Multilevel inverse-based factorization preconditioner for solving sparse linear systems in electromagnetics

Yiming Bu [1,2], Bruno Carpentieri [3], Zhaoli Shen [1,2], and Tingzhu Huang [2]

[1]*Institute of Mathematics and Computer Science, University of Groningen, Groningen, 9712 CP, The Netherlands*
yangyangbu@126.com, z.shen@rug.nl

[2]School of Mathematical Sciences,
University of Electronic Science and Technology of China, Chengdu, 611731, China
tingzhuhuang@126.com

[3]School of Science and Technology,
Nottingham Trent University, Burton Street, Nottingham NG1 4BU, UK
bruno.carpentieri@ntu.ac.uk

***Abstract*—We introduce an algebraic recursive multilevel approximate inverse-based preconditioner, based on a distributed Schur complement formulation. The proposed preconditioner combines recursive combinatorial algorithms and multilevel mechanisms to maximize sparsity during the factorization. Experiments on selected sparse linear systems from electromagnetics applications demonstrate the potential of the proposed method, also against other state-of-the-art solvers.**

***Index Terms*—Approximate inverse preconditioners, Computational Electromagnetics, Krylov subspace methods, Sparse matrices.**

I. INTRODUCTION

We consider multilevel approximate inverse-based factorization preconditioners for solving systems of linear equations

$$Ax = b \tag{1}$$

where $A \in \mathbb{C}^{n\times n}$ is a typically large nonsymmetric sparse matrix arising from finite difference, finite element or finite volume discretization of systems of partial differential equations in electromagnetism applications. Approximate inverse methods directly approximate A^{-1} as the product of sparse matrices, so that the preconditioning operation reduces to forming one (or more) sparse matrix-vector product(s). Due to their inherent parallelism and numerical robustness, this class of methods are receiving renewed consideration for iterative solutions of large linear systems on emerging massively parallel computer systems. In practice, however, some questions need to be addressed. First of all the computed preconditioner could be singular. In the second place, these techniques usually require more CPU-time to compute the preconditioner than Incomplete LU factorization (ILU)-type methods. Third, the computation of the sparsity pattern of the approximate inverse can be problematic, as the inverse of a general sparse matrix is typically fairly dense. This leads to prohibitive computational and storage costs.

In this paper we present experiments with an algebraic recursive multilevel inverse-based factorization preconditioner that attempts to remedy these problems. The solver, proposed in [1], uses recursive combinatorial algorithms to preprocess the structure of A and to produce a suitable ordering of the unknowns of the linear system that can maximize sparsity in the approximate inverse. An efficient tree-based recursive data structure is generated to compute and apply the multilevel approximate inverse fast and efficiently. We assess the effectiveness of the sparse approximate inverse to reduce the number of iterations of Krylov methods for solving matrix problems arising from electromagnetism applications, also against other popular solvers in use today.

II. THE MULTILEVEL FRAMEWORK

We divide the solution of the linear system into the following five distinct phases:

1) a *scale phase*, where the matrix A is scaled by rows and columns so that the largest entry of the scaled matrix has magnitude smaller than one;
2) a *preorder phase*, where the structure of A is used to compute a suitable ordering that maximizes sparsity in the approximate inverse factors;
3) an *analysis phase*, where the sparsity preserving ordering is analyzed and an efficient data structure is generated for the factorization;
4) a *factorization phase*, where the nonzero entries of the preconditioner are actually computed;
5) a *solve phase*, where all the data structures are accessed for solving the linear system.

A. Scale phase.

Prior to solving the system, we scale it by rows and columns to reduce its condition number. We replace system (1) with

$$D_1^{1/2} A y = D_1^{1/2} b, \quad y = D_2^{1/2} x \tag{2}$$

where the $n \times n$ diagonal scaling matrices have the form

$$D_1(i,j) = \begin{cases} \frac{1}{\max\limits_i |a_{ij}|} & , \text{ if i } = \text{ j} \\ \\ 0 & , \text{ if i } \neq \text{ j} \end{cases},$$

$$D_2(i,j) = \begin{cases} \frac{1}{\max\limits_j |a_{ij}|} & , \text{ if i } = \text{ j} \\ \\ 0 & , \text{ if i } \neq \text{ j} \end{cases}.$$

For simplicity, we still refer to the scaled system (2) as $Ax = b$.

