NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nla.782

Parallel numerical solution of the time-harmonic Maxwell equations in mixed form

Dan Li[1], Chen Greif[1,*,†] and Dominik Schötzau[2]

[1]*Department of Computer Science, The University of British Columbia, Vancouver, BC, Canada V6T 1Z4*
[2]*Department of Mathematics, The University of British Columbia, Vancouver, BC, Canada V6T 1Z2*

SUMMARY

We develop a fully scalable parallel implementation of an iterative solver for the time-harmonic Maxwell equations with vanishing wave numbers. We use a mixed finite element discretization on tetrahedral meshes, based on the lowest order Nédélec finite element pair of the first kind. We apply the block diagonal preconditioning approach of Greif and Schötzau (*Numer. Linear Algebra Appl.* 2007; **14**(4):281–297), and use the nodal auxiliary space preconditioning technique of Hiptmair and Xu (*SIAM J. Numer. Anal.* 2007; **45**(6):2483–2509) as the inner iteration for the shifted curl–curl operator. Algebraic multigrid is employed to solve the resulting sequence of discrete elliptic problems. We demonstrate the performance of our parallel solver on problems with constant and variable coefficients. Our numerical results indicate good scalability with the mesh size on uniform, unstructured, and locally refined meshes. Copyright © 2011 John Wiley & Sons, Ltd.

Received 20 June 2010; Revised 3 February 2011; Accepted 21 February 2011

KEY WORDS: parallel iterative solvers; saddle-point linear systems; preconditioners; time-harmonic Maxwell equations

1. INTRODUCTION

Consider the time-harmonic Maxwell equations in mixed form in a lossless medium with perfectly conducting boundaries: find the vector field u and the scalar multiplier p such that

$$\nabla\times\mu_r^{-1}\nabla\times u - k^2\varepsilon_r u + \varepsilon_r\nabla p = f \quad \text{in } \Omega, \tag{1a}$$

$$\nabla\cdot(\varepsilon_r u) = 0 \quad \text{in } \Omega, \tag{1b}$$

$$n_\Gamma \times u = 0 \quad \text{on } \Gamma, \tag{1c}$$

$$p = 0 \quad \text{on } \Gamma. \tag{1d}$$

Here $\Omega \in \mathbb{R}^3$ is a polyhedral domain, which we assume is simply connected with a connected boundary $\Gamma = \partial\Omega$, f is a generic source and n_Γ denotes the outward unit normal on Γ. The electromagnetic parameters μ_r and ε_r denote the relative permeability and permittivity, respectively, which are scalar functions of position, uniformly bounded from above and below:

$$0 < \mu_{\min} \leqslant \mu_r \leqslant \mu_{\max} < \infty \quad \text{and} \quad 0 < \varepsilon_{\min} \leqslant \varepsilon_r \leqslant \varepsilon_{\max} < \infty.$$

*Correspondence to: Chen Greif, Department of Computer Science, The University of British Columbia, Vancouver, BC, Canada V6T 1Z4.
†E-mail: greif@cs.ubc.ca

The wave number k is given by $k^2 = \omega^2 \varepsilon_0 \mu_0$, where $\mu_0 = 4\pi \times 10^{-7}$ (H/m) and $\varepsilon_0 = \frac{1}{36\pi} \times 10^{-9}$(F/m) are the permeability and permittivity in vacuum, respectively, and $\omega \neq 0$ is the angular frequency. We assume throughout that $k^2 \varepsilon_r$ is not a Maxwell eigenvalue.

The mixed formulation is a natural and well-established way of dealing with the high nullity of the curl–curl operator. It yields a stable and well-posed problem for vanishing wave numbers [1–5]. In this paper we thus focus on the case

$$k \ll 1$$

including $k = 0$. The latter case is of much interest in many applications, such as magnetostatics [6]. The mixed formulation readily allows for non-divergence-free data. Large values of k are of extreme importance in wave propagation and other applications, but we have not yet studied our proposed technique for such ranges.

Finite element discretization using Nédélec elements of the first kind [7] for the approximation of u and standard nodal elements for p yields a saddle-point linear system of the form

$$\underbrace{\begin{pmatrix} A - k^2 M & B^{\mathrm{T}} \\ B & 0 \end{pmatrix}}_{\mathscr{K}} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}. \tag{2}$$

The saddle-point matrix $\mathscr{K}$ is of size $(m+n) \times (m+n)$ and is symmetric indefinite. The matrix $A \in \mathbb{R}^{n \times n}$ corresponds to the μ_r^{-1}-weighted discrete curl–curl operator; $B \in \mathbb{R}^{m \times n}$ is the ε_r-weighted divergence operator with full row rank; $M \in \mathbb{R}^{n \times n}$ is the ε_r-weighted vector mass matrix; $f \in \mathbb{R}^n$ is now the load vector associated with the right-hand side in (1a), and the vectors $u \in \mathbb{R}^n$ and $p \in \mathbb{R}^m$ represent the finite element coefficients. Note that A is symmetric positive-semidefinite with nullity m.

The saddle-point matrix problem (2) inherits the properties of the continuous formulation: it is stable in the limit case $k = 0$ and directly deals with the discrete gradients, without a need for further postprocessing.

In [8], Schur complement-free block diagonal preconditioners were designed for iteratively solving system (2) with constant coefficients. These preconditioners are motivated by spectral equivalence properties. Each iteration of the scheme requires inverting a scalar Laplacian and solving a linear system with $A + \gamma M$, where γ is a given positive parameter. There are several efficient solution methods for doing so. When a hierarchy of structured meshes is available, geometric multigrid can be applied [9]; for unstructured meshes, algebraic multigrid (AMG) approaches have been explored in [10–12], using the smoothers introduced in [9]. See also [13, 14] for an analysis of multigrid methods and overlapping Schwarz preconditioners for $A - k^2 M$. Recently, a highly efficient nodal auxiliary space preconditioner has been proposed in [15]; it reduces solving for $A + \gamma M$ into essentially two scalar elliptic problems on the nodal finite element space. In [16], a massive parallel implementation of the nodal auxiliary space preconditioners was developed, which can also deal with the limit case $\gamma = 0$, using a gradient projection approach.

