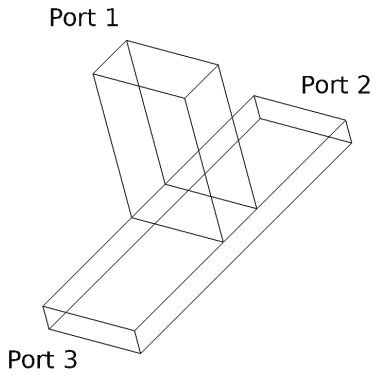


Figure 17: Drawing of a WR75 T-Junction.

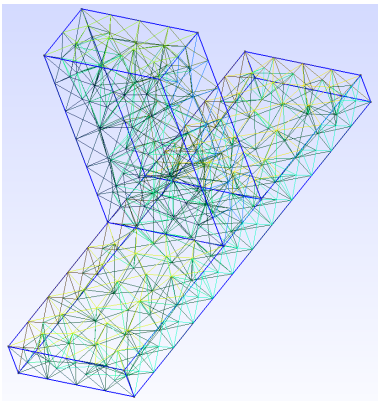
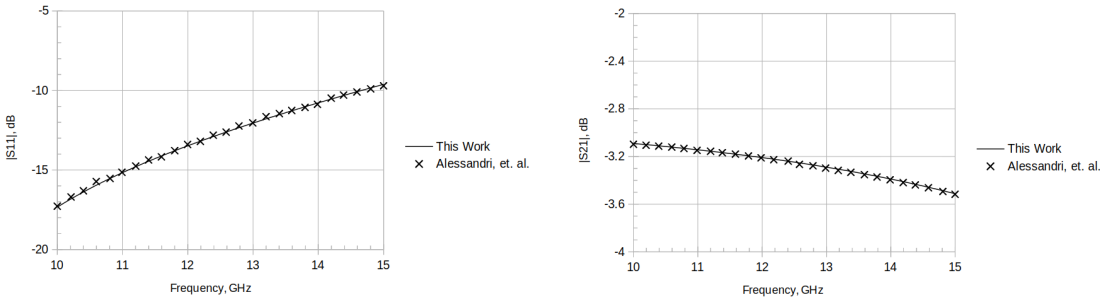


Figure 18: Mesh for the WR75 T-Junction.

The simulation uses 4^{th}-order elements with no adaptive refinement, and the results and comparisons to [13] are shown in Fig. 19. The agreement is excellent.



(a) S11. (b) S21.

Figure 19: Simulation results and comparisons to [13].

6.6 WR75 Puck Discontinuity

The WR75 with dielectric puck discontinuity described in [14] is simulated with results comparison to both [14] and [12]. The project is in the OpenParEM3D distribution in `regression/WR75/dielectric-loading_study`. The structure is drawn in FreeCAD as shown in Fig. 20, and the mesh is built using gmsh with all default settings.

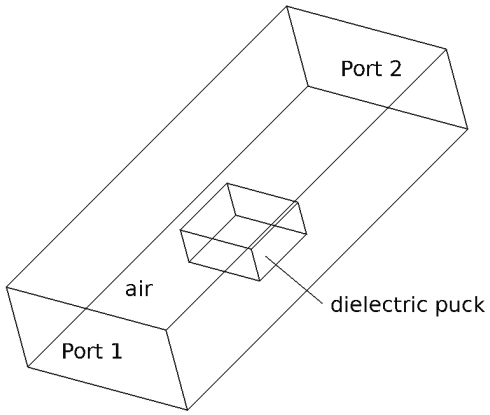


Figure 20: Drawing of a WR75 waveguide loaded by a dielectric puck with $\epsilon_r = 6$.

The structure is simulated once with 4^{th}-order elements *without* adaptive mesh refinement and then again with 2^{nd}-order elements *with* adaptive mesh refinement at 12.5 GHz then 10 GHz using a relative tolerance of 0.01. The results and comparisons are shown in Fig. 21, where the agreement is excellent. The 4^{th}-order and 2^{nd}-order results are almost indistinguishable, and the run times are very similar with less than 0.8% difference.

