UCRL-TR-226197

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Parallel eigensolver for H(curl) problems using H1-auxiliary space AMG preconditioning

Tz. V. Kolev, P. S. Vassilevski

November 16, 2006

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

PARALLEL EIGENSOLVER FOR $\boldsymbol{H}(\mathrm{curl})$ PROBLEMS USING $\boldsymbol{H}^1$-AUXILIARY SPACE AMG PRECONDITIONING

TZANIO V. KOLEV AND PANAYOT S. VASSILEVSKI

Abstract. This report describes an application of the recently developed $\boldsymbol{H}^1$-auxiliary space preconditioner for $\boldsymbol{H}(\mathrm{curl})$ problems to the Maxwell eigenvalue problem. The auxiliary space method based on the new (HX) finite element space decomposition introduced in [7], was implemented in the *hypre* library, [10, 11] under the name AMS. The eigensolver considered in the present paper, referred to as the AME, is an extension of the AMS. It is based on the locally optimal block eigensolver LOBPCG [9] and the parallel AMG (algebraic multigrid) solver BoomerAMG [2] from the *hypre* library. AME is designed to compute a block of few minimal nonzero eigenvalues and eigenvectors, for general unstructured finite element discretizations utilizing the lowest order Nédélec elements. The main goal of the current report is to document the usage of AME and to illustrate its parallel scalability.

1. Introduction

The computation of Maxwell eigenvalues is a challenging problem with important practical applications in the design of linear accelerators, optical waveguides and lasers. Together with the definite, semidefinite and indefinite Maxwell equations, it is one of the the major problems in computational electromagnetics. In this report we show how the previously developed preconditioner for (semi)definite Maxwell problems can be used to obtain a robust solver for the corresponding eigenvalue problem.

Let Ω be a three-dimensional, simply-connected domain, and $\mathcal{T}_h$ be a shape regular mesh on Ω. Associated with $\mathcal{T}_h$ are the spaces of continuous piecewise linear finite elements S_h, and the lowest order Nédélec finite element space $\boldsymbol{V}_h$. The Maxwell eigenvalue problem has the following variational form: find the nonzero eigenvectors $\mathbf{u} \in \boldsymbol{V}_h$ and eigenvalues λ, satisfying

$$(\alpha \,\mathrm{curl}\,\mathbf{u}, \mathrm{curl}\,\mathbf{v}) = \lambda\,(\beta\,\mathbf{u}, \mathbf{v})\,, \qquad \text{for all } \mathbf{v} \in \boldsymbol{V}_h \tag{1.1}$$

and

$$(\beta\,\mathbf{u}, \nabla\phi) = 0\,, \qquad \text{for all } \phi \in S_h\,. \tag{1.2}$$

Here $\alpha > 0$ and $\beta > 0$ are given scalar coefficients, and the second condition guarantees that the eigenvectors (scaled by β) are discretely divergence free.

Let $\boldsymbol{A}$ and $\boldsymbol{M}$ be the stiffness and (weighted) mass matrices corresponding to (1.1). Then, we are interested in finding few minimal nonzero eigenpairs $(\lambda, \mathbf{x})$ of the linear

Date: October 12, 2006–beginning; Today is November 15, 2006.

1991 *Mathematics Subject Classification.* 65F10, 65N20, 65N30.

This work was performed under the auspices of the U. S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

eigenproblem

$$\boldsymbol{A}\,\mathbf{x} = \lambda \boldsymbol{M}\mathbf{x} \tag{1.3}$$

subject to

$$G^t \boldsymbol{M}\mathbf{x} = 0\,. \tag{1.4}$$

Here G is the discrete gradient matrix representing the edges of the mesh in terms of its vertices, see [11] for more details. Note that the matrix $\boldsymbol{A}$ is singular, which motivates many eigensolvers to replace (1.3) with the equivalent problem

$$(\boldsymbol{A} + \boldsymbol{M})\,\mathbf{x} = (\lambda + 1)\boldsymbol{M}\mathbf{x}\,. \tag{1.5}$$

Our approach can use any of the above formulations, but we prefer (1.3), since it is somewhat cheaper, both in terms of memory usage and total running time.

