

# A hybrid recursive multilevel incomplete factorization preconditioner for solving general linear systems

Yiming Bu<sup>1,2</sup>, Bruno Carpentieri<sup>1</sup>, Zhaoli Shen<sup>1,2</sup>, Tingzhu Huang<sup>2</sup>

---

## Abstract

In this paper we introduce an algebraic recursive multilevel incomplete factorization preconditioner, based on a distributed Schur complement formulation, for solving general linear systems. The novelty of the proposed method is to combine factorization techniques of both implicit and explicit type, recursive combinatorial algorithms, multilevel mechanisms and overlapping strategies to maximize sparsity in the inverse factors and consequently reduce the factorization costs. Numerical experiments demonstrate the good potential of the proposed solver to precondition effectively general linear systems, also against other state-of-the-art iterative solvers of both implicit and explicit form.

*Keywords:* linear systems; iterative solvers; preconditioners; sparse approximate inverse methods; multilevel reordering algorithms.

---

## 1. Introduction

Krylov subspace methods may be considered the method of choice for solving large and sparse systems of linear equations arising from the discretization of (systems of) partial differential equations on modern parallel computers. This  
5 class of algorithms are iterative in nature. At every step  $k$ , they compute the approximate solution  $x_k$  of a linear system  $Ax = b$  from the Krylov subspace of

---

<sup>1</sup>Johann Bernoulli Institute for Mathematics and Computing Science - University of Groningen, 9747 AG Groningen, The Netherlands.

<sup>2</sup>School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China.

<sup>3</sup>Corresponding author: Bruno Carpentieri - e-mail: bcarpentieri@gmail.com

dimension  $k$

$$K_k(A, b) = \text{span}\{b, Ab, A^2b, \dots, A^{k-1}b\},$$

according to different criteria for each given method. The computation requires matrix-vector products with the coefficient matrix  $A$  plus vector operations,  
 10 thus potentially reducing the cumbersome costs of sparse direct solvers on large problems, especially in terms of memory. All of the iterative Krylov methods converge rapidly if  $A$  is somehow close to the identity. Therefore, it is common replacing the original system  $Ax = b$  by

$$M^{-1}Ax = M^{-1}b, \tag{1}$$

or

$$AM^{-1}y = b, \quad x = M^{-1}y, \tag{2}$$

15 for a nonsingular matrix  $M \approx A$ . Systems (1) and (2) are referred to as *left* and *right preconditioned* systems, respectively, and  $M$  as the *preconditioner matrix*. In the case  $M$  is factorized as the product of two sparse matrices,  $M = M_1M_2$ , like in the Hermitian and positive definite case, one might solve the modified linear system

$$M_1^{-1}AM_2^{-T}y = M_1^{-1}b, \quad x = M_2^{-T}y. \tag{3}$$

20 If one may choose  $M$  so that  $M^{-1}A$ ,  $AM^{-1}$  or  $M_1^{-1}AM_2^{-T}$  approximate the identity, and linear systems with  $M$  or with  $M_1$  and  $M_2$  as coefficient matrices are easy to invert, it is more efficient to apply a Krylov subspace method to the modified linear system.

Optimal analytic preconditioners based on low order discretizations, nearby  
 25 equations that are simple to solve, or similar ideas have been proposed in the literature for specific problems. However, the problem-specific approach is generally sensitive to the characteristics of the underlying operator and to the details of the geometry. In this study, we pursue an algebraic approach where the preconditioner  $M$  is computed only from the coefficient matrix  $A$ . Although not  
 30 optimal for any specific problem, algebraic methods are universally applicable, they can be adapted to different operators and to changes in the geometry

by tuning a few parameters, and are well suited for solving irregular problems defined on unstructured grids.

Roughly speaking, most of the existing techniques can be divided into  
 35 either implicit or explicit form. A preconditioner of *implicit* form is defined by any nonsingular matrix  $M \approx A$ , and requires to solve an extra linear system with  $M$  at each step of an iterative method. The most important example in this class is represented by the Incomplete  $LU$  decomposition (ILU), where  $M$  is implicitly defined as  $M = \bar{L}\bar{U}$ ,  $\bar{L}$  and  $\bar{U}$  being triangular  
 40 matrices that approximate the exact  $L$  and  $U$  factors of  $A$  according to a prescribed dropping strategy adopted during the Gaussian elimination process. These methods are considered amongst the most reliable in a general setting. Well known theoretical results on the existence and the stability of the factorization can be proved for the class of  $M$ -matrices [35], and recent studies  
 45 are involving more general matrices, both structured and unstructured. The quality of the factorization on difficult problems can be enhanced by using several techniques such as reordering, scaling, diagonal shifting, pivoting and condition estimators (see e.g. [16, 44, 36, 7, 9]). As a result of this active development, in the last years successful results are reported with ILU-type  
 50 preconditioners in many areas that were of exclusive domain of direct solution methods like in circuits simulation, power system networks, chemical engineering plants modelling, graphs and other problems not governed by PDEs, or in areas where direct methods have been traditionally preferred, like in structural analysis, semiconductor device modelling and computational fluid dynamics  
 55 applications (see e.g. [41, 6, 1, 34, 43]). One problem with ILU-techniques is the severe degradation of performance observed on vector, parallel and GPUs machines, mainly due to the sparse triangular solves [33]. In some cases, reordering techniques may help to introduce nontrivial parallelism. However, parallel orderings may sometimes degrade the convergence rate, while more fill-  
 60 in diminishes the overall parallelism of the solver [17].

*Explicit* preconditioning tries to mitigate such difficulties by approximating directly  $A^{-1}$ , as the product  $M$  of sparse matrices, so that the preconditioning

operation reduces to forming one (or more) sparse matrix-vector product, and consequently the application of the preconditioner may be easier to parallelize and numerically stable. Some methods can also perform the construction phase in parallel [23, 10, 26, 37, 38]; additionally, on certain indefinite problems with large nonsymmetric parts, the explicit approach can provide better results than ILU techniques (see e.g. [14, 8, 24]). In practice, however, some questions need to be addressed. The computed matrix  $M$  could be singular, and the construction cost is typically much higher than for ILU-type methods, especially for sequential runs. The main issue is the selection of the non-zero pattern of  $M$ . The idea is to keep  $M$  reasonably sparse while trying to capture the ‘large’ entries of the inverse, which are expected to contribute the most to the quality of the preconditioner. On general problems it is difficult to determine the best structure for  $M$  in advance, and the computational and storage costs required to achieve the same rate of convergence of preconditioners given in implicit form may be prohibitive in practice.