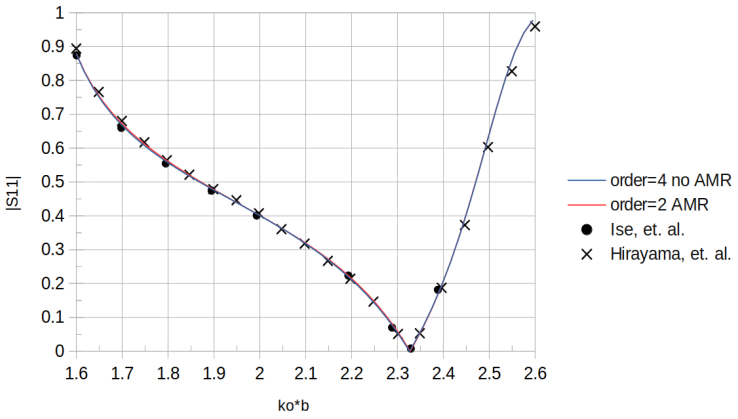


Figure 21: Results and comparisons to [14] and [12].

6.7 Lossy Stripline

The lossy stripline described in [15] is simulated for comparison with the paper's simulation and measurement. The project can be found in the OpenParEM3D distribution in `regression/stripline/Simonovich_stripline_study`. Using the file `regression/stripline/Simonovich_stripline_study/builder.txt`, the structure is generated using builder [see "OpenParEM2D_User_Manual.pdf]. A section just 1 mm long is constructed, and the results are multiplied by 6*25.4 to obtain results for a 6 in line. The structure drawing is shown in Fig. 22, where extra physical detail is automatically added by builder to focus meshing on the trace edges.

The geo file from builder is meshed in gmsh with all default settings except that the mesh "Element size factor" is set to 1.8 to lower the mesh density, and the mesh is shown in Fig. 23. Note that the mesh is quite dense even with the relaxation applied with the element size factor. With a mesh this dense to begin with, low-order finite elements are suitable.

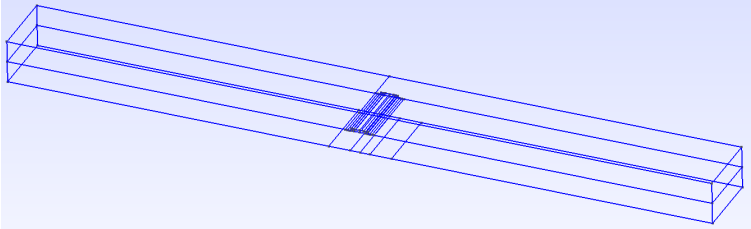


Figure 22: Short 1 mm section of stripline as generated by builder.

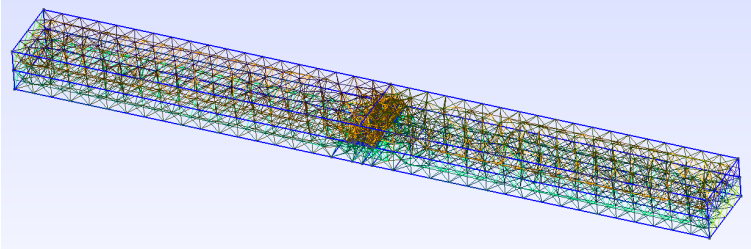


Figure 23: Mesh of the short section of stripline.

The structure is simulated with 2nd-order elements with adaptive refinement at 50 GHz and convergence on S using a relative tolerance of 0.001. The results and comparisons to the simulation and measurement in [15] are shown in Fig. 24. The agreement between both simulations and the measurement are excellent.

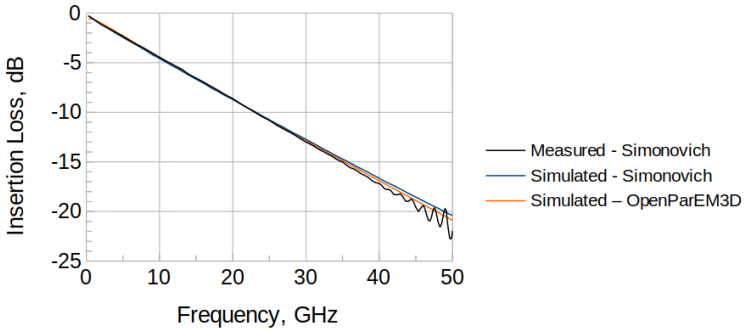


Figure 24: Simulation results and comparisons to [15].

6.8 Lossless WR90 Rectangular Waveguide

A 100 mm-long straight section of lossless WR90 rectangular waveguide is simulated at 9 GHz to extremes to test accuracy, self-consistency, numerical noise floors, and adaptive mesh refinement. The correct answers are known analytically as $|S11| = 0$, $|S21| = 1$, and S21 phase=-20.753057470156°, so exact error calculations can be made. Note that all of the regression suite cases using WR90 rectangular waveguide also make error calculations against analytical results. The project can be found in the OpenParEM3D distribution in `regression/WR90/straight_study`.

