

Improved Efficiency and Accuracy of the Magnetic Polarizability Tensor Spectral Signature Object Characterisation for Metal Detection

J. Elgy^{a,*}, P. D. Ledger^a

^a*School of Computer Science and Mathematics, Keele University, Keele, Staffordshire, UK*

Abstract

Keywords:

1. Introduction

1.1. Model Problem

$$\begin{aligned} \nabla \times \nabla \times \mathbf{E} - k^2 \mathbf{E} &= \mathbf{0} & \text{in } \Omega \in \mathbb{R}^3 \\ \mathbf{n} \times \mathbf{E} &= \mathbf{n} \times \mathbf{E}^{(exact)} & \text{on } \Gamma = \partial\Omega \end{aligned} \quad (1)$$

with $\mathbf{E}^{(exact)} = \mathbf{p}e^{i\mathbf{k} \cdot \mathbf{x}}$ being the known solution for the electric field at position \mathbf{x} for wavevector $\mathbf{k} = [k_x, k_y, k_z]$ and perpendicular amplitude $\mathbf{p} = [p_x, p_y, p_z]$. For the standard $\mathbf{H}(\text{curl})$ conforming finite element space [4] the weak form of this problem is: Find $\mathbf{u} \in \mathbf{H}_D(\text{curl})$ such that

$$\int_{\Omega} \nabla \times \mathbf{u} \nabla \times \mathbf{v} \, d\Omega - \int_{\Omega} k^2 \mathbf{u} \mathbf{v} \, d\Omega = 0, \quad (2)$$

for all $\mathbf{v} \in \mathbf{H}_0(\text{curl})$ and $k^2 = |\mathbf{k}|^2 > 0$ being homogeneous and isotropic in Ω . Here, $\mathbf{H}(\text{curl})$ is the high order space defined by

$$\mathbf{H}(\text{curl}) := \{\mathbf{a} \in L^2(\Omega)^3 \mid \nabla \times \mathbf{a} \in L^2(\Omega)^3\},$$

where $L^2(\Omega)$ denotes the space of square integrable functions. Considering Dirichlet boundary conditions and setting \mathbf{v} to vanish on the boundary, the appropriate subspaces for this problem are

$$\begin{aligned} \mathbf{H}_D(\text{curl}) &:= \left\{ \mathbf{a} \in \mathbf{H}(\text{curl}) \mid \mathbf{n} \times \mathbf{a} = \mathbf{n} \times \mathbf{E}^{(exact)} \text{ on } \Gamma \right\} \\ \mathbf{H}_0(\text{curl}) &:= \left\{ \mathbf{a} \in \mathbf{H}(\text{curl}) \mid \mathbf{n} \times \mathbf{a} = \mathbf{0} \text{ on } \Gamma \right\}. \end{aligned}$$

The Galerkin finite element discretisation of the variational statement (2) is the large linear system

$$\mathbf{A}\mathbf{q} = \mathbf{r} \quad (3)$$

where \mathbf{A} is a large sparse matrix of size N_d . The symmetric sparse matrix \mathbf{A} is indefinite, and consequently may be difficult to solve.

[Should expand, mostly here to have something to cross reference.](#)

1.2. Perfectly Matched Layers

Perfectly Matched Layers (PMLs) are artificial absorbing boundary layers typically used in association with wave equation problems (such as (1), where boundary reflections are eliminated via the matching condition. The PML is constructed such that the EM field decays rapidly to 0 at the outer boundary by forcing a suitably rapid decay in the wave.

The implementation of the PML is handled via the `NGSolve` interface where the assembly of the linear and bilinear forms accounts for it.

**

Email addresses: j.elgy@keele.ac.uk (J. Elgy), p.d.ledger@keele.ac.uk (P. D. Ledger)

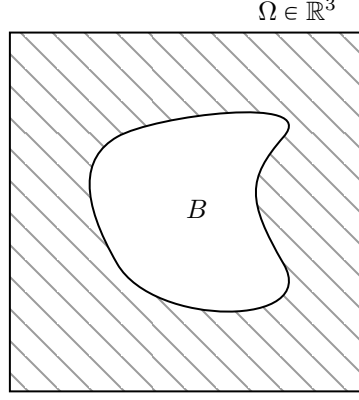


Figure 1: Illustration of the electromagnetic problem with PMLs outside the object B .

2. Preconditioners

Three different preconditioners are considered, The Local (Jacobi) preconditioner, the Balancing Domain Decomposition with Constraints (BDDC) preconditioner, and a geometric Multigrid preconditioner.