In this study, we present an algebraic multilevel solver for preconditioning general nonsymmetric linear systems which attempts to combine characteristics of both approaches. Assuming that the matrix  $A$  admits the factorization  $A = LU$ , with  $L$  a unit lower and  $U$  an upper triangular matrix, our method approximates the inverse factors  $L^{-1}$  and  $U^{-1}$ . Sparsity in the approximate inverse factors is maximized by employing recursive combinatorial algorithms. Robustness is enhanced by combining the factorization with recently developed overlapping strategies and by using efficient local solvers.

The paper is organized as follows. In Section 2 we describe the proposed multilevel preconditioner. In Section 3 we show how to combine our preconditioner with overlapping strategies, and in Section 4 we assess its overall performance by showing several numerical experiments on realistic matrix problems, also against other state-of-the-art solvers. Finally, in Section 5 we conclude the study with some remarks and perspectives for future work.

## 2. The AMES solver

Let

$$Ax = b \tag{4}$$

be a  $n \times n$  general linear system with nonsingular, possibly indefinite and  
 95 nonsymmetric matrix  $A = \{a_{ij}\} \in \mathbb{R}^{n \times n}$ , and vectors  $x, b \in \mathbb{R}^n$ . We assume  
 that  $A$  admits for a triangular decomposition

$$A = LU$$

and we precondition system (4) as

$$M_L A M_U y = M_L b$$

with  $M_L \approx L^{-1}$  and  $M_U \approx U^{-1}$ , clearly preserving symmetry and/or positive  
 definiteness of  $A$ . This approach of preconditioning linear systems has been  
 100 extensively investigated in a series of papers by Kolotilina and Yeremin [29,  
 30, 32, 31], who prescribed the nonzero pattern of the inverse factors  $M_L$  and  
 $M_U$  of  $A$  in advance equal to the pattern of the lower and upper triangular part  
 of  $A + A^T$ , respectively, and determined the entries of  $M_L$  and  $M_U$  explicitly  
 by solving linear equations involving the principal submatrices of  $A$  (*the ‘FSAI’*  
 105 *preconditioner*). Chow suggested to use as pattern for the inverse factors the  
 structure of the lower and upper triangular part of  $(A + A^T)^p$ , where  $p$  is a  
 positive integer [12, 13, 45]. The larger  $p$ , in general the higher the quality of the  
 computed preconditioner, although the construction, storage and application  
 costs tend to increase rapidly with  $p$ . Blocking and adaptive strategies have  
 110 been recently studied to overcome these problems [26, 18, 25]. Benzi and Tũma  
 proposed to compute the entries of matrices  $M_L$  and  $M_U$  by means of a (two-  
 sided) Gram-Schmidt orthogonalization process with respect to the bilinear form  
 associated with  $A$ , and to determine the best structure for the inverse factors  
 dynamically, during the construction (*the ‘AINV’ preconditioner*). Sparsity is  
 115 preserved in the process by discarding elements having magnitude smaller than  
 a given positive threshold [3, 4].

In this study we analyse multilevel mechanisms, recursive combinatorial algorithms and overlapping techniques, combined with efficient local solvers, to enhance robustness and reduce costs for the approximation of the inverse factors.

120 We refer to the resulting preconditioner as AMES (Algebraic Multilevel Explicit Solver). It is easier to describe the AMES method by using graph notation, dividing the solution of system (4) in five distinct phases:

1. a *scale phase*, where the coefficient matrix  $A$  is scaled by rows and columns so that the largest entry of the scaled matrix has magnitude smaller than  
125 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;
- 130 4. a *factorization phase*, where the nonzero entries of the preconditioner are computed;
5. a *solve phase*, where all the data structures are accessed for solving the linear system.

Below we describe each phase separately.

### 135 2.1. Scale phase.

We initially scale system (4) by rows and columns as

$$D_1^{1/2}Ay = D_1^{1/2}b, \quad y = D_2^{1/2}x, \quad (5)$$

where the  $n \times n$  diagonal scaling matrices  $D_1$  and  $D_2$  have the form

$$D_1(i, j) = \begin{cases} \frac{1}{\max_i |a_{ij}|} & , \text{ if } i = j \\ 0 & , \text{ if } i \neq j \end{cases}, \quad D_2(i, j) = \begin{cases} \frac{1}{\max_j |a_{ij}|} & , \text{ if } i = j \\ 0 & , \text{ if } i \neq j \end{cases}.$$

For simplicity, we still refer in this paper to the scaled system (5) as  $Ax = b$ .

## 2.2. Preorder phase.

140 We use standard notation of graph theory to describe this computational step. We denote by  $\Omega(\tilde{A})$  the undirected graph associated with the matrix

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

First,  $\Omega(\tilde{A})$  is partitioned into  $p$  non-overlapping subgraphs  $\Omega_i$  of roughly equal size by using the multilevel graph partitioning algorithms available in the Metis package [28]. For each partition  $\Omega_i$  we distinguish two disjoint sets of nodes (or  
 145 vertices): *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  $\Omega_i$  form a so called *separable* or *independent cluster*. Upon renumbering the vertices of  $\Omega$  one cluster after another, followed by the interface nodes as last, and permuting  $A$  according to this new ordering, a block bordered  
 150 linear system is obtained, with coefficient matrix of the form

$$\tilde{A} = P^T A P = \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} B_1 & & & F_1 \\ & B_2 & & F_2 \\ & & \ddots & \vdots \\ & & & B_p & F_p \\ E_1 & E_2 & \cdots & E_p & C \end{pmatrix}. \quad (6)$$

In (6), each diagonal block  $B_i$  corresponds to the interior nodes of  $\Omega_i$ , and the blocks  $E_i$  and  $F_i$  correspond to the interface nodes of  $\Omega_i$ ; 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  
 155 diagonal blocks  $B_i$  of  $\tilde{A}$ ; the procedure is repeated recursively until a maximum number of levels is reached, or until the blocks at the last level are sufficiently small to be easily factorized. As an example, in Figure 1(b) we show the structure of the sparse matrix *rd2048* from Tim Davis matrix collection [15] after three reordering levels.