The rectangular waveguide is set up in FreeCAD then minimally meshed with gmsh, and the resulting mesh is shown in Fig. 25. At 9 GHz this mesh is very coarse, with the longest tetrahedron edge 1.53λ long. Overall, the waveguide is 2.058λ long as shown by the field plot in Fig. 26.

The waveguide is solved for finite element orders from 1 to 10. Adaptive mesh refinement is allowed to proceed until either 40 iterations complete or until memory runs out. MPI is run with 10 cores. Larger numbers of cores are not possible with such a small initial mesh.

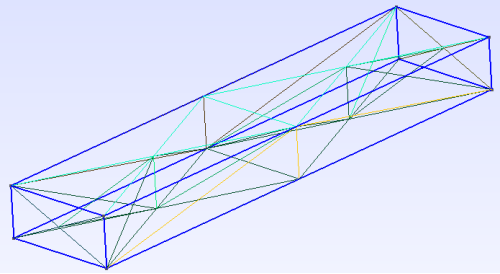


Figure 25: Initial mesh for the 100 mm long WR90 rectangular waveguide.

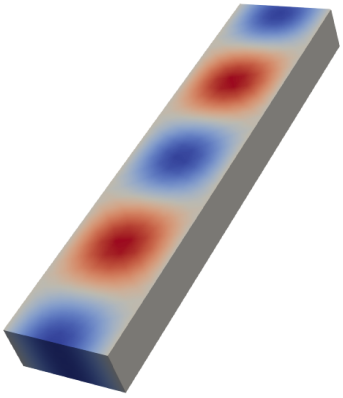
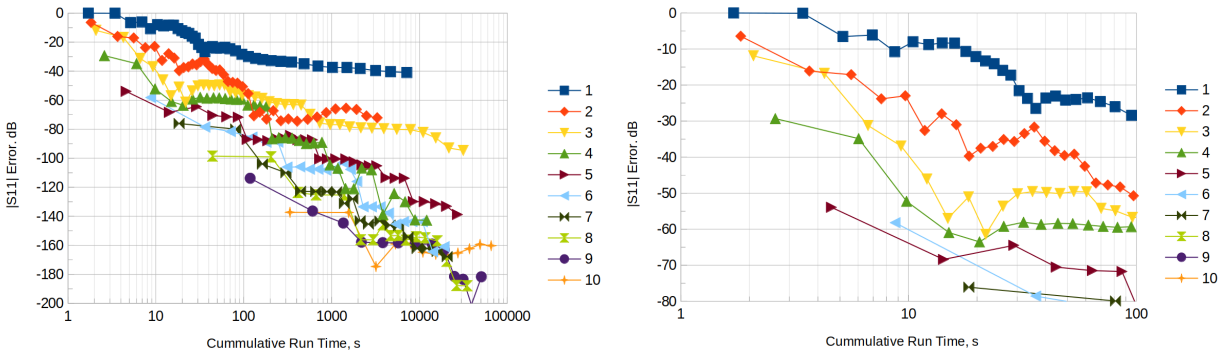


Figure 26: Re(E_y) at 9 GHz.

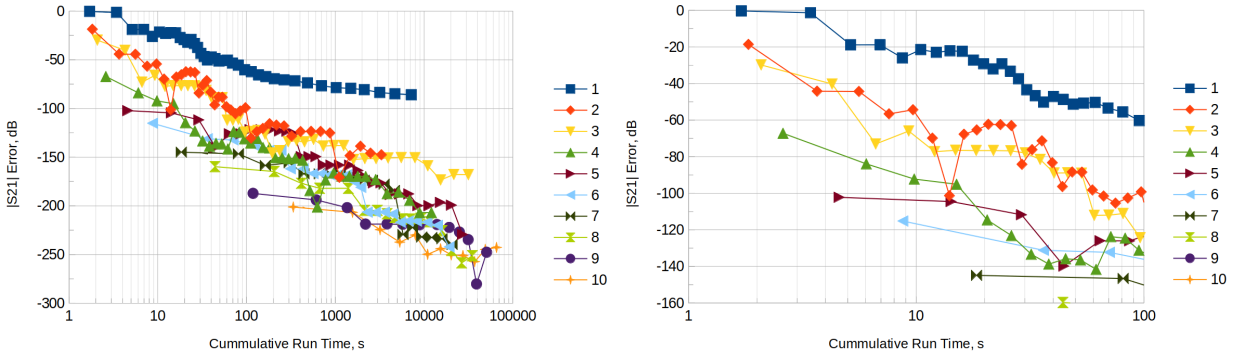
The S11 error is calculated as simply 20*log10(abs(S11)), and the results are plotted in Fig. 27 against the cumulative run time. For all finite element orders, adaptive refinement drives down the errors. The results for orders 8, 9, and 10 imply a noise floor better than -160 dB. For an error of 0.01%, so -40 dB, finite element orders below 5 take some number of iterations to hit this accuracy target, with more iterations required for lower orders. Also for this error target, the shortest run time is achieved with 5^{th}-order finite elements with no adaptive refinement.