B. Preorder phase.

We describe this step using standard notation of graph theory. First, we compute the undirected graph $\Omega(\tilde{A})$ associated with the matrix

$$\tilde{A} = \begin{cases} A, & \text{if A is symmetric,} \\ A + A^T, & \text{if A is unsymmetric.} \end{cases}$$

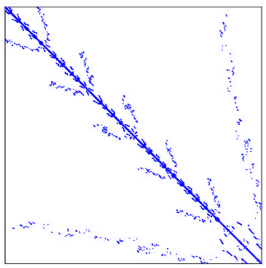
Then, $\Omega(\tilde{A})$ is partitioned into p non-overlapping subgraphs Ω_i of roughly equal size by using the multilevel graph partitioning algorithms available in the Metis package [2]. For each partition Ω_i we distinguish two disjoint sets of nodes: *interior nodes* that are connected only to nodes in the same partition, and *interface nodes* that straddle between two different partitions; the set of interior nodes of Ω_i form a so called *separable* or *independent cluster*. After renumbering the vertices of Ω one cluster after another, followed by the interface nodes as last, and permuting A according to this new ordering, a block bordered linear system is obtained, with coefficient matrix of the form

$$\tilde{A} = P^T A P = \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} B_1 & & & F_1 \\ & \ddots & & \vdots \\ & & B_p & F_p \\ E_1 & \cdots & E_p & C \end{pmatrix}. \tag{3}$$

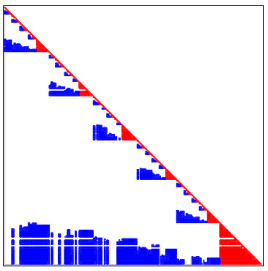
In (3), each diagonal block B_i corresponds to the interior nodes of Ω_i; the blocks E_i and F_i correspond to the interface nodes of Ω_i; the block C is associated to the mutual interactions between the interface nodes. In our multilevel scheme we apply the same block downward arrow structure to the diagonal blocks of $\tilde{A}$ recursively, until a maximum number of levels is achieved or until the blocks at the last level are sufficiently small and easy to factorize. As an example, in Figure 1(a) we show the structure of the general sparse matrix *rdb2048* from Tim Davis matrix collection [3] after three reordering levels.

C. Analysis phase.

The data format for storing the block bordered form (3) of $\tilde{A}$ is defined, allocated and initialized using a tree structure. The root is the whole graph Ω and the leaves at each level are the independent clusters of each subgraph. In other terms, each node of the tree corresponds to one partition Ω_i or equivalently to one block B_i of $\tilde{A}$. The information stored at each node are



(a) The structure of *rdb2048* after permutation.



(b) The structure of the inverse factor. In red are displayed the entries actually stored.

Fig. 1. Structure of the multilevel inverse-based factorization for the matrix *rdb2048*.

the entries of the off-diagonal blocks E and F of $B_i's$ father, and those of the block C of B_i after its permutation, except at the last level of the tree where we store the entire block B. These blocks are stored in sparse format.

D. Factorization phase.

In this phase, we compute the approximate inverse factors $\tilde{L}^{-1}$ and $\tilde{U}^{-1}$ of $\tilde{A}$, which have the following form

$$\tilde{L}^{-1} \approx \begin{pmatrix} U_1^{-1} & & & W_1 \\ & \ddots & & \vdots \\ & & U_p^{-1} & W_p \\ & & & U_S^{-1} \end{pmatrix},$$

$$\tilde{U}^{-1} \approx \begin{pmatrix} L_1^{-1} & & & \\ & \ddots & & \\ & & L_p^{-1} & \\ G_1 & \cdots & G_p & L_S^{-1} \end{pmatrix}$$

where $B_i = L_i U_i$, and

$$W_i = -U_i^{-1} L_i^{-1} F_i U_S^{-1}, \quad G_i = -L_S^{-1} E_i U_i^{-1} L_i^{-1}, \tag{4}$$

and L_S, U_S are the triangular factors of the Schur complement matrix

$$S = C - \sum_{i=1}^{p} E_i B_i^{-1} F_i.$$

During the factorization, fill-in may occur in $\tilde{L}^{-1}$ and $\tilde{U}^{-1}$ but only within the nonzero blocks. Additional sparsity is gained by applying the arrow structure (3) to the diagonal blocks recursively. This can be seen in Figure 1(b). For computing the factorization we only need to invert explicitly the last level blocks and the small Schur complements at each reordering level. The blocks W_i, G_i do not need to be assembled. They may be applied using Eqn (4). For the *rdb2048* problem, in Figure 1(b) we display in red the entries that we actually stored for computing the exact multilevel inverse factorization; these are only 34% of the nonzeros of A.