160 To reduce factorization costs, a similar permutation is applied to the Schur

complement matrix  $S = C - EB^{-1}F$  as follows

$$\tilde{S} = \begin{pmatrix} B_{S1} & & & F_{S1} \\ & B_{S2} & & F_{S2} \\ & & \ddots & \vdots \\ & & & B_{Sp} & F_{Sp} \\ E_{S1} & E_{S2} & \cdots & E_{Sp} & C_S \end{pmatrix}. \quad (7)$$

### 2.3. Analysis phase.

In the analysis phase, a suitable data structure for storing the linear system is defined, allocated and initialized. We use a tree structure to store the block bordered form (6) of  $\tilde{A}$ . The root is the whole graph  $\Omega$ , and the leaves at  
165 each level are the independent clusters of each subgraph. Each node of the tree corresponds to one partition  $\Omega_i$  of  $\Omega(\tilde{A})$ , or equivalently to one block  $B_i$  of matrix  $\tilde{A}$ . The information stored at each node are 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  
170 its permutation, except at the last level of the tree where we store the entire block  $B_i$ . All these matrices are represented in the computer memory using a compressed sparse row storage format, except for blocks  $F_i$  that are stored in compressed sparse column format. Blocks  $E_i$  and  $F_i$  can be very sparse; many of their rows and columns can be zero, and this leads to a significant saving of  
175 computation.

### 2.4. Factorization phase.

The approximate inverse factors  $\tilde{L}^{-1}$  and  $\tilde{U}^{-1}$  of  $\tilde{A}$  write in the following form

$$\tilde{L}^{-1} \approx \begin{pmatrix} U_1^{-1} & & & W_1 \\ & U_2^{-1} & & W_2 \\ & & \ddots & \vdots \\ & & & U_p^{-1} & W_p \\ & & & & U_S^{-1} \end{pmatrix}, \quad \tilde{U}^{-1} \approx \begin{pmatrix} L_1^{-1} & & & & \\ & L_2^{-1} & & & \\ & & \ddots & & \\ & & & L_p^{-1} & \\ G_1 & G_2 & \cdots & G_p & L_S^{-1} \end{pmatrix} \quad (8)$$



where

$$B_i = L_i U_i, W_i = -U_i^{-1} L_i^{-1} F_i U_S^{-1}, G_i = -L_S^{-1} E_i U_i^{-1} L_i^{-1} \quad (9)$$

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

$$S = C - \sum_{i=1}^p E_i B_i^{-1} F_i. \quad (10)$$

Some fill-in may occur in  $\tilde{L}^{-1}$  and  $\tilde{U}^{-1}$  during the factorization, but only within the nonzero blocks. This two-level reordering scheme was used in the context of factorised approximate inverse methods for the parallelization of the AINV preconditioner in [2]. Differently from [2], we apply the arrow  
 185 structure (6) recursively to the diagonal blocks and to the first level Schur complement as well, to gain additional sparsity. The multilevel factorization algorithm requires to invert only 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 explicitly, as they may be applied using Eqn (9). For the *rd2048*  
 190 problem, in Figure 1(c) we display in red the actual extra storage required by the exact multilevel inverse factorization in addition to matrix  $A$ ; these represent only 34% of the total nonzeros of  $A$ . From the knowledge of the red entries, the blue ones can be retrieved from Eqn (9), using the off-diagonal blocks of  $A$ . We also permute the large Schur complement at the first level into a block bordered  
 195 structure, until we reach a maximal number of levels or a given minimal size. The last-level matrix is inverted inexactly. An inexact solver is also used to factorize the last-level blocks  $B_i$  in (10).

### 2.5. Solve phase.

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

$$(PAP^T)^{-1} = \begin{pmatrix} U^{-1} & W \\ 0 & U_S^{-1} \end{pmatrix} \times \begin{pmatrix} L^{-1} & 0 \\ G & L_S^{-1} \end{pmatrix} \quad (11)$$

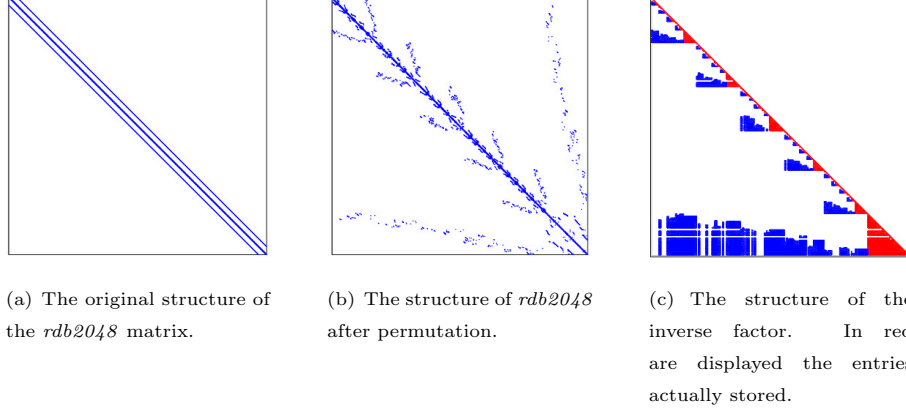


Figure 1: Structure of the multilevel inverse-based factorization for the matrix *rdb2048*.

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. (11), we obtain the following expression for the exact inverse

$$\begin{pmatrix} B^{-1} + B^{-1}FS^{-1}EB^{-1} & -B^{-1}FS^{-1} \\ -S^{-1}EB^{-1} & S^{-1} \end{pmatrix}. \quad (12)$$

205 We can derive preconditioners from Eqn. (12) by computing approximate solvers  $\tilde{B}^{-1}$  for  $B$  and  $\tilde{S}^{-1}$  for  $S$ . Hence the preconditioner matrix  $M$  will have 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.

210 Notice that Algorithm 1 is called recursively at lines 1-3, as  $\tilde{B}$  and  $\tilde{S}$  also have a block bordered structure upon permutation.

---

**Algorithm 1** *The preconditioning operation in the AMES solver.*

---

- 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:  $y_2 = p_3 - p_2$
- 

### 3. Combining the AMES solver with overlapping

In [20], Grigori, Nataf and Qu have introduced an *overlapping technique* to enhance the robustness of multilevel incomplete LU factorization preconditioning computed from matrices reordered in arrow form, e.g. using the nested dissection method by George [19]. The multilevel mechanism incorporated in the AMES preconditioner described in the previous section is based on a nested dissection-like ordering, and thus it can easily accomodate for overlapping. We have tested this idea in our numerical experiments, and in this section we shortly describe the procedure adopted. The results of our experiments are reported in Section 4.