In this paper we extend our work in [8] and develop a fully scalable parallel implementation for efficiently solving (2) in complex domains in three dimensions. The outer iterations are based on the approach in [8], extended to the variable coefficient case, and the inner iterations are solved using the method of Hiptmair and Xu [15]. We use algebraic multigrid solvers for each elliptic problem, and accomplish almost linear complexity in the number of degrees of freedom. For our implementation, we use state-of-the-art software packages (PETSc [17], Hypre [18], and METIS [19]) to optimize the performance of our solvers. We have also developed our own mesher for structured meshes, and we use TetGen [20] for unstructured and locally refined meshes. We present an extensive set of numerical experiments, solving problems with several millions degrees of freedom. Our numerical results scale well with the mesh size on uniform, unstructured, and locally refined meshes.

The remainder of the paper is structured as follows. In Section 2 we analyze the properties of the discrete operators. The preconditioning approach is presented in Section 3. In Section 4 we provide numerical examples to demonstrate the scalability and performance of the proposed solvers. Finally, we draw some conclusions in Section 5.

2. FINITE ELEMENT DISCRETIZATION

To discretize problem (1), we partition the domain Ω into shape-regular tetrahedra of a sufficiently small mesh size h. The electric field is approximated with Nédélec elements of the first family and the multiplier is approximated with nodal elements of order ℓ [4, 7]. We denote the two resulting finite element spaces by V_h and Q_h, respectively. On V_h we enforce the homogeneous boundary condition (1c), whereas on Q_h we impose (1d). Figure 1 shows the degrees of freedom on the lowest order Nédélec elements in 2D and 3D.

Let $\langle\psi_j\rangle_{j=1}^n$ and $\langle\phi_i\rangle_{i=1}^m$ be finite element bases for the spaces V_h and Q_h respectively:

$$V_h = \operatorname{span}\langle\psi_j\rangle_{j=1}^n, \quad Q_h = \operatorname{span}\langle\phi_i\rangle_{i=1}^m. \tag{3}$$

Then, the weak formulation of (1) yields a linear system of the form (2), see [4, 8], where the entries of the matrices and the load vector are given by

$$\begin{aligned}
A_{i,j} &= \int_\Omega \mu_r^{-1}(\nabla\times\psi_j)\cdot(\nabla\times\psi_i)\,\mathrm{d}x, \quad 1\leqslant i,j\leqslant n,\\
M_{i,j} &= \int_\Omega \varepsilon_r\psi_j\cdot\psi_i\,\mathrm{d}x, \quad 1\leqslant i,j\leqslant n,\\
B_{i,j} &= \int_\Omega \varepsilon_r\psi_j\cdot\nabla\phi_i\,\mathrm{d}x, \quad 1\leqslant i\leqslant m, \quad 1\leqslant j\leqslant n,\\
f_i &= \int_\Omega f\cdot\psi_i\,\mathrm{d}x, \quad 1\leqslant i\leqslant n.
\end{aligned}$$

Let us introduce a few additional matrices that play an important role in this formulation. First, note that $\nabla Q_h \subset V_h$, and define the matrix $C\in\mathbb{R}^{n\times m}$ by

$$\nabla\phi_j = \sum_{i=1}^n C_{i,j}\psi_i, \quad j=1,\ldots,m. \tag{4}$$

For a function $q_h\in Q_h$ given by $q_h=\sum_{j=1}^m q_j\phi_j$, we then have

$$\nabla q_h = \sum_{i=1}^n\sum_{j=1}^m C_{i,j}q_j\psi_i,$$

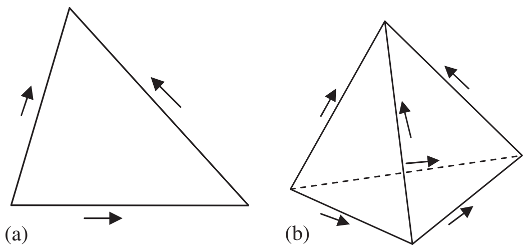

Figure 1. A graphical illustration of the degrees of freedom for the lowest order Nédélec element in 2D and 3D. Degrees of freedom are the average value of tangential component of the vector field on each edge. (a) 2D and (b) 3D.

so Cq is the coefficient vector of ∇q_h in the basis $\langle \psi_i \rangle_{i=1}^n$. In the lowest order case, the entries of C are

$$C_{i,j} = \begin{cases} 1 & \text{if node } j \text{ is the head of edge } i, \\ -1 & \text{if node } j \text{ is the tail of edge } i, \\ 0 & \text{otherwise.} \end{cases}$$

Define the ε_r-weighted scalar Laplacian on Q_h as $L = (L_{i,j})_{i,j=1}^m \in \mathbb{R}^{m \times m}$ with

$$L_{i,j} = \int_\Omega \varepsilon_r \nabla \phi_j \cdot \nabla \phi_i \, \mathrm{d}x. \tag{5}$$

Finally, we set $Q = (Q_{i,j})_{i,j=1}^m \in \mathbb{R}^{m \times m}$ as the ε_r-weighted scalar mass matrix on Q_h, that is

$$Q_{i,j} = \int_\Omega \varepsilon_r \phi_j \cdot \phi_i \, \mathrm{d}x.$$

Let us state a few stability results that extend the analysis in [8] from constant material coefficients to the variable coefficient case.