E. Solve phase.

In the solve phase, the multilevel factorization is applied at every iteration step of a Krylov method for solving the linear system. Notice that the inverse factorization of A may be written as

$$\left(PAP^T\right)^{-1} = \begin{pmatrix} U^{-1} & W \\ 0 & U_S^{-1} \end{pmatrix} \times \begin{pmatrix} L^{-1} & 0 \\ G & L_S^{-1} \end{pmatrix} \quad (5)$$

where $W = -U^{-1}L^{-1}FU_S^{-1}$, $G = -L_S^{-1}EU^{-1}L^{-1}$, and L_S, U_S are the inverse factors of the Schur complement matrix $S = C - EB^{-1}F$.

From Eqn. (5), we obtain the following expression for the exact inverse

$$\begin{pmatrix} B^{-1} + B^{-1}FS^{-1}EB^{-1} & -B^{-1}FS^{-1} \\ -S^{-1}EB^{-1} & S^{-1} \end{pmatrix}. \quad (6)$$

We can derive preconditioners from Eqn. (6) by computing approximate solvers $\tilde{B}^{-1}$ for B and $\tilde{S}^{-1}$ for S. Hence the preconditioner M has the form

$$M = \begin{pmatrix} \tilde{B}^{-1} + \tilde{B}^{-1}F\tilde{S}^{-1}E\tilde{B}^{-1} & -\tilde{B}^{-1}F\tilde{S}^{-1} \\ -\tilde{S}^{-1}E\tilde{B}^{-1} & \tilde{S}^{-1} \end{pmatrix}.$$

and the preconditioning operation $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = M \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ writes as Algorithm 1. As $\tilde{B}$ and $\tilde{S}$ have a block bordered structure, Algorithm 1 is called recursively at lines 1-3 .

Algorithm 1 *Preconditioning operation.*

1: $p_1 = \tilde{B}^{-1}x_1$
2: $[p_2, p_3] = \tilde{S}^{-1}[E \cdot p_1, x_2]$
3: $[p_4, p_5] = \tilde{B}^{-1}[F \cdot p_2, F \cdot p_3]$
4: $y_1 = p_1 + p_4 - p_5$
5: $y_2 = p_3 - p_2$

III. NUMERICAL EXPERIMENTS

We show some preliminary results with the proposed Algebraic Multilevel Explicit Solver (AMES) for solving a set of matrix problems arising from electromagnetics applications [3]. We summarize the list of problems in Table 1. In our experiments, we choose ILUPACK [8] as the local solver in AMES to invert the diagonal blocks at the last level, and the Schur complements at each level. Notice that in this case the entries of the inverse factors are not computed explicitly, and the application of the preconditioner is carried out through a backward and forward substituion procedure. We solve the right preconditioned system $AMy = b$, $x = My$ instead of (1), using restarted GMRES [4] preconditioned by AMES. This combination is tested for different reduction levels for the blocks B_i and also when the Schur complement matrix S is reduced. Finally, we compare AMES against two other popular algebraic preconditioners for linear systems, that are the Algebraic Recursive Multilevel Method (ARMS) by Saad and Suchomel [5] and the SParse Approximate Inverse preconditioner (SPAI) by Grote and Huckle [6], at roughly equal memory costs.[1] We use the zero vector as initial guess in our code, and we terminate the solution process when the norm of residual is below 10^{-12} or the iterations count exceeds 5000. For the performance comparison, we report on the memory ratio $\frac{nnz(M)}{nnz(A)}$, number of iterations (Its), and time costs for performing the preordering phase (t_p), the factorization phase (t_f) and the solving phase (t_s). The experiments are run in double precision floating point arithmetic in Fortran95, on a PC equipped with an Intel(R) Core(TM) i5-3470 running at 3.20GHz and with 8 GB of RAM and 6144KB of cache memory.

