Examination of Preconditioners for Simple Magnetostatic Problems

J. Elgy[a,*]

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

Abstract

Keywords:

1. Introduction

1.1. Model Problem

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

$$\boldsymbol{H}(\text{curl}) := \left\{ \boldsymbol{a} \in \mathrm{L}^2(\Omega)^3 | \nabla \times \boldsymbol{a} \in \mathrm{L}^2(\Omega)^3 \right\},$$

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

$$\boldsymbol{H}_D(\text{curl}) := \left\{ \boldsymbol{a} \in \boldsymbol{H}(\text{curl}) | \boldsymbol{n} \times \boldsymbol{a} = \boldsymbol{n} \times \boldsymbol{E}^{(\text{exact})} \text{ on } \Gamma \right\}$$
$$\boldsymbol{H}_0(\text{curl}) := \left\{ \boldsymbol{a} \in \boldsymbol{H}(\text{curl}) | \boldsymbol{n} \times \boldsymbol{a} = \boldsymbol{0} \text{ on } \Gamma \right\}.$$

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

$$\mathbf{A}\mathbf{q} = \mathbf{r} \tag{1}$$

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.

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

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

**

Email address: `j.elgy@keele.ac.uk` (J. Elgy)

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 (1), the multigrid preconditioner

Algorithm 1 Multigrid preconditioner $\mathbf{P}_l^{-1} : \boldsymbol{r} \rightarrow \boldsymbol{q}$

for $0 \leqslant l \leqslant L$ **do**
  if $l > 0$ **then**
    $\boldsymbol{q} \leftarrow \mathbf{0}$
    $\boldsymbol{q} \leftarrow \boldsymbol{w} + \mathbf{P}_1$
  else
    $\boldsymbol{q} \leftarrow \mathbf{A}_0^{-1}\boldsymbol{r}$
  end if
end for

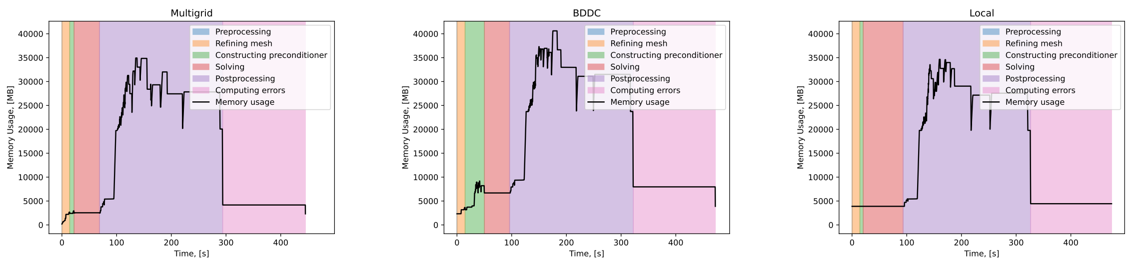
3. Software

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) [4, 6, 3], `SciPy` [5] (version 1.10.1), `NumPy` (version 1.24.2) [2].

4. Results

Example of a unit radius sphere formed on a sequence of meshes, m_0, m_1, m_2, where the subscript refers to the levels of refinement. The sphere is discretised using 12 198, 58 965, and 247 952 unstructured tetrahedral elements with $p = 2$. The following comparison (Figures 1 and 2) compares the computation time and memory usage of the local, BDDC, and multigrid preconditioners on the finest mesh. The figures show that, in this example, the Multigrid preconditioner is slightly faster and requires less memory than the BDDC or Local preconditioners, although, from Figure 1, we observe that the removal of the gradient terms, which involves the inverse of a matrix formed on the finest grid, and the computation of the errors contributes a majority of the time taken.



(*a*) (*b*) (*c*)

Figure 1: Magnetic sphere with unit radius and $\mu_r = 20$ discretised using $p = 2$ and a sequence of meshes with 12 198, 58 965, and 247 952 unstructured tetrahedra. Figure shows memory usage and computation time when considering ((*a*) BDDC, (*b*) Multigrid, and (*c*) local preconditioners.

Given that the postprocessing and construction of the projection operator is independent of the preconditioner, and requires similar computational resources in all 3 examples, future examples will not include this step with the understanding that the error in the curl of $\mathbf{A}$ is unaffected, i.e. the curl of a gradient is zero.

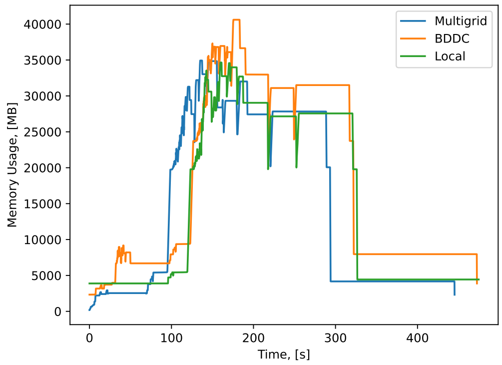


Figure 2: Magnetic sphere with unit radius and $\mu_r = 20$ discretised using $p = 2$ and a sequence of meshes with 12 198, 58 965, and 247 952 unstructured tetrahedra. Figure shows memory usage and computation time when considering each preconditioner.

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

[4] 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.

[5] 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 SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17* (2020), 261–272.

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