Denote by $\langle \cdot, \cdot \rangle$ the standard Euclidean inner product in $\mathbb{R}^n$ or $\mathbb{R}^m$, and by null$(\cdot)$ the null space of a matrix. For a given positive-(semi)definite matrix W and a vector x, we define the (semi)norm

$$|x|_W = \sqrt{\langle Wx, x \rangle}.$$

Proposition 2.1
The following stability properties of the matrices A and B hold:

(i) Continuity of A:

$$|\langle Au, v \rangle| \leqslant |u|_A |v|_A, \quad u, v \in \mathbb{R}^n.$$

(ii) Continuity of B:

$$|\langle Bv, q \rangle| \leqslant |v|_M |q|_L, \quad v \in \mathbb{R}^n, \;\; q \in \mathbb{R}^m.$$

(iii) The matrix A is positive definite on null(B) and

$$\langle Au, u \rangle \geqslant \alpha |u|_M^2, \quad u \in \mathrm{null}(B)$$

with a stability constant α which is independent of the mesh size.

(iv) The matrix B satisfies the discrete inf–sup condition

$$\inf_{0 \neq q \in \mathbb{R}^m} \sup_{0 \neq v \in \mathrm{null}(A)} \frac{\langle Bv, q \rangle}{|v|_M |q|_L} \geqslant 1.$$

Proof
The first two properties follow directly from the Cauchy–Schwarz inequality.

To show (iii), we first recall the discrete Poincaré–Friedrichs inequality from [3, Theorem 4.7]. Let $u \in \mathrm{null}(B)$ and let u_h be the associated finite element function. Then, we have

$$\int_\Omega |\nabla \times u_h|^2 \, \mathrm{d}x \geqslant \beta \int_\Omega |u_h|^2 \, \mathrm{d}x,$$

where $\beta > 0$ is independent of the mesh size.

Consequently, we bound $\langle Au, u \rangle$ as follows:

$$\langle Au, u \rangle = \int_\Omega \mu_r^{-1} |\nabla \times u_h|^2 \, \mathrm{d}x \geqslant \frac{\beta}{\mu_{\max} \varepsilon_{\max}} |u|_M^2 = \alpha |u|_M^2,$$

where $\alpha = \beta / (\mu_{\max} \varepsilon_{\max})$.

To prove (iv), let $0 \neq q_h \in Q_h$ and v be the coefficient vector of $v_h = \nabla q_h$ in the basis $\langle \psi_i \rangle_{i=1}^{n}$. Then it follows that $v \in \text{null}(A)$ and

$$\sup_{0 \neq v \in \text{null}(A)} \frac{\langle Bv, q \rangle}{|v|_M |q|_L} = \sup_{0 \neq v \in \text{null}(A)} \frac{\int_\Omega \varepsilon_r v_h \cdot \nabla q_h \, \mathrm{d}x}{\left(\int_\Omega \varepsilon_r v_h \cdot v_h \, \mathrm{d}x\right)^{\frac{1}{2}} |q|_L} \geqslant \frac{|q|_L^2}{|q|_L^2} = 1,$$

which shows (iv). □

The properties stated in Proposition 2.1 and the theory of mixed finite element methods [21, Chapter 2] ensure that the saddle-point system (2) is invertable (provided that the mesh size is sufficiently small).

3. THE SOLVER

To iteratively solve the saddle-point system (2) we use MINRES [22] as an *outer solver*. This is discussed in Section 3.1. To solve each outer iteration, we apply an *inner solver* based on [15] and presented in Section 3.2. In Section 3.3, we outline the complete solution procedure.

3.1. The outer solver

Following the analysis for constant coefficients in [8], we propose the following block diagonal preconditioner to iteratively solve (2):

$$\mathscr{P}_{M,L} = \begin{pmatrix} \mathscr{P}_M & 0 \\ 0 & L \end{pmatrix}, \tag{6}$$

where

$$\mathscr{P}_M = A + \gamma M \tag{7}$$

and

$$\gamma = 1 - k^2 > 0 \tag{8}$$

since $k \ll 1$.

By proceeding as in [8, Theorem 5.2], we immediately have the following result; see [23] for further details.

Theorem 3.1
Suppose $k \ll 1$, the preconditioned matrix $\mathscr{P}_{M,L}^{-1}\mathscr{K}$ has two eigenvalues $\lambda_+ = 1$ and $\lambda_- = -1/(1-k^2)$, each with algebraic multiplicity m. The remaining eigenvalues satisfy the bound

$$\frac{\alpha - k^2}{\alpha + 1 - k^2} < \lambda < 1, \tag{9}$$

where α is the constant in Proposition 2.1 (iii).

3.2. The inner solver

The overall computational cost of using $\mathscr{P}_{M,L}$ depends on the ability to efficiently solve linear systems whose associated matrices are $\mathscr{P}_M$ in (7) and L in (5).