2.1. Local Preconditioner

The Local preconditioner, as implemented in `NGSolve`, is a simple Jacobi preconditioner. For this preconditioner, the preconditioner matrix \mathbf{P} is chosen as $\mathbf{P} := \text{diag}(\mathbf{A})$. Using Static condensation, the system matrix \mathbf{A} can be partitioned into local L and interface E degrees of freedom as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{LL} & \mathbf{A}_{LE} \\ \mathbf{A}_{EL} & \mathbf{A}_{EE} \end{bmatrix},$$

where internal degrees of freedom are eliminated via the Schur Complement [1]

$$\mathbf{S} = \mathbf{A}_{EE} - \mathbf{A}_{EL}\mathbf{A}_{LL}^{-1}\mathbf{A}_{LE}.$$

When assembling the bilinear form for use with static condensation, `NGSolve` computes \mathbf{S} , thus

$$\mathbf{A} = \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{S} \end{bmatrix},$$

and the preconditioner corresponds to

$$\mathbf{P} = \begin{bmatrix} 0 & 0 \\ 0 & \text{diag}(\mathbf{S}) \end{bmatrix}.$$

2.2. Multigrid Preconditioner

The multigrid preconditioner implemented by `NGSolve` uses a sequence of successively refined meshes such that a direct solve for the coarsest mesh is possible. Block Gauss-Seidel smoothers are then employed for finer meshes.

1 V cycle Uses vertex patch for complex FES smoothing

For the linear system (3), the multigrid preconditioner

Perform 1 iteration of Gauss-Sidel smoothing.

$$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{P}_{loc}^{-1}(\mathbf{r} - \mathbf{A}\mathbf{x})$$

3. Software

The computational resources were used to perform the simulations in this paper correspond to workstations with the following specifications: Intel i7-9700K CPU with a clock speed of 3.60GHz with 64GB DDR4 RAM.

The software used for this work `NGSolve` (version 6.2.2302) and `Netgen` (version 6.2.2302) [7, 9, 6], `SciPy` [8] (version 1.10.1), `NumPy` (version 1.24.2) [2].

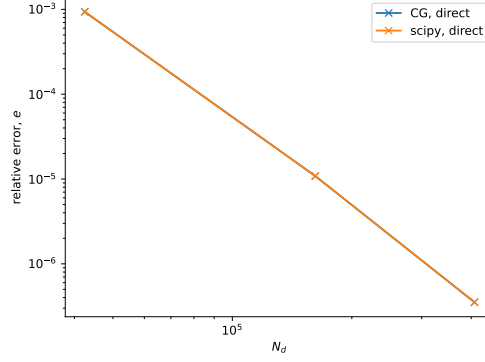


Figure 2: Non-magnetic non-conducting sphere of unit radius discretised using 14989 unstructured tetrahedral elements using $p = 0, 1, 2, 3$. A comparison between GMRES and conjugate gradient solvers, where differences are indistinguishable on this scale.

4. Conversion to Scipy Iterative Solvers

The iterative solvers available in `NGSolve`, notably `CGSolver` and `GMRESSolver` do not allow for the execution of arbitrary callback functions in the same way as many other solvers from other established python libraries. For this reason, it is difficult to retain the residual at each iteration of the `NGSolve` solvers. Furthermore, the `GMRESSolver` does not appear to have a restart option available to the API and is in general poorly documented. For this reason, the solving of the linear system used in static condensation is instead implemented using and

The `NGSolve` implementation of GMRES also gives noticeably different results to the `CGSolver`, even when using the same direct preconditioner, so I don't really trust it.

5. Numerical Results

In this section, we consider a range of numerical examples to illustrate the performance of the different preconditioners.

5.1. Non-Magnetic Sphere

We first consider the case of a sphere of radius 1 centred at $\mathbf{x} = \mathbf{0}$, discretised using $h = 0.04$, resulting in Ω being discretised by 14989 unstructured tetrahedral elements. The amplitude and wavevector are chosen as $\mathbf{p} = [0, 1, 0]$ and $\mathbf{k} = [1, 0, 0]$, respectively. We therefore expect the exact solution to have unit magnitude and a wavelength of 2π . Using $TOL = 1 \times 10^{-12}$ and a direct inverse preconditioner we compare the performance of the `NGSolve` conjugate gradient solver (which is valid given the applied preconditioner) with the `SciPy` GMRES solver. Figure 2 shows the error between the approximate and exact solutions in the L_2 norm,