Let $\boldsymbol{B}$ be the $\boldsymbol{H}^1$-auxiliary space preconditioner [7, 11] corresponding to $\boldsymbol{A}$, or $\boldsymbol{A}+\boldsymbol{M}$ if we have chosen the formulation (1.5). We use $\boldsymbol{B}$ in the iterative block eigensolver LOBPCG [9]. One advantage of this choice is that we don't need a guess for the minimal eigenvalue, as in most shift-and-invert algorithms. LOBPCG computes the matrix $\mathbf{X}_{k+1}$ representing the next eigenvectors approximation based on a three-term recurrence involving $\mathbf{X}_k$, $\mathbf{X}_{k-1}$ and $\boldsymbol{B}(\boldsymbol{A}\mathbf{X}_k - \boldsymbol{M}\mathbf{X}_k\boldsymbol{\Lambda}_k)$. More specifically, each $\mathbf{X}_k$ has number of columns equal to the number of eigenvectors we want to compute. Here $\boldsymbol{\Lambda}_k$ is a diagonal matrix representing the current approximations of the eigenvalues. Since we want to perform this iteration in the subspace satisfying (1.4), we have to guarantee that the result of $\boldsymbol{B}$ is discretely divergence free. This can be achieved by filtering the gradient component of the vectors using the projection

$$P = I - G\,(G^t\boldsymbol{M}G)^{-1}\,G^t\boldsymbol{M}.$$

In practice, P is only approximated and there are a couple of different ways to do that, see [6, 12]. In the current implementation of AME, we follow each application of the preconditioner by an approximation of P where the inverse is replaced by a PCG-AMG solver for the Laplacian–like matrix $G^t\boldsymbol{M}G$.

The remainder of the report is organized as follows: the parallel version of the eigensolver, as implemented in the *hypre* library, is described in Section 2. Section 3 contains a set of numerical experiments demonstrating the parallel scalability and overall performance of the method.

2. The AME implementation in *hypre*

In this section we discuss the parallel implementation of the $\boldsymbol{H}^1$-based auxiliary space eigensolver in the *hypre* library under the name AME (Auxiliary Maxwell Eigensolver). Below we list the sequence of *hypre* calls needed to create and use it. First, we allocate the `HYPRE_Solver` object:

```
HYPRE_Solver solver;
HYPRE_AMECreate(&solver);
```

We assume that the user has built an AMS preconditioner for the stiffness matrix $\boldsymbol{A}$. See the report [11] for a detailed description of this process. The preconditioner should be passed to AME before the setup phase using the following command:

```
HYPRE_Solver ams_precond;
HYPRE_AMESetAMSSolver(solver, ams_precond);
```

The user is also required to provide the mass matrix $\boldsymbol{M}$.

```
HYPRE_AMESetMassMatrix(solver, M);
```

The number of desired eigenvectors, referred to as `block_size`, is set with

```
HYPRE_AMESetBlockSize(solver, block_size);
```

The remaining solver parameters are optional. For example, the user can set the maximum number of iterations, the convergence tolerance and the output level with

```
HYPRE_AMESetMaxIter(solver, maxit);     /* default value: 100 */
HYPRE_AMESetTol(solver, tol);           /* default value: 1e-6 */
HYPRE_AMESetPrintLevel(solver, print);  /* default value: 1 */
```

After the above calls, the eigensolver is ready to be constructed. The setup call reads,

```
HYPRE_AMESetup(solver);
```

Once the setup has been completed, the eigenproblem can be solved by calling

```
HYPRE_AMESolve(solver);
```

The computed eigenvalues and eigenvectors are obtained as follows:

```
double *eigenvalues;
HYPRE_ParVector *eigenvectors;
HYPRE_AMEGetEigenvalues(solver, &eigenvectors)
HYPRE_AMEGetEigenvectors(solver, &eigenvectors)
```

Finally, the eigensolver can be destroyed with

```
HYPRE_AMEDestroy(&solver);
```

More details on the implementation can be found in the files `ampes.h` and `ampes.c` located in the `parcsr_ls` directory of *hypre* .

3. Numerical experiments

In this section we present results from numerical experiments with AME based on LOBPCG and using AMS with its default parameters, see [11]. The convergence tolerance in the LOBPCG (the norm of the maximal residual) was set to 10^{-6}. The input matrices and vectors were constructed in parallel using the unstructured finite element package aFEM. In our experiments, we tried to keep the problem size per processor approximately the same (while increasing the number of processors), although the resulting load balance varied somewhat with the number of processors. The following notation is used in the tables of this section:

- np denotes the number of processors used,
- N is the total size of the problem,
- n_{it} is the number of LOBPCG iterations,
- t_{setup}, t_{solve} and t denote the average times (in seconds) needed for setup, solve and time to solution (setup and solve), respectively, on a machine with 2.4GHz Xeon processors.