The linear system L arises from a standard scalar elliptic problem, for which many efficient solution methods exist. On the other hand, efficiently inverting $\mathscr{P}_M$ is the computational bottleneck in the inner iteration. Recently, Hiptmair and Xu proposed effective auxiliary space preconditioners for linear systems arising from conforming finite element discretizations of

H(curl)-elliptic variational problems [15], based on fictitious spaces as developed in [24, 25]. The preconditioner is

$$\mathscr{P}_V^{-1} = \mathrm{diag}(\mathscr{P}_M)^{-1} + P(\bar{L} + \gamma\bar{Q})^{-1}P^{\mathrm{T}} + \gamma^{-1}C(L^{-1})C^{\mathrm{T}}, \tag{10}$$

with γ as in (8). The matrix $\bar{L} = \mathrm{diag}(\hat{L}, \hat{L}, \hat{L})$ is the μ_r^{-1}-weighted vector Laplacian on Q_h^3, where $\hat{L}$ denotes the μ_r^{-1}-weighted (rather than ε_r-weighted) version of L in (5). Furthermore, the matrix $\bar{Q} = \mathrm{diag}(Q, Q, Q)$ is the ε_r-weighted vector mass matrix on Q_h^3, C is the null-space matrix in (4), and P is the matrix representation of the nodal interpolation operator $\Pi_h^{\mathrm{curl}} : Q_h^3 \to V_h$. In the lowest order case, the operator Π_h^{curl} is based on the path integrals along edges; for a finite element function $w_h \in Q_h^3$ it is given by

$$\Pi_h^{\mathrm{curl}} w_h = \sum_j \left(\int_{e_j} w_h \cdot \mathrm{d}\vec{s} \right) \psi_j,$$

where e_j is the interior edge associated with the basis function ψ_j. We have $P = [P^{(1)}, P^{(2)}, P^{(3)}]$, where $P \in \mathbb{R}^{n \times 3m}$ and $P^{(k)}$ are matrices in $\mathbb{R}^{n \times m}$. The entries of $P^{(k)}$ are given by

$$P_{i,j}^{(k)} = \begin{cases} 0.5 d_i t_i^{(k)} & \text{if node } j \text{ is the head/tail of edge } i, \\ 0 & \text{otherwise,} \end{cases}$$

where $t_i^{(k)}$ is the kth component of the unit tangential vector on edge i and d_i is the length of edge i.

In the constant coefficient case, it was shown in [15, Theorem 7.1] that for $0 < \gamma \leqslant 1$, the spectral condition number $\kappa_2(\mathscr{P}_V^{-1}\mathscr{P}_M)$ is independent of the mesh size. Although there seems to be no theoretical analysis available for the variable coefficient case, the preconditioner $\mathscr{P}_V$ was experimentally shown to be effective in this case as well [15].

3.3. Solution algorithm

We run preconditioned MINRES as the outer solver for the linear system (2). The preconditioner is the block diagonal matrix $\mathscr{P}_{M,L}$, defined in (6). For each outer iteration, we need to solve a linear system of the form

$$\begin{pmatrix} \mathscr{P}_M & 0 \\ 0 & L \end{pmatrix} \begin{pmatrix} v \\ q \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}. \tag{11}$$

Two Krylov subspace solvers are applied as the inner iterations. The linear system associated with the $(1,1)$ block

$$\mathscr{P}_M v = c \tag{12}$$

is solved using conjugate gradient (CG) with the preconditioner $\mathscr{P}_V$, which is defined in (10). In each CG iteration, we need to solve a linear system of the form

$$\mathscr{P}_V w = r. \tag{13}$$

Following (10), this can be done by solving the two linear systems

$$(\bar{L} + \gamma\bar{Q})y = s, \tag{14a}$$

$$\bar{L}z = t, \tag{14b}$$

where $s = P^{\mathrm{T}}r$ and $t = C^{\mathrm{T}}r$. We run one AMG V-cycle to compute y and z, and we set

$$w = \mathrm{diag}(\mathscr{P}_M)^{-1}r + Py + \gamma^{-1}Cz. \tag{15}$$

Algorithm 1 Solve $\mathscr{K}x=b$; see (2)

```
1: initialize MINRES for (2)
2: while MINRES not converged do
3:   set $c$, $d$ to be the right-hand-side for the current inner iteration; see (11)
4:   initialize CG for (12)
5:   while CG not converged do
6:     run one AMG V-cycle to approximate $(\bar{L}+\gamma\bar{Q})^{-1}$ and update $y$ in (14a)
7:     run one AMG V-cycle to approximate $L^{-1}$ and update $z$ in (14b)
8:     update $w$ in (13), using (15)
9:     update $v$ in (12)
10:  end while
11:  initialize CG for (16)
12:  while CG not converged do
13:    apply AMG preconditioner to approximate $L$
14:    update $q$ in (16)
15:  end while
16:  update $x$ in (2)
17: end while
```

The linear system associated with the $(2,2)$ block of (11)

$$Lq=d \tag{16}$$

is solved using CG with an AMG preconditioner.

Our approach is summarized in Algorithm 1. The inner iteration for (12) is initialized in line 4 and laid out in lines 5–10, where CG iterations preconditioned with $\mathscr{P}_V$ are used. The inner iteration for (16) is initialized in line 11 and provided in lines 12–15, where a CG scheme with an AMG preconditioner is used. Once the two iterative solvers converge, we update the approximated solution x for the next outer iterate in line 16.

4. NUMERICAL EXPERIMENTS

This section is devoted to assessing the numerical performance and parallel scalability of our implementation on different test cases. We use our own mesher to generate structured meshes, TetGen [20] for unstructured and locally refined meshes and METIS [19] to partition the elements into non-overlapping subdomains. We use PETSc [17] as the framework of our iterative solution code. For AMG preconditioning, we use BoomerAMG [26], which is part of the Hypre [18] package. In all experiments, the relative residual of the outer iteration is set to 1e–6 and the relative residual of the inner iteration is set to 1e–8, unless explicitly specified. The code is executed on a cluster with up to 12 nodes. Each node has eight 2.6 GHz Intel processors and 16 GB RAM.

The following notation is used to record our results: *np* denotes the number of processors of the run, its is the number of outer MINRES iterations, its_{i_1} is the number of inner CG iterations for solving $\mathscr{P}_M$, its_{i_2} is the number of CG iterations for solving L, while t_s and t_a denote the average times needed in seconds for the solve phase and the assemble phase, respectively. The parameter t_AMG is the time spent in seconds in one BoomerAMG V-cycle for solving L.