$$e = \frac{\left(\int_{\Omega} (\mathbf{E}^{(hp)} - \mathbf{E}^{(exact)}) \cdot \overline{(\mathbf{E}^{(hp)} - \mathbf{E}^{(exact)})} d\Omega \right)^{1/2}}{\left(\int_{\Omega} (\mathbf{E}^{(exact)}) \cdot \overline{(\mathbf{E}^{(exact)})} d\Omega \right)^{1/2}}, \quad (4)$$

for the aforementioned discretisation and $p = 0, 1, 2, 3$ for both the GMRES and conjugate gradient solvers. The figure shows rapid decay of the error as p is increased. Similar agreement between the two solvers is also observed for other configurations. Due to the application of the direct inverse preconditioner, the solvers converge in 1 or 2 iterations.

Having established that the `SciPy` GMRES solver and the `NGSolve` CG Solver are in agreement, we now consider the case of the BDDC preconditioner, when used in conjunction with the `SciPy` GMRES solver. First, in Figure 3, we establish that the error, e , is invariant to the choice of preconditioner, for a sufficiently accurate solution to (3). The figure shows that applying both the BDDC and Multigrid preconditioners results in an accurate approximation to (2), whereas the Jacobi (local) preconditioner does not. This can be explained by the solution to (3) not being sufficiently accurate. A demonstration of the convergence behaviour for this problem can be seen in Figure 4, where the relative L_2 residual is plotted for each iteration for each preconditioner for $p = 0, 1, 2, 3$. The figure shows that while the BDDC and multigrid

preconditioners converge quickly, the Jacobi preconditioner does not, and is unsuitable for this problem. In the case of the Multigrid and BDDC preconditioners, the lowest order terms are solved directly, as a result the $p = 0$ case is solved in 1 iteration.

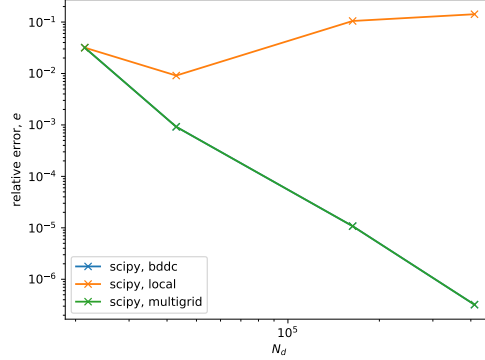


Figure 3: Non-magnetic non-conducting sphere of unit radius discretised using 14989 unstructured tetrahedral elements using $p = 0, 1, 2, 3$. Computed error estimate, e for BDDC, Jacobi (local), and Multigrid preconditioners.

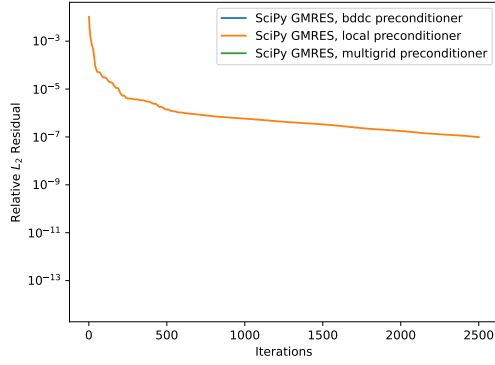
From the figure, we see that the Multigrid preconditioner results in the fewest number of iterations, however this does not account for the computation time required for each iteration. Figure 5 shows the wall clock computation time for each iteration of the GMRES solver for this sphere problem using (a) $p = 0$, (b) $p = 1$, (c) $p = 2$, and (d) $p = 3$. Note that this does not include the time required to construct the preconditioner or consider memory requirements. Timings were performed using a single threaded implementation of the iterative solver using the workstation and software provided in Section 3. Similar to Figure 4, we observe that the Multigrid preconditioner is (slightly) faster than both the BDDC and Jacobi preconditioners.

This may be a consequence of the wavenumber, and by extension wavelength, so we next consider the case where $|\mathbf{k}|$ is increased from 1 to 2 to 4. In these tests, the discretisation of the sphere is kept constant (using radius 1 and 15243 unstructured tetrahedra); the increase in wavenumber is reported in terms of the number of elements per wavelength. Figure 6 shows a comparison for the same sphere discretisation, with $h = 0.08$, where the wavelength is increased from 2π to 4π to 8π between the Multigrid and BDDC preconditioners with respect to the number of iterations required to reach a stopping tolerance of 1×10^{-12} . The figure shows a significant improvement in the required number of iterations, in line with what was reported in [3] for a similar problem for a block Jacobi preconditioner.