3.1. **Constant coefficients.** In this subsection we consider constant coefficient eigenproblems with $\alpha = \beta = 1$. In the first example, the domain is the unit cube meshed with an unstructured tetrahedral mesh, and we compute the first five eigenpairs. The initial coarse mesh, before serial or parallel refinement, is shown in Figure 1. The results listed in Table 1 show that the formulation (1.3) is preferable to the non-singular variant,

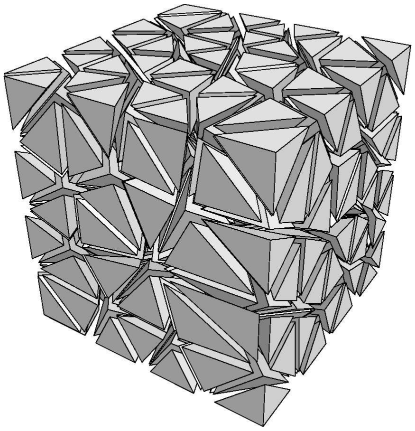


FIGURE 1. Initial unstructured tetrahedral mesh on the unit cube.

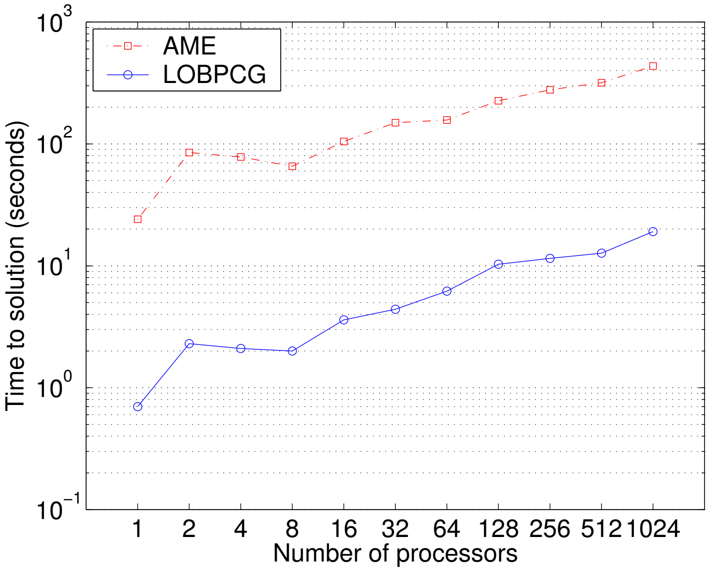


FIGURE 2. Time to solution: AME for the first five Maxwell eigenvalues of the unit cube versus LOBPCG-AMG for the first four eigenvalues of the Laplacian.

(1.5) not only for its reduced memory requirements, but also for its slightly improved convergence.

To better assess the quality of the method, we compared AME with LOBPCG based on the BoomerAMG preconditioner [2] applied to the Laplace eigenproblem discretized with linear finite elements on the same mesh. Note that the two problem sizes are quite different. Note also that while we compute five eigenvalues in the left columns, we only compute four on the right. Those numbers were chosen to align with the eigenvalue multiplicities, since otherwise the eigenproblem is not well-posed and the convergence of LOBPCG suffers (which is typical for any iterative block eigensolver). The results listed in Table 1 and plotted on Figure 2 show that, though not perfectly scalable, the behavior of AME is qualitatively similar to that of LOBPCG-AMG. The trend observed in our previous numerical experiments with AMS that the auxiliary space methods for

	AME				LOBPCG			
np	N	n_{it}	t_{setup}	t_{solve}	N	n_{it}	t_{setup}	t_{solve}
`using singular matrix` (1.3)								
1	37,940	18	1.3s	22.8s	5,941	16	0.1s	0.6s
2	105,877	21	3.9s	81.1s	17,478	19	0.2s	2.1s
4	184,820	22	3.7s	74.5s	29,059	19	0.2s	1.9s
8	293,224	23	3.2s	62.3s	43,881	21	0.2s	1.8s
16	697,618	26	4.6s	100s	110,745	25	0.4s	3.2s
32	1,414,371	27	6.4s	143s	225,102	22	0.7s	3.7s
64	2,305,232	29	6.9s	150s	337,105	28	1.0s	5.2s
128	5,040,829	32	9.1s	217s	779,539	37	1.3s	9.0s
256	10,383,148	35	11.0s	267s	1,682,661	36	1.5s	10.0s
512	18,280,864	33	13.7s	304s	2,642,337	33	2.4s	10.3s
1024	38,367,625	41	17.8s	418s	5,845,443	37	4.7s	14.4s
`using non-singular matrix` (1.5)								
1	37,940	18	1.5s	24.8s	5,941	16	0.1s	0.6s
2	105,877	21	4.2s	88.8s	17,478	19	0.2s	2.1s
4	184,820	22	4.1s	82.4s	29,059	19	0.2s	1.9s
8	293,224	23	3.4s	66.7s	43,881	21	0.2s	1.8s
16	697,618	26	5.0s	109s	110,745	25	0.4s	3.2s
32	1,414,371	27	6.8s	157s	225,102	22	0.7s	3.7s
64	2,305,232	29	7.5s	164s	337,105	28	1.0s	5.2s
128	5,040,829	32	9.8s	234s	779,539	37	1.3s	9.0s
256	10,383,148	37	11.9s	300s	1,682,661	36	1.5s	10.0s
512	18,280,864	34	14.9s	334s	2,642,337	33	2.4s	10.3s
1024	38,367,625	42	20.7s	458s	5,845,443	37	4.7s	14.4s