4.1. Example 1

The first example is a simple domain with a structured mesh. The domain is a cube, $\Omega=(-1,1)^3$. We test both homogeneous and inhomogeneous coefficient cases. In the homogeneous case, we set

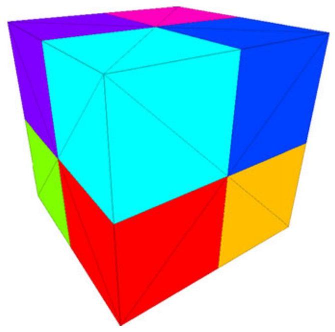
Figure 2. Example 1. Distribution of material coefficients.

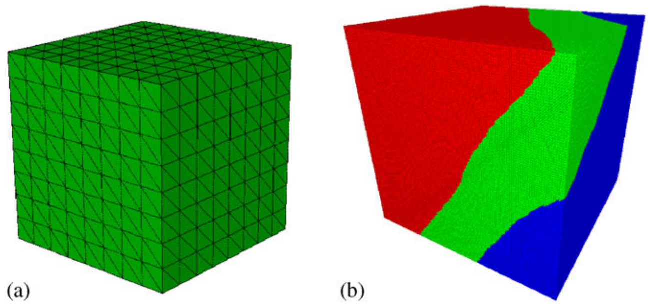

Figure 3. Example 1. (a) Structured mesh and (b) grid C1 partitioned on three processors.

$\mu_r = \varepsilon_r = 1$. In the variable coefficient case, we assume that there are eight subdomains in the cube as shown in Figure 2 and each subdomain has piecewise constant coefficients. The coefficients are

$$\mu_r = \varepsilon_r = \begin{cases} 1a & \text{if } x<0 \text{ and } y<0 \text{ and } z<0, \\ 2a & \text{if } x>0 \text{ and } y<0 \text{ and } z<0, \\ 3a & \text{if } x<0 \text{ and } y>0 \text{ and } z<0, \\ 4a & \text{if } x>0 \text{ and } y>0 \text{ and } z<0, \\ 5a & \text{if } x<0 \text{ and } y<0 \text{ and } z>0, \\ 6a & \text{if } x>0 \text{ and } y<0 \text{ and } z>0, \\ 7a & \text{if } x<0 \text{ and } y>0 \text{ and } z>0, \\ 8a & \text{otherwise,} \end{cases} \tag{17}$$

where a is a constant. We set the right-hand side so that the solution of (1) is given by

$$u(x,y,z) = \begin{pmatrix} u_1(x,y,z) \\ u_2(x,y,z) \\ u_3(x,y,z) \end{pmatrix} = \begin{pmatrix} (1-y^2)(1-z^2) \\ (1-x^2)(1-z^2) \\ (1-x^2)(1-y^2) \end{pmatrix} \tag{18}$$

and

$$p(x,y,z) = (1-x^2)(1-y^2)(1-z^2). \tag{19}$$

In this example, the homogeneous boundary conditions in (1) are satisfied.

Uniformly refined meshes are constructed shown in Figure 3(a). The number of elements and matrix sizes are given in Table I. Figure 3(b) shows how grid C1 is partitioned across three

Table I. Example 1. Number of elements (Nel) and the size of the linear systems ($n+m$) for grids C1–C3.

Grid	Nel	$n+m$
C1	7 146 096	9 393 931
C2	14 436 624	19 034 163
C3	29 478 000	38 958 219

Table II. Example 1. Partitioning of grid C1.

Processor	Local elements	Local DOFs
1	2 359 736	3 119 317
2	2 424 224	3 199 406
3	2 362 136	3 075 208

Table III. Example 1. Iteration counts and computation times for various grids, $k=0$.

np	Grid	its	its_{i_1}	its_{i_2}	t_s (s)	t_a (s)	t_{AMG} (s)
3	C1	5	34	7	1,473.58	44.83	16.03
6	C2	5	35	9	1,634.17	45.26	20.55
12	C3	5	34	9	1,879.06	48.93	25.39

Table IV. Example 1. Iteration counts for various values of k.

		$k=0$			$k=\frac{1}{8}$			$k=\frac{1}{4}$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
3	C1	5	34	7	5	34	7	5	34	7
6	C2	5	35	9	5	35	9	5	35	9
12	C3	5	34	9	5	34	9	5	34	9

processors. Elements with the same color are stored on the same processor. Elements with the same color are clustered together, which means that the communication cost is minimal. Table II shows the local numbers of elements and degrees of freedom on each processor for grid C1. The number of degrees of freedom on each processor is roughly the same, which indicates that the load is balanced.

In our first experiment, material coefficients are homogeneous. Scalability results are shown in Table III and the results with different values of k are shown in Table IV. To test scalability, we refine the mesh and increase the number of processors in a proportional manner, so that the problem size per processor remains constant. Full scalability would then imply that the computation time also remains constant. We observe in Table III that when the mesh is refined, the numbers of outer and inner iterations stay constant, which demonstrates the scalability of our method. The time spent in the assembly also scales very well. The time spent in the solve increases slightly. This is because each BoomerAMG V-cycle seems to take more time when the mesh is refined. For different values of k, we have observed very similar computation times as in Table III. Table IV shows that the iteration counts stay the same for the values of k that we have selected. This demonstrates the scalability of our solver.

Setting the outer tolerance as 1e–6, we test our solver with different inner tolerances. The results are given in Table V. We see that when the inner tolerance is looser, fewer inner iterations are required; however if the tolerance is too loose, more outer iterations are required. For this example, 1e–6 is the optimal inner tolerance.

Table V. Example 1. Iteration counts and computation times for grid C1 on 16 processors for inexact inner iterations, $k=0$.

