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

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

with $\boldsymbol{E}^{(exact)} = e^{\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{x}}$ being the known solution for the electric field at position $\boldsymbol{x}$ for perpendicular wavenumber $\boldsymbol{k} = [k_x, k_y, k_z]$. For the standard $\boldsymbol{H}(\text{curl})$ conforming finite element space [1] the weak form of this problem is: Find $\boldsymbol{u} \in \boldsymbol{H}_D(\text{curl})$ such that

$$\int_\Omega \tag{2}$$

2. Software

`NGSolve` (version 6.2.2302) and `Netgen` (version 6.2.2302) [3, 4, 2],

3. 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 `SciPy` (version 1.10.1) and `NumPy` (version 1.24.2).

The following two code snippets are equivalent.

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

```
# For assembled linear form f and bilinear form a for solution vector u.
r = f.vec.CreateVector()
r.data = f.vec - a.mat * u.vec
r.data += a.harmonic_extension_trans * r

# CGSolver(.) * r is equivalent to solving preconditioned Ax=r
u.vec.data += CGSolver(A.mat, P.mat, precision=1e-10) * r
u.vec.data += a.harmonic_extension * u.vec
u.vec.data += a.inner_solve * r
```

Listing 1: Static condensation using NGSolve

```
# For assembled linear form f and bilinear form a for solution vector u.
r = f.vec.CreateVector()
r.data = f.vec - a.mat * u.vec
r.data += a.harmonic_extension_trans * r

# The projector is used to remove non-local and non-Dirichlet degrees of freedom that
    should not participate in the solve
q = r.CreateVector()
pre = Projector(mask=fes.FreeDofs(coupling=True), range=True)

tmp1 = F.vec.CreateVector()
tmp2 = F.vec.CreateVector()

# A linear operator that returns Av for the sparse matrix A
def matvec(v):
  tmp1.FV().NumPy()[:] = v
  tmp2.data = A.mat * tmp1
  tmp2.data = pre * tmp2
  return tmp2.FV().NumPy()

r.data = pre * res
A_linop = sp.sparse.linalg.LinearOperator((A.mat.height, A.mat.width), matvec)

# Solve Ax = r
q.FV().NumPy()[:], OutputStatus = sp.sparse.linalg.gmres(A_linop, r.FV().NumPy(), tol=1e
    -10, M=P.mat)
u.vec.data += q

u.vec.data += a.harmonic_extension * u.vec
u.vec.data += a.inner_solve * r
```

Listing 2: Static condensation using SciPy

4. GMRES Convergence

4.1. Choice of Preconditioner

5. Numerical Results

5.1. Non-Magnetic Sphere

References

[1] Monk, P., and Zhang, Y. Finite Element Methods for Maxwell's Equations. *Encyclopedia of Computational Mechanics Second Edition* (2017), 1–20.

[2] Schöberl, J. NETGEN - an advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science 1(1)* (1997), 41–52.

[3] 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.

[4] Zaglmayr, S. *High Order Finite Elements for Electromagnetic Field Computation.* PhD thesis, Johannes Kepler University Linz, 2006.