Table 1. Numerical results for the first five eigenvalues of the unit cube with constant coefficients ($\alpha = \beta = 1$). Comparison with LOBPCG for the first four eigenvalues of the Laplacian.

Maxwell problems inherit the properties of their internal Poisson AMG preconditioner is confirmed for the AME as well.

We also show in Table 2 results from a similar experiment with the same setup, but we now compute the first eleven eigenpairs (and the first ten for the Laplacian). This table contains the largest eigenproblem we have solved, which has a total size of close to half a billion (eleven times the size N of one vector). To estimate the scalability with respect to the number of computed eigenpairs, we plot in Figure 3 the ratio of the times from Tables 1 and 2. For this particular problem, it appears that this ratio stabilizes around three.

Next, we illustrate the handling of more complicated geometry by computing the first eight eigenvalues on an unstructured approximation of the unit ball. The results in Table 3 are consistent with those for the unit cube.

The next eigenproblem is posed on the Fichera corner $[-1, 1]^3 \setminus [-1, 0]^3$. This is a difficult problem, since some of the eigenvectors have singularities at the origin. The

	AME				LOBPCG			
np	N	n_{it}	t_{setup}	t_{solve}	N	n_{it}	t_{setup}	t_{solve}
1	37,940	24	2.2s	69.3s	5,941	25	0.1s	2.5s
2	105,877	27	6.4s	243s	17,478	26	0.2s	7.8s
4	184,820	33	6.0s	245s	29,059	31	0.2s	7.8s
8	293,224	29	5.2s	190s	43,881	27	0.3s	6.2s
16	697,618	40	7.4s	325s	110,745	32	0.4s	10.7s
32	1,414,371	49	10.9s	494s	225,102	36	0.7s	14.0s
64	2,305,232	40	12.0s	450s	337,105	39	1.0s	16.0s
128	5,040,829	50	15.7s	694s	779,539	42	1.3s	22.8s
256	10,383,148	58	18.7s	886s	1,682,661	51	1.6s	28.6s
512	18,280,864	57	23.0s	991s	2,642,337	63	2.3s	38.9s
1024	38,367,625	65	28.5s	1307s	5,845,443	53	4.6s	41.7s

TABLE 2. Numerical results for the first eleven eigenvalues of the unit cube with constant coefficients. Comparison with LOBPCG for the first ten eigenvalues of the Laplacian.

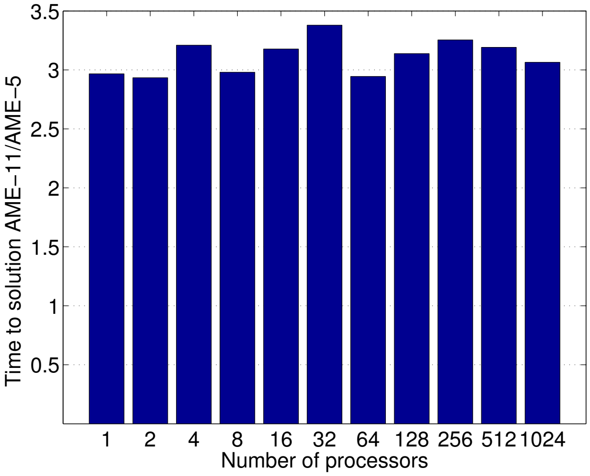
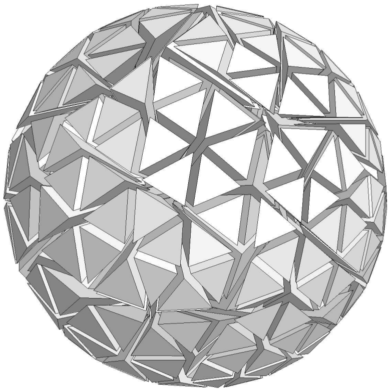


FIGURE 3. Ratio of the time to solution for the first eleven and five Maxwell eigenvalues on the unit cube.

results of the computation of the first eight eigenvalues in Table 4 show somewhat larger number of iterations compared to the previous two problems, but this is to be expected, since the eigenvectors in the previous domain are smooth.