Inner tol	its	its_{i_1}	its_{i_2}	t_s (s)
1e–10	5	43	9	557.08
1e–9	5	39	8	505.30
1e–8	5	35	8	440.92
1e–7	5	30	7	383.34
1e–6	6	21	7	375.91
1e–5	8	20	6	409.57
1e–4	21	16	5	794.15

Table VI. Example 1. Iteration counts for various values of a, $k=0$.

		$a=1$			$a=10$			$a=20$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
3	C1	21	36	8	128	33	8	238	31	8
6	C2	21	35	10	128	33	10	238	31	10
12	C3	21	33	11	130	33	11	236	31	11

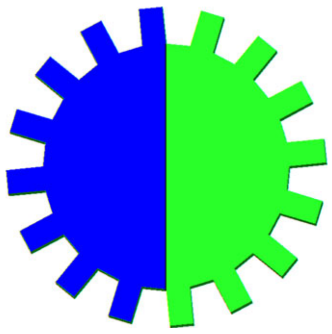

Figure 4. Example 2. Distribution of material coefficients.

In the remaining examples, we stick to 1e–6 and 1e–8 as outer and inner tolerances, respectively. We select a tight inner tolerance since one of our goals is to investigate the speed of convergence of outer iterations.

Next we test the variable coefficient case. Table VI shows the iteration counts for different variable coefficient cases. Note that the larger a is, the more variant the coefficients are in different regions. As expected, the eigenvalue bound depends on the coefficients. Table VI shows that as a increases, the eigenvalue bounds in Theorem 3.1 get looser and the iteration counts strongly increase. In the variable coefficient case, both the inner and outer iterations are not sensitive to changes in the mesh size.

4.2. Example 2

In this example, we test the problem in a complicated domain with a quasi-uniform mesh. The domain is a complicated 3D gear, which is bounded in $(0.025, 0.975)\times(0.025, 0.975)\times(0.025, 0.15292)$. We test two different cases: constant and variable coefficient cases. In the constant coefficient test, we set $\mu_r=\varepsilon_r=1$. In the variable coefficient case, there are two subdomains as shown in Figure 4. We have two experiments for this case: first, we assume $\mu_r=1$ in the domain, and for $x<0.5$, $\varepsilon_r=1$ and for $x\geqslant 0.5$, ε_r varies from 10 to 1000. In the next experiment, we assume $\varepsilon_r=1$ in the domain, and for $x<0.5$, $\mu_r=1$ and for $x\geqslant 0.5$, μ_r varies from 10 to 1000. In both

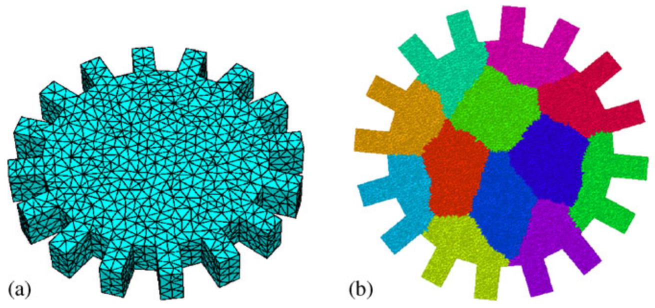

Figure 5. Example 2. (a) Quasi-uniform mesh on the gear and (b) grid G1 partitioned on 12 processors.

Table VII. Example 2. Number of elements (Nel) and the size of the linear systems ($n+m$) for grids G1–G4.

Grid	Nel	$n+m$
G1	723 594	894 615
G2	1 446 403	1 810 413
G3	2 889 085	3 650 047
G4	5 778 001	7 354 886

Table VIII. Example 2. Iteration counts and computation times for various grids, $k=0$.

np	Grid	its	its_{i_1}	its_{i_2}	t_s (s)	t_a (s)	t_{AMG} (s)
12	G1	4	58	8	72.11	2.50	0.42
24	G2	4	61	9	102.27	2.56	0.67
48	G3	4	64	9	138.87	2.55	1.05
96	G4	4	66	10	190.19	2.68	1.52

Table IX. Example 2. Iteration counts for various values of k.

		$k=0$			$k=\frac{1}{8}$			$k=\frac{1}{4}$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
12	G1	4	58	8	4	58	8	4	58	8
24	G2	4	61	9	4	61	9	4	61	9
48	G3	4	64	9	4	64	9	4	64	9
96	G4	4	66	10	4	66	10	4	66	10

constant and variable coefficient cases, we set the right-hand side function so that the exact solution is given by (18) and (19), and enforce the inhomogeneous boundary conditions in a standard way.

The domain is meshed with quasi-uniform tetrahedra as shown in Figure 5(a). The number of elements and matrix sizes is given in Table VII. Figure 5(b) shows how to partition G1 across 12 processors.

First, we test our approach with constant coefficients. The scalability results are shown in Table VIII. The results with different values of k are shown in Table IX. We observe that when the mesh is refined, both the inner and outer iteration counts stay constant. Again, the time spent in the assembly scales very well. The time spent in the solve is increasing slightly, which can be explained by the increase in t_{AMG}. We note that the computational times for different values of k are similar to those reported for $k=0$ in Table VIII. Table IX shows that the iteration counts do not change for different k values.

Table X. Example 2. Iteration counts for various values of ε_r, $\mu_r = 1$, $k = 0$.

		$\varepsilon_r = 10$			$\varepsilon_r = 100$			$\varepsilon_r = 1000$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
12	G1	5	58	8	9	47	8	18	50	8
24	G2	5	60	9	9	51	9	18	54	9
48	G3	5	63	9	9	54	10	17	54	10
96	G4	5	65	10	9	53	10	17	55	10

Table XI. Example 2. Iteration counts for various values of μ_r, $\varepsilon_r = 1$, $k = 0$.