#### 3.1. Background

Let  $\Omega = (V(\Omega), E(\Omega))$  be the graph of  $A$ ,  $V(\Omega)$  denoting the set of vertices and  $E(\Omega)$  the set of edges in  $\Omega$ . If the graph is directed, we denote an edge of  $E$  issuing from vertex  $u$  to vertex  $v$  as  $(u, v)$ ;  $u$  is called a predecessor of  $v$ , and  $v$  a successor of  $u$ . If the graph is undirected, we denote the edges of  $E$  by non-ordered pairs  $\{u, v\}$ ;  $u$  is called a neighbour of  $v$ . As in the previous section, we assume that  $\Omega$  is partitioned into  $p$  independent non-overlapping subgraphs  $\Omega_1, \dots, \Omega_p$ , and we call  $S$  the set of separator nodes, straddling between two different partitions. Goal of overlapping is to extend each independent set of  $\Omega$  by including its direct neighbours, similarly to the overlapping idea used in other domain decomposition methods, for example in the restricted additive Schwarz method [39, 40].

Following [20], we denote by  $V(\Omega_{i,ext})$  the separator nodes that are  
 235 successors of  $\Omega_i$ ,

$$V(\Omega_{i,ext}) = \{v \in V(S) | \exists u \in V(\Omega_i), (u, v) \in E(\Omega)\} \subset V(S), \quad (13)$$

and by  $V(\Omega_{ext})$  the complete set of successor nodes of all the subdomains

$$V(\Omega_{ext}) = \bigcup_{i=1:p} V(\Omega_{i,ext}). \quad (14)$$

Then  $\Omega_i$  is extended to the set  $\hat{\Omega}_i$  as

$$V(\hat{\Omega}_i) = V(\Omega_i) \cup V(\Omega_{i,ext}), \quad i = 1, \dots, p, \quad (15)$$

and the separator  $S$  is extended to  $\hat{S}$  by adding the successors of nodes in  $V(\Omega_{ext})$ , that is

$$V(\hat{S}) = V(S) \cup \{v \in V(\Omega_i), i = 1, \dots, p \mid \exists u \in V(\Omega_{ext}), (u, v) \in E(\Omega)\}. \quad (16)$$

Using this notation, the overlapped graph of  $A$ ,  $\tilde{\Omega} = (V(\tilde{\Omega}), E(\tilde{\Omega}))$ , is introduced as follows. First define the overlapped subgraph  $\tilde{\Omega}_i$  and the overlapped separator  $\tilde{S}$  as, respectively,

$$V(\tilde{\Omega}_i) = \{(x, i) : x \in \hat{\Omega}_i\},$$

$$V(\tilde{S}) = \{(x, s) : x \in \hat{S}\}.$$

240 For simplicity we refer to  $(x, i)$  as  $x_i$ . Then the vertex set  $V(\tilde{\Omega})$  of the overlapped graph  $\tilde{\Omega}$  is formed by the disjoint union of the  $V(\hat{\Omega}_i)$ 's and of  $V(\hat{S})$  as

$$V(\tilde{\Omega}) = \left( \bigcup_{i \in 1:p} V(\hat{\Omega}_i) \right) \cup V(\hat{S}). \quad (17)$$

Recall that, given the union  $B$  of a family of sets indexed by the index set  $I$ ,

$$B = \bigcup_{i \in I} A_i = \bigcup_{i \in I} \{x : x \in A_i\},$$

their disjoint union  $C$  is defined as the set

$$C = \bigcup_{i \in I} \{(x, i) : x \in A_i\}.$$

At this stage, it is useful to introduce the two projection operators  $\Pi_1$  and  $\Pi_2$  such that

$$\Pi_1 : (x, i) \mapsto x$$

and

$$\Pi_2 : (x, i) \mapsto i.$$

With this notation, the set of edges of the overlapped graph  $\tilde{\Omega}$  is defined according to their projection onto the original graph as follows

$$E(\tilde{\Omega}_i) = \{(u_i, v_i) | u_i \in V(\tilde{\Omega}_i), v_i \in V(\tilde{\Omega}_i), (\Pi_1(u_i), \Pi_1(v_i)) \in E(\Omega)\}, \quad (18)$$

$$E(\tilde{S}) = \{(u_s, v_s) | u_s \in V(\tilde{S}), v_s \in V(\tilde{S}), (\Pi_1(u_s), \Pi_1(v_s)) \in E(\Omega)\}, \quad (19)$$

$$E(\tilde{\Omega}_i, \tilde{S}) = \{(u_i, v_s) | u_i \in V(\tilde{\Omega}_i), v_s \in V(\tilde{S}), (\Pi_1(u_i), \Pi_1(v_s)) \in E(\Omega),$$

245

$$\nexists v_i \in V(\tilde{\Omega}_i), (u_i, v_i) \in E(\tilde{\Omega}_i)\}, \quad (20)$$

$$E(\tilde{S}, \tilde{\Omega}_i) = \{(u_s, v_i) | u_s \in V(\tilde{S}), v_i \in V(\tilde{\Omega}_i), (\Pi_1(u_s), \Pi_1(v_i)) \in E(\Omega),$$

$$\nexists v_s \in V(\tilde{S}), (u_s, v_s) \in E(\tilde{S})\}. \quad (21)$$

The following property, established in [20], ensures an equivalence between the equations of the overlapped system and those of the original system.

**Property 1.** *Let  $\Omega$  be the associated directed graph of a given system of linear equations and  $u$  be a vertex of  $V(\Omega)$ . Let  $\tilde{\Omega}$  be the overlapped graph, and let  $u_i$  be a vertex of  $V(\tilde{\Omega})$  such that  $\Pi_1(u_i) = u \in V(\Omega)$ . For each edge  $(u, v) \in E(\Omega)$ , there is a unique  $v_j \in V(\tilde{\Omega})$  such that we have both  $\Pi_1(v_j) = v$  and  $(u_i, v_j) \in E(\tilde{\Omega})$ .*

250

This property shows that there exists a bijection between the nonzeros of  
 255 the equation corresponding to vertex  $u$  in the original system and the nonzeros  
 of the equation corresponding to its dual  $u_i$ , where  $\Pi_1(u_i) = u$ . From a matrix  
 viewpoint, to each nonzero entry  $\tilde{a}_{u_i, v_i}$  in the overlapped matrix there is a unique  
 nonzero entry  $a_{u, v}$  in the original matrix, where  $\Pi_1(u_i) = u$  and  $\Pi_1(v_i) = v$ .  
 Therefore there is a one-to-one correspondence between equations in the original  
 260 system and those in the overlapped system. By solving the overlapped system,  
 we can automatically obtain the solution of the original system.