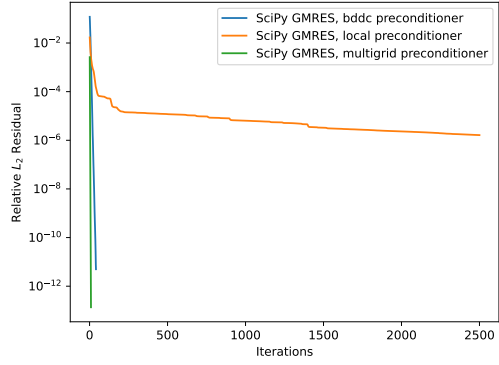
To provide a further examination of the performance of the preconditioners, we examine the (complex) eigenvalues of the preconditioned matrix $\mathbf{P}^{-1}\mathbf{A}$ where \mathbf{P}^{-1} approximates the inverse of \mathbf{A} .

Appendix A. Code for NGSolve and SciPy Solvers

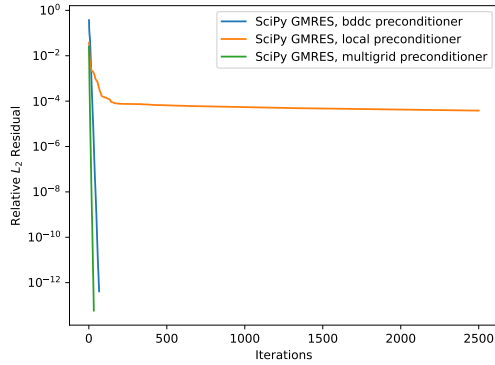
The following two code snippets are equivalent.



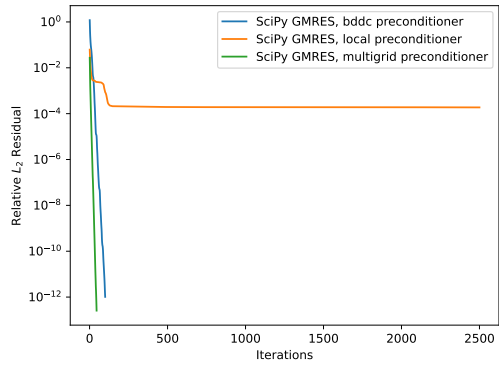
(a)



(b)

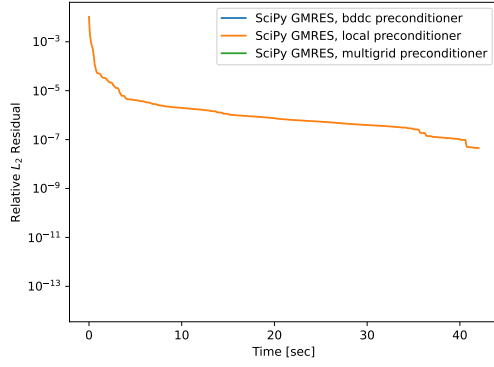


(c)

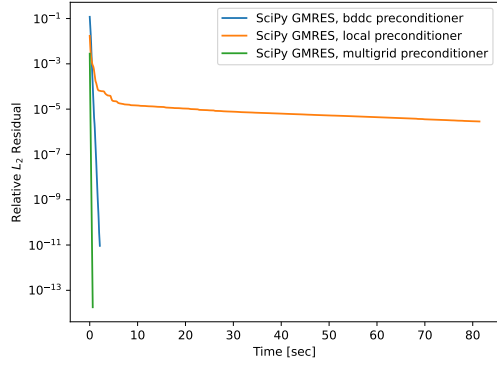


(d)

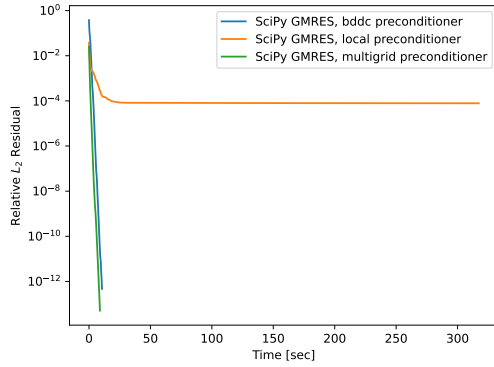
Figure 4: Non-magnetic non-conducting sphere of unit radius discretised using 15243 unstructured tetrahedral elements using $p = 0, 1, 2, 3$. Computed residual to (3) for BDDC, Jacobi (local), and Multigrid preconditioners. Figure shows that both BDDC and Multigrid converge rapidly, and that the local preconditioner does not result in convergence. (a) $p = 0$, (b) $p = 1$, (c) $p = 2$, (d) $p = 3$,