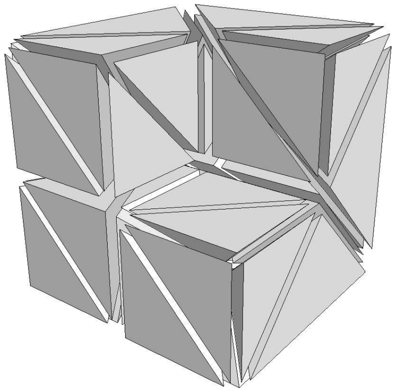
The exact eigenvalues of the Fichera corner are not known, but there are several available benchmarks. In Table 5 we compare our results (on the finest level) with three previously published such computations.

We conclude the set of constant coefficient experiments by considering two problems that involve fixed but complicated geometry and come from real-life applications. The meshes were kindly provided to us by the EMSolve project in Lawrence Livermore National Laboratory [4]. The first problem is a simplified version of the Titan target chamber shown in Figure 4. It has $123,681$ unknowns, and we compute the first eight eigenvalues.



np	N	n_{it}	t_{setup}	t_{solve}
1	33,722	21	1.4s	36s
2	94,759	24	4.3s	131s
4	164,735	24	4.2s	122s
8	255,700	28	3.6s	107s
16	610,773	32	5.1s	168s
32	1,226,798	35	8.9s	287s
64	1,990,184	35	15.2s	434s
128	4,351,501	42	15.3s	520s
256	8,948,940	46	19.5s	722s
512	15,701,584	49	21.7s	768s
1024	32,928,989	60	25.5s	990s

Table 3. Initial mesh and numerical results for the first eight eigenvalues of an approximation of the unit ball.



np	N	n_{it}	t_{setup}	t_{solve}
1	24,676	34	1.1s	30.0s
2	64,741	40	2.8s	93.8s
4	117,607	38	3.1s	93.6s
8	119,816	44	3.0s	93.5s
16	440,411	54	4.3s	150s
32	886,282	49	6.5s	206s
64	1,512,592	55	9.8s	312s
128	3,245,243	69	10.8s	406s
256	6,635,848	68	14.7s	533s
512	12,013,856	83	18.2s	725s
1024	24,972,012	100	22.2s	964s

Table 4. Initial mesh and numerical results for the first eight eigenvalues of the Fichera corner.

The second problem, shown in Figure 6 is a prototype of a linear accelerator induction cell with $593,773$ unknowns, and we computed the first ten eigenvalues. Plots of the magnitudes of some computed eigenvectors are shown in Figures 5 and 7.

For problems where the problem size is fixed, one is interested in the strong scalability of the eigensolver, i.e. how much faster the problem can be solved using more processors. We present such results in Figure 8. The conclusion is that, after the initial parallelization step, the scalability is quite reasonable. For example, the second problem took more than half an hour on one processor, while we were able to solve it in less than 3 minutes using 32 processors.

3.2. **Variable coefficients.** We previously demonstrated that the AMS preconditioner is capable of handling coefficients having different values in different regions of the domain (see [11]). Therefore we expect that the new eigensolver will also be robust with respect

eigenvalue	Dauge [3]	Bramble et al. [1]	Zaglmayr [12]	AME
1	3.31381	3.23432	3.21999	3.21599
2	5.88635	5.88267	5.88044	5.88022
3	5.88635	5.88371	5.88046	5.88047
4	10.6945	10.6789	10.6857	10.6855
5	10.6945	10.6832	10.6937	10.6956
6	10.7006	10.6945	10.6937	10.6964
7	12.3345	12.3653	12.3169	12.3128
8	12.3345	12.3723	12.3177	12.3146

Table 5. Fichera corner, eigenvalue benchmark results.