### 3.2. Example

We display a simple example from [20] to describe shortly how the  
 overlapping procedure works in practice. We consider a  $5 \times 5$  matrix having  
 265 the structure shown in Figure 2(a). The graph consists of two independent  
 subgraphs  $\Omega_1 = \{1, 2\}$ ,  $\Omega_2 = \{3\}$  and one separator  $S = \{4, 5\}$ . We just pick  
 the first subgraph and the separator set to explain. Separator nodes that are  
 successors of  $\Omega_1$  are the set

$$V(\Omega_{1,ext}) = \{4, 5\}$$

and we have

$$V(\hat{\Omega}_1) = V(\Omega)_1 \cup V(\Omega_{1,ext}) = \{1, 2, 4, 5\},$$

270 so that

$$V(\tilde{\Omega}_1) = \{1_1, 2_1, 4_1, 5_1\}.$$

Analogously,

$$V(\Omega_{2,ext}) = \{4, 5\}$$

and

$$V(\Omega_{ext}) = \Omega_{1,ext} \cup \Omega_{2,ext} = \{4, 5\}.$$

Next, we compute the overlapped separator set  $\tilde{S}$ . The vertices of  $V(\Omega_1)$  and  
 $V(\Omega_2)$  directed by  $V(\Omega_{ext})$  are  $\{1, 3\}$ , so

$$V(\hat{S}) = V(S) \cup \{1, 3\} = \{4, 5, 1, 3\}$$

275 and

$$V(\tilde{S}) = \{4_s, 5_s, 1_s, 3_s\}.$$

According to Eqns. (18)-(21), the edges of the overlapped subdomain  $E(\tilde{\Omega}_1)$  are defined based on their projection onto the original graph. The first diagonal block of the overlapped matrix is formed by picking the  $V(\hat{\Omega}_1) = \{1, 2, 4, 5\}$  rows and columns of the original matrix

$$\begin{array}{c} 1 \quad 2 \quad 4 \quad 5 \\ 1 \begin{pmatrix} \diamond & \diamond & \diamond & \diamond \\ 2 \begin{pmatrix} \diamond & \diamond & & \\ 4 \begin{pmatrix} \diamond & & \diamond & \\ 5 \begin{pmatrix} \diamond & & & \diamond \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix}.$$

Similarly for the other two diagonal blocks, and this is shown in Figure 2(b).

From Eqn. (20), we can construct the edges from  $\tilde{\Omega}_1$  to  $\tilde{S}$ . These are the nonzero entries of the overlapped interface block  $\tilde{F}_1$ , adopting the same notation as in (6). We pick the  $V(\hat{\Omega}_1) = \{1, 2, 4, 5\}$  rows and  $V(\hat{S}) = \{4, 5, 1, 3\}$  columns of the original matrix, and we set the columns corresponding to the common vertexes of  $\hat{\Omega}_1$  and  $\hat{S}$  to zeros. In our example this results in zeroing out the columns of  $\hat{F}_1$  indexed by  $\hat{\Omega}_1 \cup \hat{S} = \{4, 5, 1\}$ , giving

$$\begin{array}{c} 4 \quad 5 \quad 1 \quad 3 \\ 1 \begin{pmatrix} \times & \times & \diamond & \\ 2 \begin{pmatrix} & & \diamond & \\ 4 \begin{pmatrix} \star & & \times & \times \\ 5 \begin{pmatrix} & \times & \times & \times \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix} \longrightarrow \begin{array}{c} 4 \quad 5 \quad 1 \quad 3 \\ 1 \begin{pmatrix} & & & \\ 2 \begin{pmatrix} & & & \\ 4 \begin{pmatrix} & & & \times \\ 5 \begin{pmatrix} & & & \times \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix}.$$

Similar procedure is followed for the other blocks  $\tilde{F}_i, \tilde{E}_i$ . Finally, the overlapped matrix has the form given in Figure 2(b). The block arrow structure of the original matrix is preserved. However, symmetry is lost and the sparsity pattern also changes significantly.

280

### 3.3. Analysis

It is interesting to analyse the effect that overlapping may produce on

$$\left[ \begin{array}{cc|c|cc} a_{11} & a_{12} & & a_{14} & a_{15} \\ a_{21} & a_{22} & & & \\ \hline & & a_{33} & a_{34} & a_{35} \\ \hline a_{41} & & a_{43} & a_{44} & \\ a_{51} & & a_{53} & & a_{55} \end{array} \right]$$

(a) The original matrix

$$\left[ \begin{array}{cccc|ccc|ccc} a_{11} & a_{12} & a_{14} & a_{15} & & & & & & \\ a_{21} & a_{22} & & & & & & & & \\ a_{41} & & a_{44} & & & & & & a_{43} & \\ a_{51} & & a_{55} & & & & & & a_{53} & \\ \hline & & & & a_{33} & a_{34} & a_{35} & & & \\ & & & & a_{43} & a_{44} & & & a_{41} & \\ & & & & a_{53} & & a_{55} & & a_{51} & \\ \hline & & & & & & & a_{44} & a_{41} & a_{43} \\ & & & & & & & & a_{55} & a_{51} & a_{53} \\ & a_{12} & & & & & & a_{14} & a_{15} & a_{11} & \\ & & & & & & & a_{34} & a_{35} & & a_{33} \end{array} \right]$$

(b) The matrix after one-level overlapping

Figure 2: Matrix structure before and after applying the overlapping procedure.