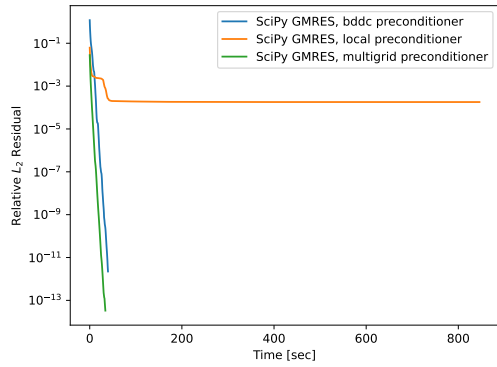
(a)



(b)

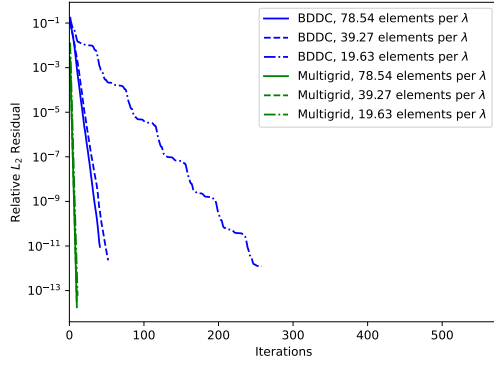


(c)

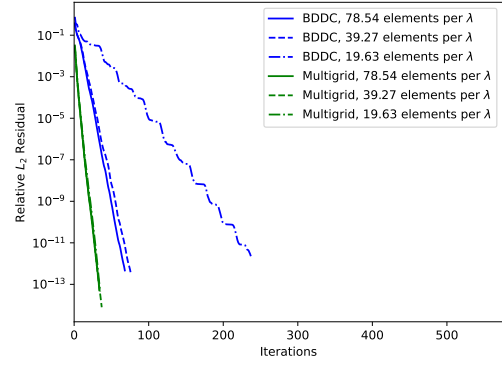


(d)

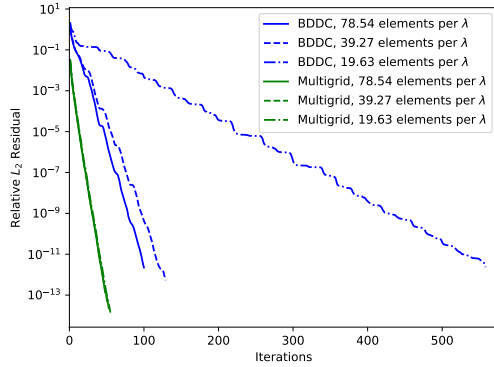
Figure 5: Non-magnetic non-conducting sphere of unit radius discretised using 15243 unstructured tetrahedral elements using $p = 0, 1, 2, 3$. Computation time for BDDC, Jacobi (local), and Multigrid preconditioners. Figure shows that both BDDC and Multigrid converge rapidly, and that the local preconditioner does not result in convergence. (a) $p = 0$, (b) $p = 1$, (c) $p = 2$, (d) $p = 3$,



(a)



(b)



(c)

Figure 6: Non-magnetic non-conducting sphere of unit radius discretised using 15243 unstructured tetrahedral elements using $p = 1, 2, 3$. Number of iterations for BDDC and Multigrid preconditioners when considering $|\mathbf{k}| = 1, 2$, and 4. Figure shows that both BDDC and Multigrid converge rapidly, with the number of iterations required for the BDDC preconditioner reducing as the number of elements per λ increases. (a) $p = 1$, (b) $p = 2$, (c) $p = 3$.