(a) Results. (b) Zoom.

Figure 27: |S11| error.

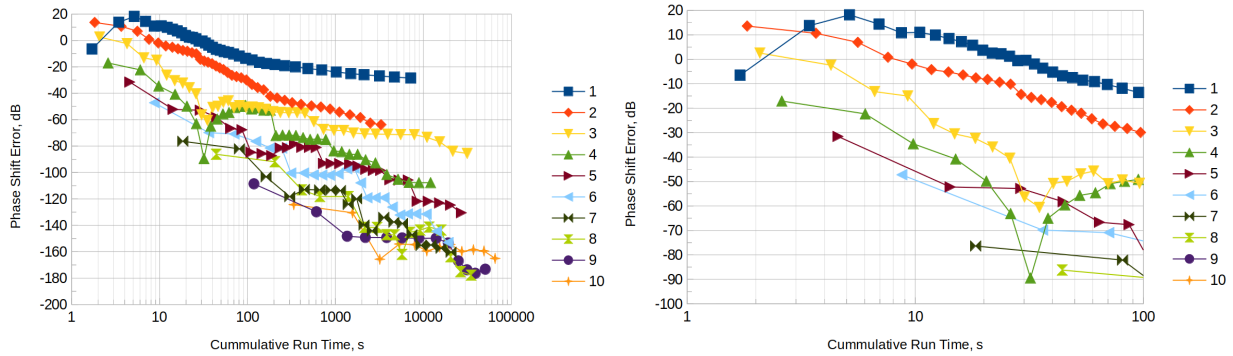
The S21 error is calculated as 20*log10(abs(1-abs(S21)), and the results are plotted in Fig. 28 against the cumulative run time. The errors are lower than for S11 with a lower noise floor, and the adaptive refinement behavior is similar.



(a) Results. (b) Zoom.

Figure 28: |S21| error.

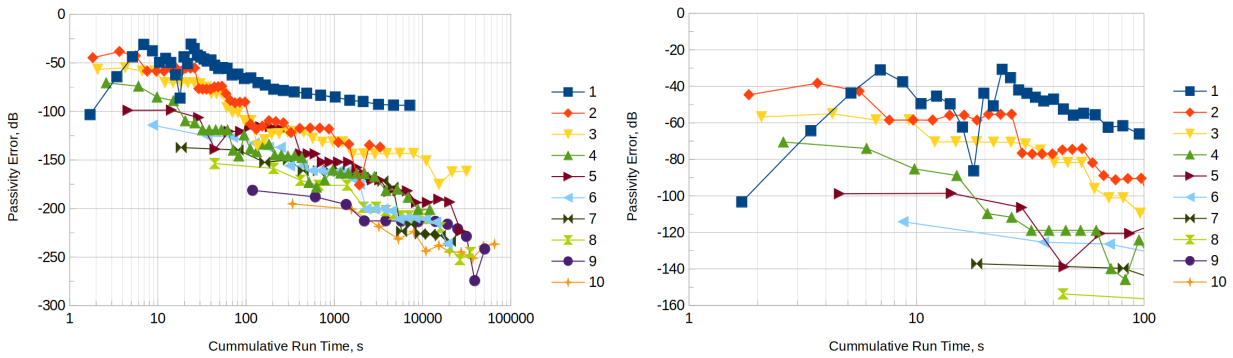
The phase shift error for S21 is calculated as 20*log10(abs((theta-arg(S21)/theta)), where theta=-20.753057470156, and the results are plotted in Fig. 29. The results are higher than for S11 but with a similar noise floor, and the adaptive refinement behavior is similar. For a 0.01% accuracy limit, 1^{st}-order elements are not able to reach the target within a reasonable number of iterations. Orders below 6 require adaptive refinement, and like for S11, higher orders require fewer iterations to reach a given level of error. The shortest run time that achieves this accuracy target is 6^{th}-order with no adaptive refinement. Considering both the S11 and phase shift results, for this problem, the 6^{th}-order element is able to accurately model 1.53λ long finite elements.