the AMES method. According to (15) and (18), the size and the number of nonzeros in each subgraph is increased after overlapping. According to (20), the interconnections between subdomains and separator are reduced in the overlapped system, as the original interconnectivities are all removed. The more nodes are added to the subgraphs, the richer they are in terms of information about the system matrix, and a larger performance improvement may be expected. In the overlapped system, each block  $\tilde{B}_i$  has the following structure

$$\begin{array}{cc} & V(\Omega_i) & V(\Omega_{i,ext}) \\ \begin{array}{c} V(\Omega_i) \\ V(\Omega_{i,ext}) \end{array} & \left( \begin{array}{cc} B & E(\Omega_i, \Omega_{i,ext}) \\ E(\Omega_{i,ext}, \Omega_i) & E(\Omega_{i,ext}) \end{array} \right). \end{array}$$

From Eqn. (13) we see that the set of neighboring nodes  $V(\Omega_{i,ext})$  corresponds to the nonzero columns of the block  $F_i$ , and the nonzero elements of  $F_i$  are determined by the set of interconnections  $E(\Omega_i, \Omega_{i,ext})$ . Therefore, the more dense and larger the blocks  $F_i, i = 1 : p$ , (that is, the size of separator) in the original matrix, the more nodes and interconnections are added to subdomains, and a larger reduction of the number of iterations can be achieved.

### 3.4. Algorithmics

The AMES preconditioning algorithm described in Section 2 with one extra overlapping phase writes as follows:

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 can maximize sparsity in the approximate inverse factors;
3. an *overlap phase*, which extends each block  $B_i$  and the Schur complement, and generates the overlapped matrix  $\tilde{A}$  and the right-hand side vector  $\tilde{b}$ ;
4. an *analysis phase*, where the sparsity preserving and overlapping orderings are analyzed and an efficient data structure is generated for the factorization;

5. a *factorization phase*, where the entries of  $\tilde{A}$  are processed to explicitly  
 310 compute the approximate inverse factors;
6. a *solve phase*, that accesses all the data structures for solving the  
 overlapped linear system.
7. a *restriction phase*, that restricts the solution obtained from the  
 overlapped system to the original system, and obtains the solution.

315

#### 4. Numerical experiments

In this section we present the results of our numerical experiments to illustrate the performance of the AMES preconditioner, also against other state-of-the-art methods and software for solving general linear systems. The selected  
 320 matrix problems are extracted from the public-domain matrix repository available at the University of Florida [15], and arise from various application fields. We present a summary of the characteristics of each linear system in Table 1. We applied AMES as a preconditioner for the Generalized Minimal Residual (GMRES) method by Saad and Schultz [42]. In all our runs we  
 325 started the iterative solution from the zero vector and we stopped it when either the initial residual was reduced by twelve orders of magnitude or when no convergence was achieved after 5000 matrix-vector products. To limit memory costs, we restarted the GMRES algorithm every 500 iterations. The right-hand side  $b$  of the linear system was chosen so that the solution is the vector of all  
 330 ones, that is  $b = Ae$  with  $e = [1, \dots, 1]^T$ . In each run we recorded the following performance measures:

1. the density ratio  $\frac{nnz(M_L + M_U)}{nnz(A)}$ , that is the ratio between the number of  
 nonzeros in the preconditioner matrix  $M = M_U M_L$  versus the number of  
 nonzeros in the coefficient matrix  $A$ ;
- 335 2. the number of iterations  $Its$  required to reduce the initial residual by 12  
 orders of magnitude starting from the zero vector;

Matrix problem	$n$	Field	$nnz(A)$
orsirr_1	1,030	Oil reservoir simulation	6,858
1138.bus	1,138	Bus Power System	4,054
bcsstk27	1,224	BCS Structural Engineering Matrix	28,675
epb0	1,794	Plate-fin heat exchanger	7,764
cz20468	20,468	Closest Point Method	206,076
raefsky3	21,200	Fluid Structure Interaction	1,488,768
ABACUS_shell_ud	23,412	ABAQUS benchmark	218,484
sme3Db	29,067	3D structural mechanics problem	2,081,063
viscoplastic2	32,769	FEM discretization	381,326
cz40948	40,948	Closest Point Method	412,148
rma10	46,835	3D CFD Model	2,374,001
finan512	74,752	Portfolio optim	596,992
helm2d03	392,257	Helmholtz eq. on a unit square	2,741,935
parabolic_fem	525,825	Parabolic FEM	3,674,625

Table 1: Set and characteristics of the test matrix problems.

3. the CPU time cost in seconds for completing the preorder phase (denoted by  $t_p$ ), for constructing the approximate inverse factorization ( $t_f$ ), and for solving the linear system ( $t_s$ ). Symbol “-” means that the corresponding phase does not apply to the given run. For example, some of the preconditioners used for the comparison against our method do not have a preorder phase.

The codes were developed in Fortran 95 and the experiments were run in double precision floating point arithmetic on a PC equipped with an Intel(R) Core(TM)2 Duo CPU E8400, 3.00 GHz of frequency, 4 GB of RAM and 6144 KB of cache memory. In the coming sections we study the effect of using different parameter settings, and we illustrate the overall performance on the selected matrix problems.

#### 4.1. Performance of the multilevel mechanism

350

The AMES method can be seen as a multilevel generalization of factorized approximate inverse techniques such as the FSAI preconditioner by Kolotilina and Yeremin, and the AINV preconditioner by Benzi and Tũma, that we recalled in Section 2. Therefore, first we present some comparison between  
355 these methods, to show the benefit of the multilevel mechanism. The results are reported in Table 2. For these runs, we considered four matrix problems from Table 1, that are *orsirr\_1*, *1138\_bus*, *bcsstk27* and *epb0*. In our AMES solver, we inverted the last level blocks using ILU, FSAI and AINV factorizations. For ILU, we used the multilevel implementation available in the ILUPACK package [5]  
360 (this combination is referred to as *AMES\_ILU* in the table). For FSAI, we used the structure of the nonzero pattern of the lower (resp. upper) triangular part of the symmetrized block for the approximate inverse factors, and also the square of this pattern (this combination is referred to as *AMES\_FSAI*). Finally, for AINV we used the implementation kindly provided by the authors  
365 (this combination is referred to as *AMES\_AINV*). The dropping threshold value selected for the *AINV*, *AMES\_AINV* and *AMES\_ILU* methods (referred to as *Drop* in the Table) is an absolute value, and was computed so that the resulting preconditioners had roughly equal memory cost. We used the default value for the parameter *condest* = 10 (norm bound for the inverse factors  $L^{-1}$  and  $U^{-1}$ )  
370 in ILUPACK.