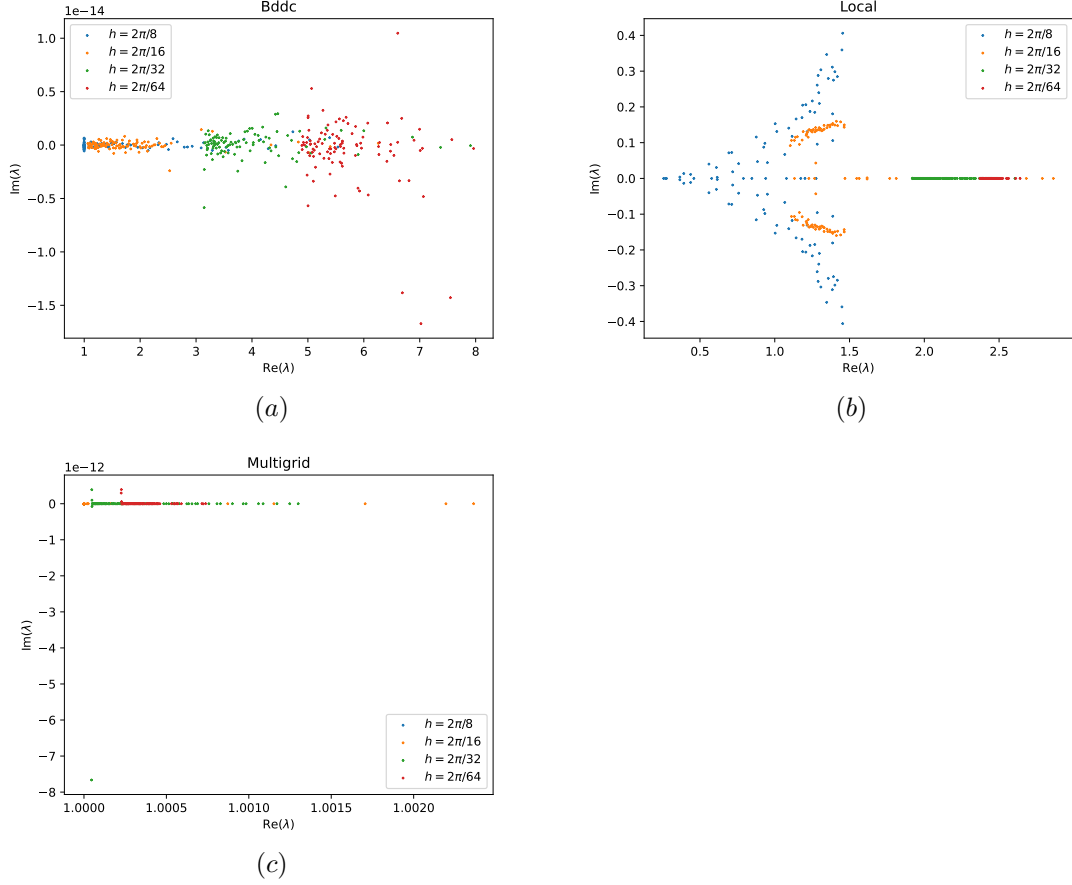


Figure 7: Non-magnetic non-conducting sphere of unit radius discretised using $h = 2\pi/8$, $2\pi/16$, $2\pi/32$, and $2\pi/64$ using $p = 1$. 100 largest magnitude eigenvalues for the preconditioned system matrix for the BDDC, Local, and Multigrid preconditioners when considering $|\mathbf{k}| = 1$. Figure shows an improvement in clustering as h is decreased for both the Local and Multigrid preconditioners, with the opposite effect being observed for the BDDC preconditioner. (a) BDDC, (b) Local, (c) Multigrid.