		$\mu_r = 10$			$\mu_r = 100$			$\mu_r = 1000$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
12	G1	5	59	8	7	61	8	15	69	8
24	G2	5	64	9	7	63	9	15	64	9
48	G3	5	65	9	7	62	9	14	67	9
96	G4	5	65	10	7	64	10	14	65	10

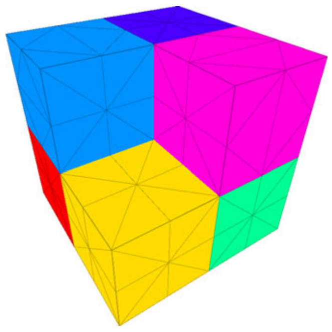
Figure 6. Example 3. Distribution of material coefficients.

Next, we test the variable coefficient example. Tables X and XI show the results for different variable coefficient cases. Again, we observe that when the variation of the material coefficients increases, the iteration counts increase. We also observe that both the inner and outer iterations are not sensitive to changes in the mesh size for the example with unstructured meshes.

4.3. Example 3

The last example is the Fichera corner problem. We are interested in testing our solver on a series of locally refined meshes. The domain $\Omega = (-1, 1)^3 \backslash [0, 1) \times [0, -1) \times [0, 1)$ is a cube with a missing corner. We also test both homogeneous and inhomogeneous coefficients. In the homogeneous coefficient case, we set $\mu_r = \varepsilon_r = 1$. In the inhomogeneous case, we assume that there are seven subdomains in the domain as shown in Figure 6 and each subdomain has piecewise constant coefficients. The coefficients are the same as (17). In both tests, we set the right-hand side function so that the exact solution is given by (18) and (19), and enforce the inhomogeneous boundary conditions in a standard way.

The domain is discretized with locally refined meshes toward the corner. Figure 7 shows an example of a sequence of locally refined meshes. The number of elements and matrix sizes is given in Table XII. Figure 8 shows the partitioning of F1 on four processors.

First, we assume that the material coefficients are homogeneous. The scalability results are shown in Table XIII. The results with different values of k are shown in Table XIV. On the locally

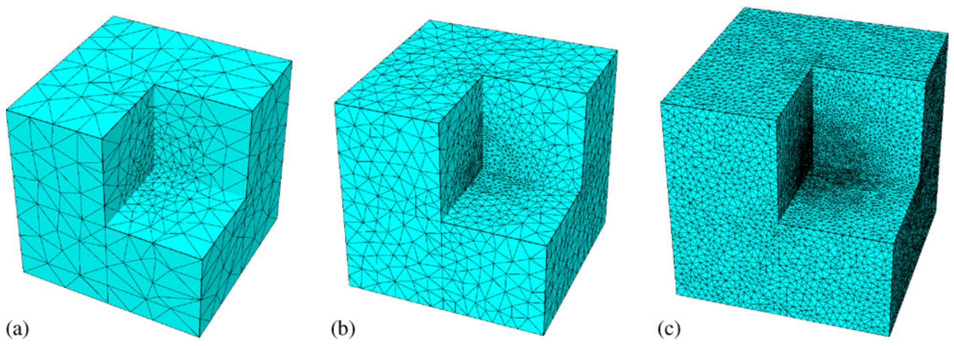

Figure 7. Example 3. A sequence of locally refined meshes.

Table XII. Example 3. Number of elements (Nel) and the size of the linear systems ($n+m$) for grids F1–F4.

Grid	Nel	$n+m$
F1	781 614	957 277
F2	1 543 937	1 917 649
F3	3 053 426	3 832 895
F4	6 072 325	7 689 953

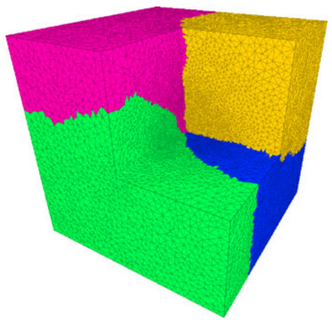

Figure 8. Example 3. Illustration of grid F1 partitioned on four processors.

Table XIII. Example 3. Iteration counts and computation times for various grids, $k=0$.

np	Grid	its	its_{i_1}	its_{i_2}	t_s (s)	t_a (s)	t_{AMG} (s)
4	F1	5	59	8	204.82	6.54	0.88
8	F2	5	56	9	213.16	6.39	1.17
16	F3	5	58	10	253.81	6.29	1.65
32	F4	5	62	10	314.88	6.38	1.99

refined meshes, we also observe that when the mesh is refined, both the inner and outer solvers are scalable. Again, the time spent in the assembly scales very well. The time spent in the solve is increasing, which is due to the increasing cost of BoomerAMG V-cycles. Table XIV shows that the iteration counts do not change for different wave numbers. As in the previous examples, in our experiments the computational times are also roughly the same as in Table XIII for different values of k.

Table XIV. Example 3. Iteration counts for various values of k.

		$k=0$			$k=\frac{1}{8}$			$k=\frac{1}{4}$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
4	F1	5	59	8	5	59	8	5	59	8
8	F2	5	56	9	5	56	9	5	56	9
16	F3	5	58	10	5	58	10	5	58	10
32	F4	5	62	10	5	62	10	4	63	10

Table XV. Example 3. Iteration counts for various values of a, $k=0$.

		$a=1$			$a=10$			$a=20$		
np	Grid	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}	its	its_{i_1}	its_{i_2}
4	F1	13	59	8	86	77	9	158	92	9
8	F2	13	55	9	81	69	9	153	82	9
16	F3	13	56	10	85	61	10	144	79	10
32	F4	13	60	10	84	60	11	139	82	10

Next, we test the variable coefficient case. Table XV shows the results for different variable coefficient cases. Again, we observe that when the coefficients vary more, the iteration counts increase, but the scalability with respect to the mesh size is very good.