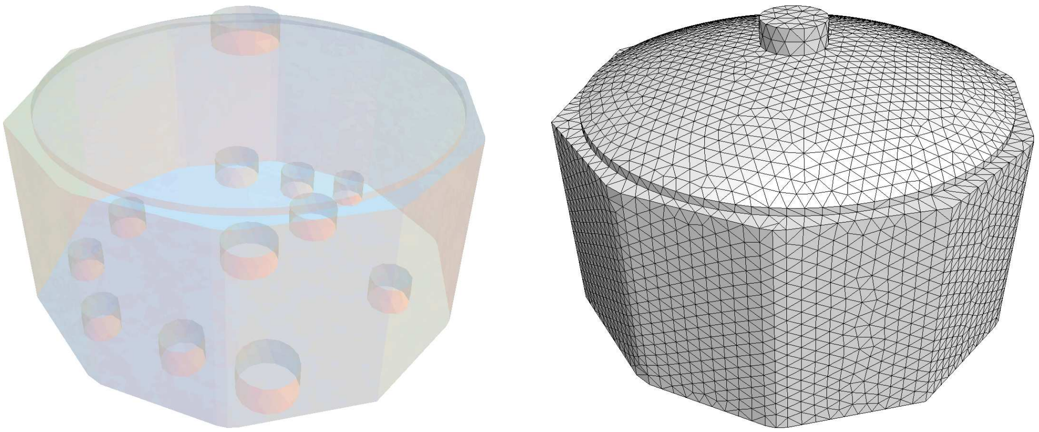


Figure 4. Domain and initial mesh for the Titan target chamber problem.

to jumps in the coefficients. As an illustration, we consider a problem where $\alpha = 1$, and β has different values in two regions of the domain. The geometry and the results are presented in Table 6, cf. [5].

We observe that the for jumps of moderate size, the number of iterations does not change much. This allowed us to solve an eigenproblem of total size around 190 million and 4 orders of magnitude jumps in the coefficients in less than 8 minutes.

References

[1] J. Bramble, Tz. Kolev and J. Pasciak, The approximation of the Maxwell eigenvalue problem using a least-squares method. *Math. Comp.*, 252(74):1575–1598, 2005.

[2] V. E. Henson and U. M. Yang, BoomerAMG: A Parallel Algebraic Multigrid Solver and Preconditioner. *Applied Numerical Mathematics*, 41:155–177, 2002.

[3] M. Dauge, Benchmark Computations for Maxwell Equations for the Approximation of Highly Singular Solutions. `http://perso.univ-rennes1.fr/monique.dauge/benchmax.html`.

[4] EMSolve: Unstructured Grid Computational Electromagnetics Using Mixed Finite Element Methods. `http://www-eng.llnl.gov/emsolve/emsolve_home.html`.

[5] R. Hiptmair, Multigrid method for Maxwell's equations. *SIAM J. Numer. Anal.*, 36(1):204–225, 1999.

[6] R. Hiptmair and K. Neymeyr, Multilevel Method for Mixed Eigenproblems. *SIAM J. Sci. Comp.*, 23(6):2141–2164, 2001.