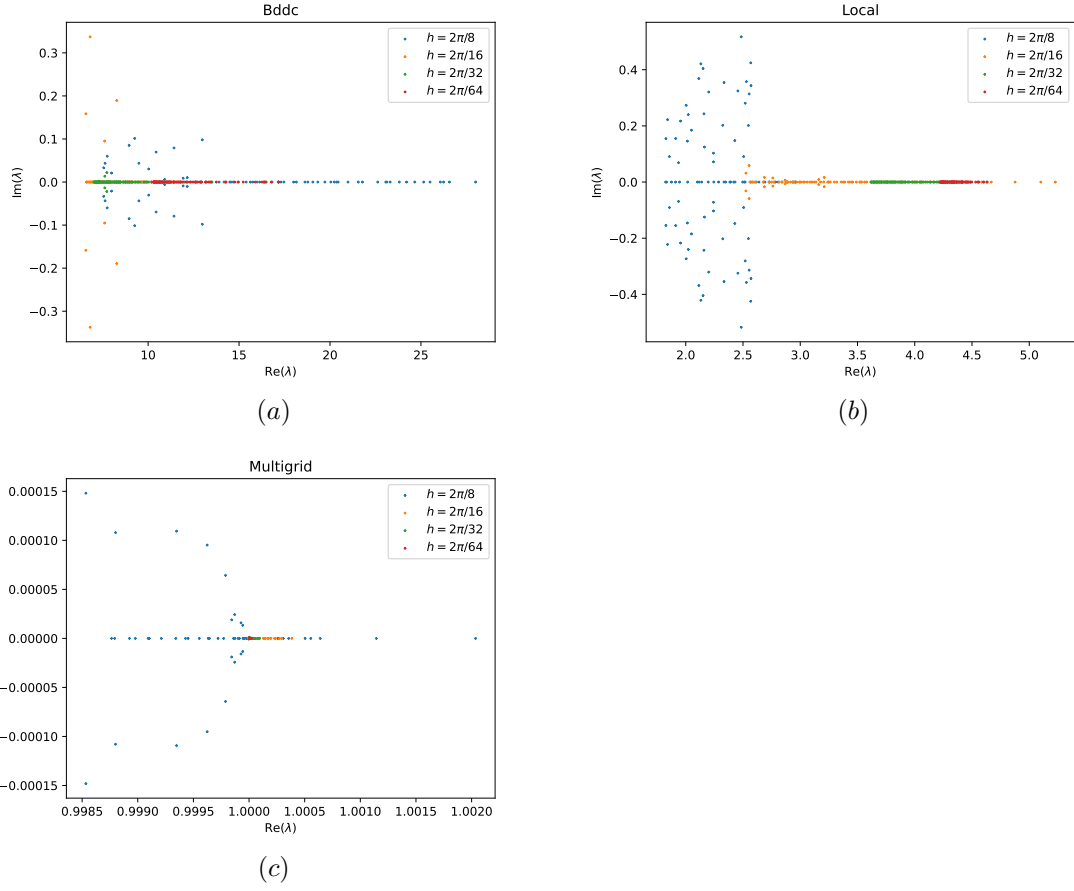


Figure 8: Non-magnetic non-conducting sphere of unit radius discretised using $h = 2\pi/8$, $2\pi/16$, $2\pi/32$, and $2\pi/64$ using $p = 2$. 100 largest magnitude eigenvalues for the preconditioned system matrix for the BDDC, Local, and Multigrid preconditioners when considering $|\mathbf{k}| = 1$. Figure shows an improvement in clustering as h is decreased for both the Local and BDDC preconditioners. (a) BDDC, (b) Local, (c) Multigrid.

```

1 # For assembled linear form f and bilinear form a for solution vector u.
2 r = f.vec.CreateVector()
3 r.data = f.vec - a.mat * u.vec
4 r.data += a.harmonic_extension_trans * r
5
6 # CGSolver(.) * r is equivalent to solving preconditioned Ax=r
7 u.vec.data += CGSolver(A.mat, P.mat, precision=1e-10) * r
8 u.vec.data += a.harmonic_extension * u.vec
9 u.vec.data += a.inner_solve * r

```

Listing 1: Static condensation using NGSolve

```

1 # For assembled linear form f and bilinear form a for solution vector u.
2 r = f.vec.CreateVector()
3 r.data = f.vec - a.mat * u.vec
4 r.data += a.harmonic_extension_trans * r
5
6 # The projector is used to remove non-local and non-Dirichlet degrees of freedom that
7   # should not participate in the solve
8 q = r.CreateVector()
9 pre = Projector(mask=fes.FreeDofs(coupling=True), range=True)
10
11 tmp1 = F.vec.CreateVector()
12 tmp2 = F.vec.CreateVector()
13
14 # A linear operator that returns Av for the sparse matrix A
15 def matvec(v):
16     tmp1.FV().NumPy()[:] = v
17     tmp2.data = A.mat * tmp1
18     tmp2.data = pre * tmp2
19     return tmp2.FV().NumPy()
20
21 r.data = pre * res
22 A_linop = sp.sparse.linalg.LinearOperator((A.mat.height, A.mat.width), matvec)
23
24 # Solve Ax = r
25 q.FV().NumPy()[:], OutputStatus = sp.sparse.linalg.gmres(A_linop, r.FV().NumPy(), tol=1e
26   -10, M=P.mat)
27 u.vec.data += q
28 u.vec.data += a.harmonic_extension * u.vec
29 u.vec.data += a.inner_solve * r

```

Listing 2: Static condensation using SciPy

Appendix B. NGSolve CG and GMRES Comparison

To further justify the swap to a SciPy implementation of the iterative solver, a comparison between the NGSolve conjugate gradients (CG) and GMRES solvers is presented in Figure B.9. The figure shows the effect of p -refinement for the problem (2) as measured using e . The presented example corresponds to a sphere of radius 1, with wavevector $\mathbf{k} = [1, 0, 0]$ and amplitude $\mathbf{p} = [0, 1, 0]$ discretised using a constant $h = 0.08$ and 15251 unstructured tetrahedra. The figure shows that while the NGSolve CG and SciPy GMRES solvers are in agreement with each other, the NGSolve GMRES solver is not.