Table 1: Set and characteristics of test matrix problems.

Matrix problem	Size	nnz(A)	Field
dw2048	2,048	10,114	Square dielectric waveguide
dw8192	8,192	41,746	Square dielectric waveguide
utm3060	3,060	42,211	Uedge Test Matrix
utm5940	5,940	83,842	Uedge Test Matrix
2cubes_sphere	101,492	874,378	FEM electromagnetics

A. Varying number of reduction levels in AMES.

We consider the *dw2048*, *dw8192* and *2cubes_sphere* problems for these experiments. Increasing the number of levels helps increase sparsity in the factors and this, in turn, may help reduce the number of iterations at similar memory cost. In our experiments, varying the number of levels n_{lev} from 1 to 3 for a given problem, we tuned the dropping threshold to keep roughly the same memory cost in each run, and then we studied the effect on convergence. The results of our experiments, reported in Table 2, show that using more levels enabled us to reduce the number of iterations at similar memory ratio, as we can gain additional sparsity during the factorization. However, the computing time for the preordering phase (t_p) will tend to increase. Probably due to our non-optimized implementation, the solution cost per iteration also tends to increase with the n_{lev}. We conclude that a small number of reduction levels is recommended to use in AMES.

Table 2: Performance of AMES with varying numbers of reduction levels.

Matrix	n_{lev}	$\frac{nnz(M)}{nnz(A)}$	Its	t_p (sec)	t_f (sec)	t_s (sec)	t_{tot} (sec)
dw2048	1	2.37	24	0.023	0.025	0.008	0.056
	2	2.33	22	0.029	0.021	0.011	0.061
	3	2.38	17	0.030	0.021	0.027	0.078
dw8192	1	3.22	87	0.067	0.109	0.312	0.488
	2	3.27	82	0.083	0.128	0.417	0.628
	3	3.28	78	0.092	0.141	0.744	0.977
2cubes_sphere	1	0.31	12	1.271	3.691	0.310	5.272
	2	0.31	12	1.503	2.552	0.598	4.653
	3	0.31	11	2.333	1.829	1.200	5.362

[1] We choose a combination of parameters for AMES, and tune the dropping threshold for ARMS and SPAI to obtain similar memory cost.

B. Varying the number of reduction levels for the Schur complement

The Schur complement matrix S relative to the block C in (3) typically preserves a good deal of sparsity that can be exploited during the factorization by reordering S in a multilevel nested dissection structure, similarly to what is done to the upper leftmost block B. We have implemented this idea at the first permutation level, using ILU factorization as local solver for the reduced Schur complement matrix. We denote by AS_{lev} the number of reduction levels used for the Schur complement. We consider again the *dw2048*, *dw8192* and *2cubes_sphere* problems in these experiments. For a certain test problem, we vary AS_{lev} keeping all the other parameters constant, and we tune the drop tolerance in the ILU factorization to have similar memory costs. The value $AS_{lev} = 0$ means that only the diagonal blocks of the upper-left block B are permuted. Clearly, the max value of AS_{lev} is limited by the size of Schur complement. From Table 3, we see that simultaneous permutation of both the diagonal blocks of B and of the Schur complement S can make the AMES solver more robust to some extent. However, the implementation cost increases and thus, although useful, this option is problem dependent. In our experiments of the coming sections, we select the value for the parameter AS_{lev} that minimizes the total solution cost.

Table 3: Performance of AMES with varying numbers of reduction levels.

Matrix	AS_{lev}	$\frac{nnz(M)}{nnz(A)}$	Its	t_p (sec)	t_f (sec)	t_s (sec)	t_{tot} (sec)
dw2048	0	2.37	24	0.023	0.025	0.008	0.056
	1	2.37	12	0.023	0.027	0.005	0.055
	2	2.37	12	0.024	0.031	0.012	0.067
dw8192	0	3.22	87	0.067	0.109	0.312	0.488
	1	3.26	21	0.067	0.164	0.057	0.288
	2	3.26	18	0.073	0.156	0.060	0.289
2cubes _sphere	0	0.31	12	1.271	3.691	0.310	5.272
	1	0.31	11	1.277	3.974	0.334	5.585
	2	0.31	11	1.288	4.016	0.350	5.654
	3	0.31	11	1.298	3.985	0.355	5.638

C. Comparing AMES against other preconditioners

Table 4: Performance comparison of the multilevel approximate inverse preconditioner against other iterative solvers.