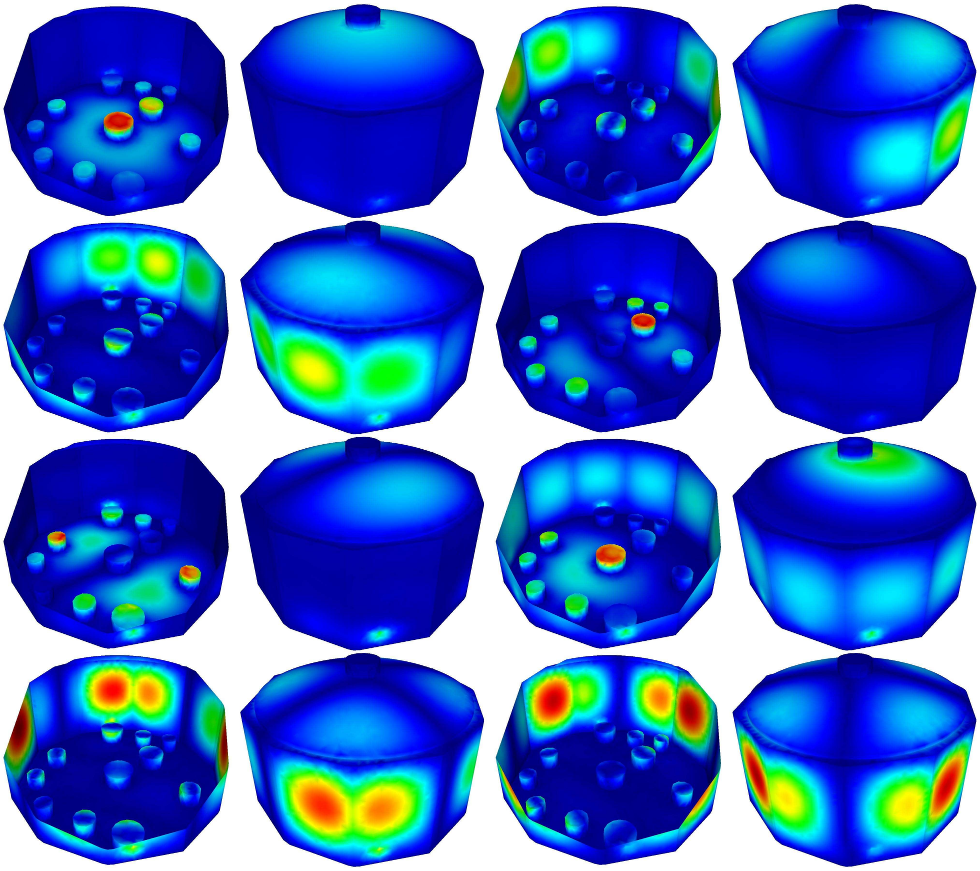


Figure 5. Magnitude of the computed electric field for the first eight eigenvalues of the the Titan target chamber.

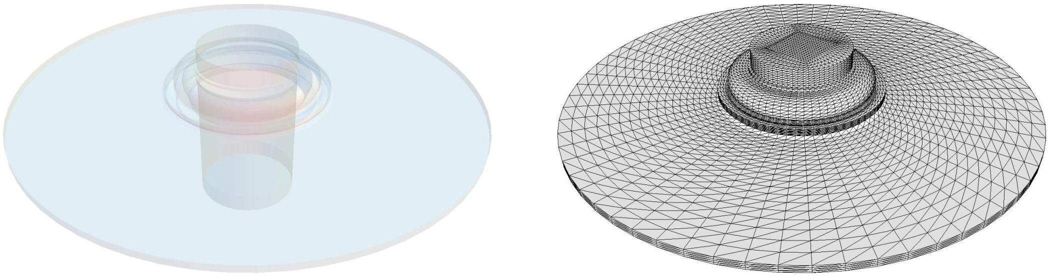


Figure 6. Domain and initial mesh for the linear accelerator cell problem.

[7] R. Hiptmair and J. Xu, Nodal auxiliary space preconditioning in $\boldsymbol{H}$(curl) and $\boldsymbol{H}$(div) spaces, *Research Report No. 2006-09*, Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule, CH-8092 Zürich, Switzerland, May 2006.

[8] *hypre*: High performance preconditioners. `http://www.llnl.gov/CASC/hypre/`.

[9] I. Lashuk, M. E. Argentati, E. Ovchinnikov and A. Knyazev, Preconditioned Eigensolver LOBPCG in hypre and PETSc, *Proceedings of the 16th International Conference on Domain Decomposition Methods.* Lecture Notes in Computational Science and Engineering, Springer Verlag, 55(2006).

[10] Tz. V. Kolev and P. S. Vassilevski, Some experience with a $\boldsymbol{H}^1$–based auxiliary space AMG for $\boldsymbol{H}$(curl)–problems, LLNL Technical Report UCRL-TR-221841.

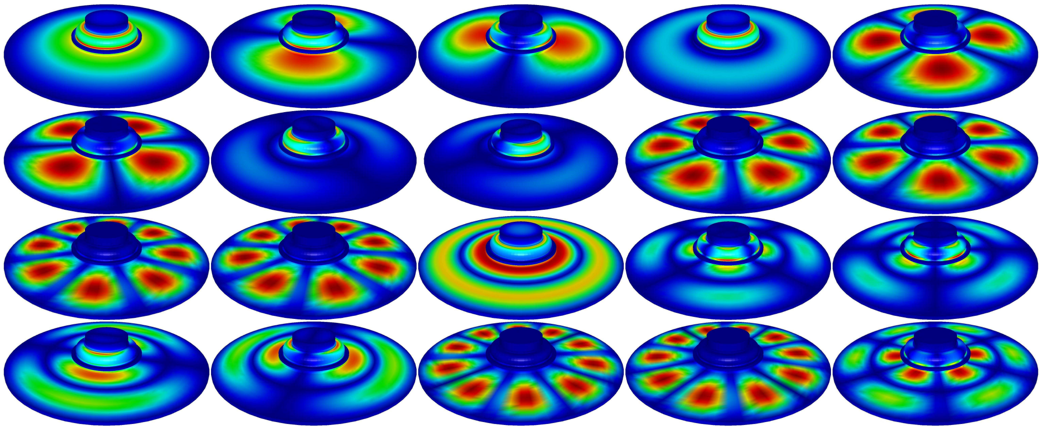


Figure 7. Magnitude of the computed electric field for the first twenty eigenvalues of the linear accelerator cell.