From the NGSolve documentation [5], the GMRESSolver uses the same syntax as the CGSolver.

References

- [1] GUYAN, R. J. Reduction of stiffness and mass matrices. *AIAA Journal* 3, 2 (1965), 380.
- [2] HARRIS, C. R., MILLMAN, K. J., VAN DER WALT, S. J., GOMMERS, R., VIRTANEN, P., COURCEAU, D., WIESER, E., TAYLOR, J., BERG, S., SMITH, N. J., KERN, R., PICUS, M., HOYER, S., VAN KERKWIJK, M. H., BRETT, M., HALDANE, A., DEL RÍO, J. F., WIEBE, M., PETERSON, P., GÉRARD-MARCHANT, P., SHEPPARD, K., REDDY, T., WECKESSER, W., ABBASI, H., GOHLKE, C., AND OLIPHANT, T. E. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362.
- [3] LEDGER, P. D. Preconditioners for the indefinite linear system arising from the hp discretization of Maxwell's equations. *Communications in Numerical Methods in Engineering* 25 (2009), 275–289.

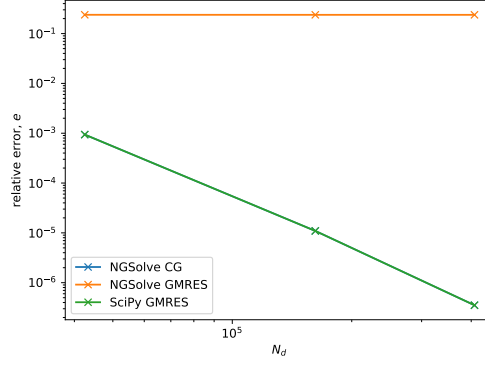


Figure B.9: Non-magnetic non-conducting sphere of unit radius discretised using 15251 unstructured tetrahedral elements using $p = 1, 2, 3$. Comparison between the NGSolve CG and GMRES solvers, and the SciPy GMRES solver using the relative error e using the direct preconditioner. Figure shows that while the NGSolve CG and SciPy GMRES output is in agreement, the NGSolve GMRES is not.

- [4] MONK, P., AND ZHANG, Y. Finite Element Methods for Maxwell's Equations. *Encyclopedia of Computational Mechanics Second Edition* (2017), 1–20.
- [5] NGSOLVE. 5.6.4 feti-dp in ngsolve iv: Inexact feti-dp. https://docu.ngsolve.org/v6.2.2304/i-tutorials/historic/unit-5.5-fetidp_inexact/feti-dp-iv.html?highlight=gmres. Accessed: 31/07/2023.
- [6] SCHÖBERL, J. NETGEN - an advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science* 1(1) (1997), 41–52.
- [7] SCHÖBERL, J. C++11 implementation of finite elements in NGSolve. Tech. rep., ASC Report 30/2014, Institute for Analysis and Scientific Computing, Vienna University of Technology, 2014.
- [8] VIRTANEN, P., GOMMERS, R., OLIPHANT, T. E., HABERLAND, M., REDDY, T., COURNAPEAU, D., BUROVSKI, E., PETERSON, P., WECKESSER, W., BRIGHT, J., VAN DER WALT, S. J., BRETT, M., WILSON, J., MILLMAN, K. J., MAYOROV, N., NELSON, A. R. J., JONES, E., KERN, R., LARSON, E., CAREY, C. J., POLAT, İ., FENG, Y., MOORE, E. W., VANDERPLAS, J., LAXALDE, D., PERKTOLD, J., CIMRMAN, R., HENRIKSEN, I., QUINTERO, E. A., HARRIS, C. R., ARCHIBALD, A. M., RIBEIRO, A. H., PEDREGOSA, F., VAN MULBREGT, P., AND SCI-PY 1.0 CONTRIBUTORS. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272.
- [9] ZAGLMAYR, S. *High Order Finite Elements for Electromagnetic Field Computation*. PhD thesis, Johannes Kepler University Linz, 2006.