(a) Results. (b) Zoom.

Figure 29: Phase shift error.

Since this is a closed lossless problem, the total power in S11 and S21 must sum to 1. The passivity error is calculated as 20*log10(abs(1-abs(S11)^2-abs(S21)^2)), and the results are shown in Fig. 30. The passivity error is good for all orders with higher orders having lower errors and adaptive refinement improving errors for increasing iterations. In general, the passivity error improves as the accuracy improves.



(a) Results. (b) Zoom.

Figure 30: Passivity error.

High-order finite elements offer some opportunity to skip adaptive mesh refinement. In Fig. 31, the S-parameter errors are plotted for orders 1 to 10 for just the first iteration. Again taking 0.01% as an accuracy target, 6^{th}-order and above finite elements can handle the 1.53λ long elements in this problem. Many of the regression suite test cases are solved without adaptive mesh refinement by using higher-order finite elements.

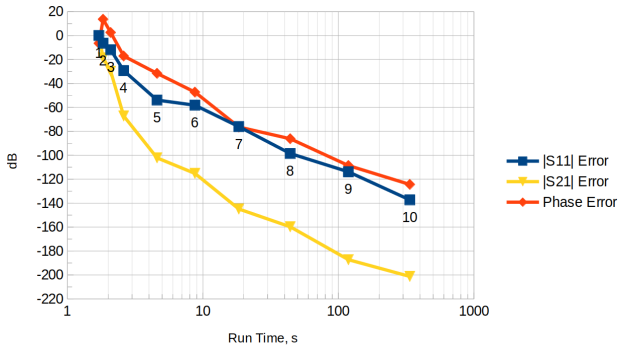


Figure 31: Errors for all finite element orders for just the first iteration.

References

[1] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, Tz. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini, "MFEM: A modular finite element methods library", *Computers and Mathematics with Applications*, vol. 81, 2021, pp. 42-74.

[2] `https://mfem.org`

[3] `https://petsc.org`

[4] `https://www.paraview.org/`

[5] Brian Young, *Digital Signal Integrity: Modeling and Simulation with Interconnects and Packages*, Prentice-Hall PTR, 2001.

[6] Petrie Meyer and David S. Prinsloo, "Generalized multimode scattering parameter and antenna far-field conversions," *IEEE Trans. Antennas and Propagation*, vol. 63, no. 11, Nov. 2015, pp. 4818-4826.

[7] Wei-Chung Weng, "Design and optimization of compact microstrip wideband bandpass filter using Taguchi's method," *IEEE Open Access Journal*, vol. 10, 2022, pp. 107242-107249.

[8] https://www.freecad.org/

[9] https://gmsh.info/

[10] Kin-Lu Wong, Chih-Hsien Wu, and Saou-Wen Su, "Ultrawide-band square planar metal-plate monopole antenna with a trident-shaped feeding strip," *IEEE Trans. Antennas and Propagation*, vol. 53, no. 4, April 2005, pp. 1262-1269.

[11] J.-S. Wang, and R. Mittra, "Finite element analysis of MMIC structures and electronic packages using absorbing boundary conditions," *IEEE Trans. Microwave Theory and Techniques*, vol. 42, no. 3, March 1994, pp. 441-449.

[12] K. Hirayama, Md. Alam, Y. Hayashi, and M. Koshiba, "Vector finite element method with mixed-interpolation-type triangular-prism element for waveguide discontinuities," *IEEE Trans. Microwave Theory and Techniques*, vol. 42, no. 12, Dec. 1994, pp. 2311-2316.

[13] F. Alessandri, M. Dionigi, and R. Rorrentino, "Rigorous analysis of compensated E-plane junctions in rectangular waveguide," *1995 IEEE MTT-S Digest*, pp. 987-990.

[14] K. Ise, K. Inoue, and M. Koshiba, "Three-dimensional finite-element solution of dielectric scattering obstacles in a rectangular waveguide," *IEEE Trans. Microwave Theory and Techniques*, vol. 38, no. 9, Sept. 1990, pp. 1352-1359.

[15] Lambert Simonovich, "A practical method to model effective permittivity and phase delay due to conductor surface roughness", *2017 DesignCon*.