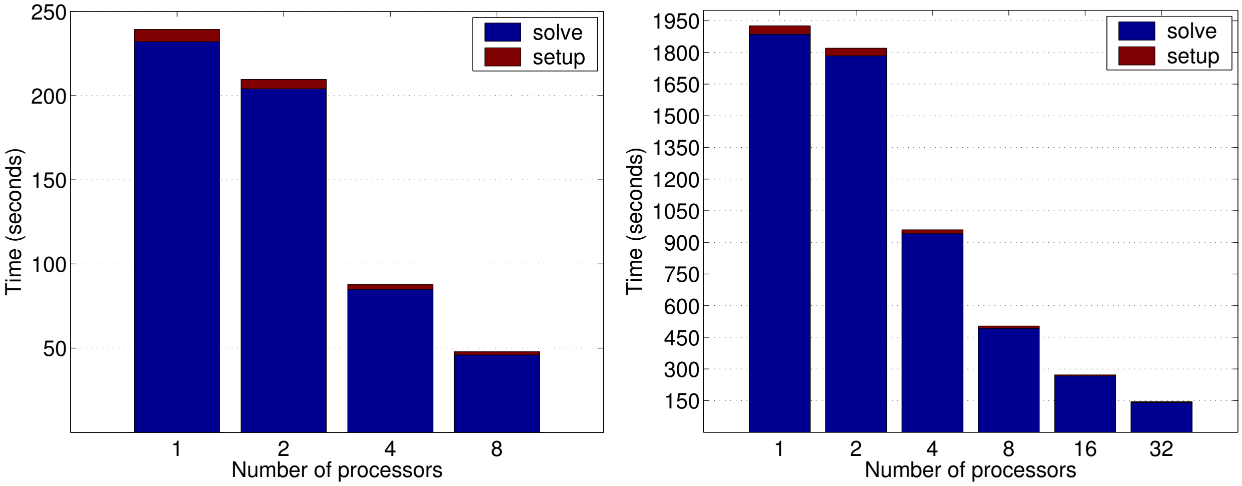


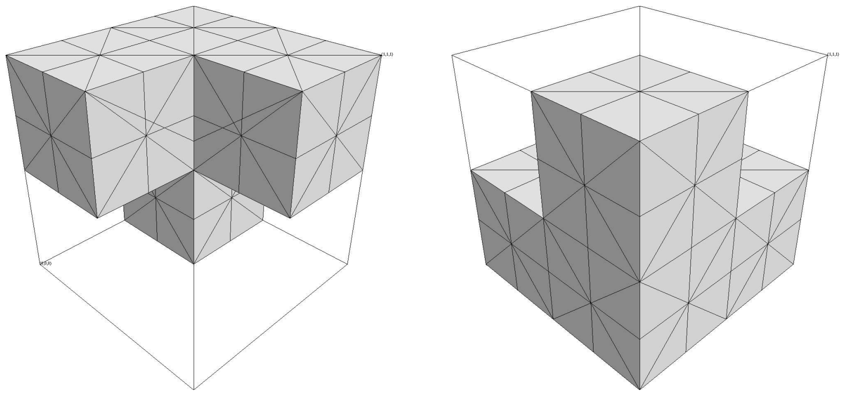
Figure 8. Setup and solution time for the Titan target chamber (left) and the linear accelerator cell (right) problems. Strong parallel scalability.

[11] Tz. V. Kolev and P. S. Vassilevski, Parallel $\boldsymbol{H}^1$–based auxiliary space AMG solver for $\boldsymbol{H}$(curl) problems, LLNL Technical Report UCRL-TR-222763.

[12] S. Zaglmayr, High Order Finite Element Methods for Electromagnetic Field Computation. *Dissertation*, Johannes Kepler University, Linz, Austria, 2006.

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, U.S.A.

E-mail address: `tzanio@llnl.gov, panayot@llnl.gov`



np	N	p							t
		-4	-2	-1	0	1	2	4	
$\alpha = 1, \beta \in \{1, 10^p\}$									
1	38,192	21	21	21	15	22	22	20	25s
2	83,278	22	22	22	18	25	26	22	64s
4	161,056	24	24	24	17	25	27	24	73s
8	296,032	25	25	24	21	27	28	25	78s
16	622,030	28	28	26	24	30	33	28	100s
32	1,249,272	32	31	31	25	29	30	27	138s
64	2,330,816	37	36	36	27	43	44	38	198s
128	4,810,140	34	34	33	29	40	41	37	256s
256	9,710,856	34	34	32	26	40	42	37	316s
512	18,497,920	48	47	45	40	53	60	52	475s
1024	37,864,880	48	48	48	34	45	48	42	478s

Table 6. Number of iterations for the computation of the first five eigenfunctions on a cube with $\alpha = 1$ and β having different values in the shown regions.