In our runs, the multilevel variants *AMES\_FSAI* and *AMES\_AINV* performed consistently better than the *FSAI* and *AINV* solvers in terms of convergence rate and storage cost. This is probably due to the multilevel mechanism that enabled us to exploit sparsity in the inverse factors more  
375 effectively. The best solutions with AMES were obtained using ILU as local solver, while the threshold-based dropping rules of the AINV method often computed a better pattern for the approximate inverse factors than the static approach used in the FSAI method. We can see evidence of this behaviour in Figures 3 - 6, where for one of the last-level blocks of the permuted coefficient

380 matrix (6) we compare the structure of its exact inverse factor  $L^{-1}$ , and of  
 the approximations  $M_L$  and  $W^T$  of  $L^{-1}$  as computed by, respectively, the  
*AMES\_FSAI* code using the square of the pattern of the symmetrized block,  
 and by the *AMES\_AINV* code. Large to small entries are depicted in different  
 colors, from red to orange and yellow. The approximation is good for the  
 385 *1138\_bus* problem (Figure 3) but poor for the *orsirr\_1* matrix (Figure 4), and  
 this is confirmed by the different convergence results for the two problems,  
 reported in Table 2. On some larger problems, like the *cz40948* and the  
*ABACUS\_shell\_ud* problems, shown in Figures 5 - 6, we found that  $L^{-1}$  had  
 no evident structure; in this case we had to increase the number of nonzeros in  
 390  $M_L$  and  $W^T$  significantly to converge. For example on the *ABACUS\_shell\_ud*  
 problem, *AMES\_AINV* converged in 468 iteration with  $\frac{n\text{nz}(Z+W)}{n\text{nz}(A)} = 11.6$  while  
*AMES\_FSAI* did not converge at this value of density. In these situations,  
 uniformly better convergence were obtained using ILU as local solver. We will  
 focus mostly on this choice of local solver for the experiments of this paper.  
 395 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 substitution procedure. Other options may be considered  
 for the last level solver, such as the ARMS method [44] and enhanced FSAI  
 methods [27], but these are not included in the presented analysis.

#### 400 4.2. Varying the number of independent clusters at the first level

We considered three matrix problems in our runs: *cz20468*,  
*ABACUS\_shell\_ud* and *cz40948*. In Table 3 we show the results varying the  
 number of independent clusters  $p$  at the first level of reordering of  $A$  in (6). For  
 each problem, we used the same number of levels  $n_{lev}$  in the AMES structure,  
 405 and tuned the drop tolerance in the local ILU factorization to keep the memory  
 ratio  $\frac{n\text{nz}(M_L+M_U)}{n\text{nz}(A)}$  roughly constant while increasing  $p$  in different runs. Clearly,  
 larger  $p$  results in more independent clusters of smaller size, and in larger Schur  
 complement matrices. In the table we report the ratio  $\frac{\text{size}B}{\text{size}A_S}$  between the  
 average size of the independent clusters  $B_i$  and the size of the Schur complement

(a) orsirr.1							
Method	Pattern	Drop/condest	<i>Its</i>	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$t_p$	$t_f$	$t_s$
<i>AMES_FSAI</i>	$A + A^T$	-	273	1.42	0.011	0.023	0.22
<i>FSAI</i>			304	1.45	-	0.070	0.23
<i>AMES_FSAI</i>	$(A + A^T)^2$	-	217	3.43	0.013	0.035	0.17
<i>FSAI</i>			236	3.76	-	0.088	0.16
<i>AMES_AINV</i>	-	0.03	67	2.27	0.016	0.014	0.034
<i>AINV</i>		0.07	80	2.22	-	0.016	0.024
<i>AMES_ILU</i>	-	8e-3/10	31	1.24	0.012	0.013	7.4e-3
(b) 1138_bus							
Method	Pattern	Drop/condest	<i>Its</i>	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$t_p$	$t_f$	$t_s$
<i>AMES_FSAI</i>	$A + A^T$	-	7	2.24	5.2e-3	0.032	1.2e-3
<i>FSAI</i>			9	2.32	-	0.074	8.9e-4
<i>AMES_FSAI</i>	$(A + A^T)^2$	-	5	2.70	5.0e-3	0.035	1.0e-3
<i>FSAI</i>			6	2.88	-	0.077	6.4e-4
<i>AMES_AINV</i>	-	0.6	13	2.85	7.0e-3	2.0e-3	1.9e-3
<i>AINV</i>		0.7	16	2.88	-	6.0e-3	3.2e-3
<i>AMES_ILU</i>	-	0/10	1	1.00	5.1e-3	3.9e-3	7.0e-4

410 at the first level. Increasing  $p$  reduces in turn  $\frac{sizeB}{sizeA_S}$  to values smaller than 1. Using ILU as local solver, the best convergence results were obtained when  $\frac{sizeB}{sizeA_S} \approx 1$ . Our experiments indicate that for good performance the size of each independent cluster should be approximately equal to that of the Schur complement.

#### 415 4.3. Varying the number of reduction levels for the diagonal blocks

We consider again matrices *cz40948*, *ABACUS-shell\_ud* and *cz20468* for our tests. We varied the number of levels  $n_{lev}$  from 1 to 3 in the multilevel reordering of the diagonal blocks. In each run we tuned the dropping threshold

(c) bcsstk27							
Method	Pattern	Drop/condest	<i>Its</i>	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$t_p$	$t_f$	$t_s$
<i>AMES.FSAI</i>	$A + A^T$	-	8	0.90	0.062	0.041	0.021
<i>FSAI</i>			19	1.27	-	0.20	4.1e-3
<i>AMES.FSAI</i>	$(A + A^T)^2$	-	5	1.16	0.063	0.071	0.018
<i>FSAI</i>			13	2.72	-	0.47	3.7e-3
<i>AMES.AINV</i>	-	1e-3	6	1.18	0.055	0.040	7.3e-3
<i>AINV</i>		0.06	16	0.98	-	0.063	5.7e-3
<i>AMES.ILU</i>	-	0.01/10	6	0.978	0.059	0.016	0.010

(d) epb0							
Method	Pattern	Drop/condest	<i>Its</i>	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$t_p$	$t_f$	$t_s$
<i>AMES.FSAI</i>	$A + A^T$	-	277	1.67	0.020	0.011	0.66
<i>FSAI</i>			400	1.69	-	0.19	0.59
<i>AMES.FSAI</i>	$(A + A^T)^2$	-	161	4.32	0.021	0.023	0.40
<i>FSAI</i>			290	4.81	-	0.23	0.27
<i>AMES.AINV</i>	-	0.5	132	3.32	0.024	4.5e-3	0.21
<i>AINV</i>		0.9	347	4.26	-	0.015	0.42
<i>AMES.ILU</i>	-	0.1/10	7	1.848	0.020	4.1e-3	0.019

Table 2: Numerical experiments on selected matrix problems illustrating the performance of the multilevel sparse approximate inverse preconditioner AMES against the factorized approximate inverse methods FSAI and AINV.