5. CONCLUSIONS

We have developed and implemented a fully scalable parallel iterative solver for the time-harmonic Maxwell equations with heterogeneous coefficients, on unstructured meshes. For our parallel implementation we use our own code, combined with PETSc [17], Hypre [18], METIS [19]), and TetGen [20].

Our mixed formulation maintains a saddle-point structure, which allows for dealing with the discrete gradients without post-processing. From a computational point of view, the saddle-point form provides some extra flexibility, which we exploit by using an inner–outer iterative approach. For the outer iterations we adapt [8] to the variable coefficient case. The inner iterations are based on the auxiliary spaces technique developed in [15].

We have shown that for moderately varying coefficients, the preconditioned matrix is well conditioned and its eigenvalues are tightly clustered; this is key to the effectiveness of the proposed approach. As our extensive numerical experiments indicate, the inner and outer iterations are scalable in terms of iteration counts and computation times, and the solver is minimally sensitive to changes in the mesh size.

More work is required for making our solver robust for highly varying or discontinuous coefficients. Currently, the iteration counts are insensitive to the mesh size, but they increase when the coefficients vary strongly. Furthermore, our results apply to low wave numbers; dealing with high wave numbers is a primary challenge of much interest and remains an item for future work.

REFERENCES

1. Chen Z, Du Q, Zou J. Finite element methods with matching and nonmatching meshes for Maxwell equations with discontinuous coefficients. *SIAM Journal on Numerical Analysis* 2000; **37**(5):1542–1570.
2. Demkowicz L, Vardapetyan L. Modeling of electromagnetic absorption/scattering problems using *hp*-adaptive finite elements. *Computer Methods in Applied Mechanics and Engineering* 1998; **152**:103–124.
3. Hiptmair R. Finite elements in computational electromagnetism. *Acta Numerica* 2002; **11**:237–339.
4. Monk P. *Finite Element Methods for Maxwell's Equations*. Oxford Science Publications: Oxford, 2003.

5. Vardapetyan L, Demkowicz L. *hp*-adaptive finite elements in electromagnetics. *Computer Methods in Applied Mechanics and Engineering* 1999; **169**:331–344.
6. Jin J. *The Finite Element Method in Electromagnetics*. Wiley: New York, 1993.
7. Nédélec JC. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik* 1980; **35**:315–341.
8. Greif C, Schötzau D. Preconditioners for the discretized time-harmonic Maxwell equations in mixed form. *Numerical Linear Algebra with Applications* 2007; **14**(4):281–297.
9. Hiptmair R. Multigrid method for Maxwell's equations. *SIAM Journal on Numerical Analysis* 1998; **36**(1):204–225.
10. Bochev P, Garasi C, Hu J, Robinson A, Tuminaro R. An improved algebraic multigrid method for solving Maxwell's equations. *SIAM Journal on Scientific Computing* 2003; **25**:623–642.
11. Jones J, Lee B. A multigrid method for variable coefficient Maxwell's equations. *SIAM Journal on Scientific Computing* 2006; **27**(5):1689–1708.
12. Reitzinger S, Schöberl J. An algebraic multigrid method for finite element discretizations with edge elements. *Numerical Linear Algebra with Applications* 2002; **9**(3):223–238.
13. Gopalakrishnan J, Pasciak JE. Overlapping Schwarz preconditioners for indefinite time harmonic Maxwell equations. *Mathematics of Computation* 2003; **72**:1–15.
14. Gopalakrishnan J, Pasciak JE, Demkowicz LF. Analysis of a multigrid algorithm for time harmonic Maxwell equations. *SIAM Journal on Numerical Analysis* 2004; **42**:90–108.
15. Hiptmair R, Xu J. Nodal auxiliary space preconditioning in $H(\text{curl})$ and $H(\text{div})$ spaces. *SIAM Journal on Numerical Analysis* 2007; **45**(6):2483–2509.
16. Kolev TV, Vassilevski PS. Parallel auxiliary space AMG for $H(\text{curl})$ problems. *Journal of Computational Mathematics* 2009; **27**(5):604–623.
17. Balay S, Gropp WD, McInnes LC, Smith BF. PETSc users manual. *Technical Report ANL-95/11—Revision 2.3.2*, Argonne National Laboratory, 2006.
18. Falgout R, Cleary A, Jones J, Chow E, Henson V, Baldwin C, Brown P, Vassilevski P, Yang UM. Hypre: high performace preconditioners. Available from: http://acts.nersc.gov/hypre/, 2010.
19. Karypis G. METIS: Family of multilevel partitioning algorithms. Available from: http://glaros.dtc.umn.edu/gkhome/views/metis/, 2010.
20. Si H. TetGen: A quality tetrahedral mesh generator and a 3D Delaunay triangulator. Available from: http://tetgen.berlios.de/, 2010.
21. Brezzi F, Fortin M. *Mixed and Hybrid Finite Element Methods*. Springer: New York, NY, U.S.A., 1991.
22. Paige C, Saunders M. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* 1975; **12**:617–629.
23. Li D. Numerical solution of the time-harmonic Maxwell equations and incompressible magnetohydrodynamics problems. *Ph.D. Thesis*, The University of British Columbia, 2010.
24. Griebel M, Oswald P. On the abstract theory of additive and multiplicative Schwarz algorithms. *Numerische Mathematik* 1995; **70**(2):163–180.
25. Xu J. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing* 1996; **56**(3):215–235.
26. Henson VE, Yang UM. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* 2000; **41**:155–177.