Matrix	Method	$\frac{nnz(M)}{nnz(A)}$	Its	t_p (sec)	t_f (sec)	t_s (sec)	t_{tot} (sec)
dw2048	AMES	2.37	12	0.023	0.027	0.005	0.055
	ARMS	2.39	670	0	0.009	0.081	0.090
	SPAI	2.37	2239	0	0.094	0.367	0.461
dw8192	AMES	3.26	21	0.067	0.164	0.057	0.288
	ARMS	3.37	+5000	0	0.040	+10.89	+10.93
	SPAI	3.33	+5000	0	0.836	+4.841	+5.677
utm3060	AMES	2.79	125	0.077	0.145	0.366	0.588
	ARMS	2.93	402	0	0.030	0.763	0.793
	SPAI	2.88	+5000	0	3.131	+3.095	+6.226
utm5940	AMES	3.50	267	0.147	0.409	2.738	3.294
	ARMS	3.51	1150	0	0.077	5.085	5.162
	SPAI	3.51	+5000	0	11.76	+11.02	+22.78
2cubes _sphere	AMES	0.31	12	1.271	3.691	0.310	5.272
	ARMS	0.32	68	0	0.262	0.986	1.248
	SPAI	0.32	8	0	3.269	0.153	3.422

From Table 4, we can clearly see that the AMES preconditioner shows a good potential of reducing the number of iterations against other state-of-the-art preconditioning techniques at similar memory costs. This result demonstrates the overall good efficiency of the fill reducing strategies implemented in the preconditioner on the selected electromagnetic problems. One exception is the 2cubes_sphere problem which has favourable properties for the SPAI method. For this problem, 71% of the rows of the coefficient matrix are diagonally dominant and 85% of the rows have 80% degree of diagonal dominance. The good decay of the entries away from the diagonal makes this problem suitable for SPAI. The AMES method still remains competitive. However, the pre-processing and solution costs for setting up and applying the multilevel recursive scheme do not pay off in this case.

IV. Conclusions

In this paper a recursive multilevel implementation of factorized sparse approximate inverse preconditioners for Krylov subspace methods is applied to solving sparse matrix problems from electromagnetics. It is widely recognized that preconditioning is an essential component of the numerical solution of linear systems. We used recursive combinatorial techniques to remedy two typical drawbacks of explicit preconditioning, that are lack of robustness and high construction cost. The numerical experiments show that these strategies can improve the performance of conventional approximate inverse methods, yielding iterative solutions that can compete favourably against other popular solvers in use today. Fine-grained blocking, filtering, postfiltering, adaptive pattern selection strategies can be considered for further optimization. Parallelism can be exploited at various levels in our method, alongside other code optimization. The parallel implementation of a fully distributed Schur complement formulation may not be trivial and will be considered in a separate study.

References

[1] Y. Bu, B. Carpentieri, Z. Shen, T.-Z. Huang, "A hybrid recursive multilevel incomplete factorization preconditioner for solving general linear systems," *Applied Numerical Mathematics* vol. 104, pp. 141–157, 2016.

[2] G. Karypis, V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.* vol. 20, pp. 359–392, 1999.

[3] T. Davis: Sparse matrix collection (1994) Available at the URL: http://www.cise.ufl.edu/research/sparse/matrices.

[4] Y. Saad, Iterative Methods for Sparse Linear Systems. SIAM Publications, second edition, 2003.

[5] Y. Saad, B. Suchomel, "ARMS: An algebraic recursive multilevel solver for general sparse linear systems," *Numer. Linear Algebra Appl.* vol. 9, no. 5, pp. 359–378, 2002.

[6] M. Grote, T. Huckle, "Parallel preconditionings with sparse approximate inverses," *SIAM J. Sci. Comput.* vol. 18, pp. 838–853, 1997.

[7] T. Davis, I. Duff, "An Unsymmetric-Pattern Multifrontal Method for Sparse LU Factorization," *SIAM. J. Matrix Anal. & Appl.* vol. 18, pp. 140–158, 1997.

[8] M. Bollhoefer, Y. Saad, O. Schenk: ILUPACK - preconditioning software package, 2010. Available online at the URL: http://ilupack.tu-bs.de.