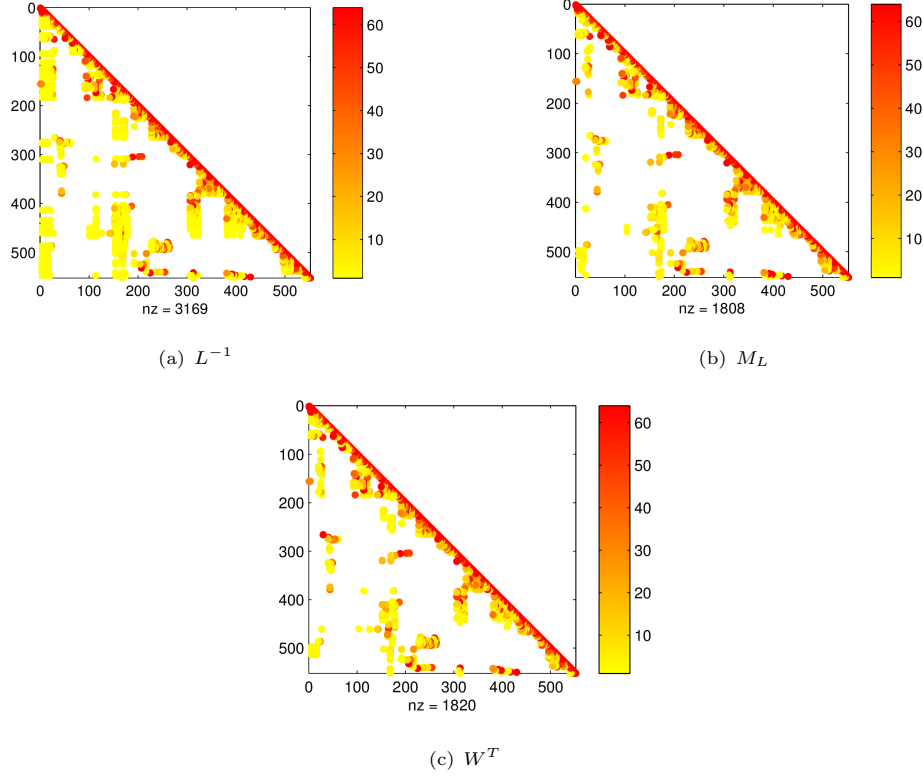


Figure 3: The exact (left) and approximate (middle and right) inverse lower triangular factors of the *1138\_bus* matrix.

parameter to have roughly the same memory cost in the experiments for each  
 420 matrix. We chose the value of  $p$  for each problem so that  $\frac{sizeB}{sizeA_S} \approx 1$  as reported  
 in Section 5.2. The last level blocks were factorized using ILUPACK [5]. The  
 results reported in Table 4 show that using more levels can reduce the number  
 of iterations for similar memory ratio as we can gain additional sparsity during  
 the factorization. However, probably due to our non optimized implementation,  
 425 the solution cost tends to increase. From our experiments, a small number of  
 reduction levels is recommended to use.



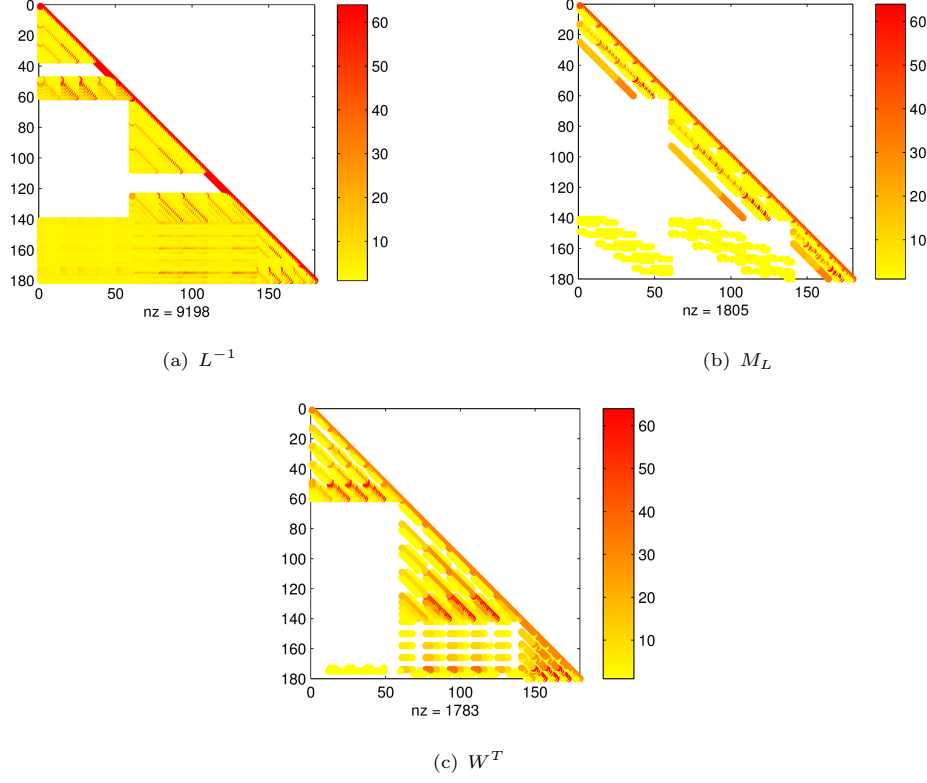


Figure 4: The exact (left) and approximate (middle and right) inverse lower triangular factors of the *orsirr\_1* matrix.

#### 4.4. Varying the number of reduction levels for the Schur complement

The Schur complement matrix relative to the block  $C$  in (6) typically preserves a good deal of sparsity, and this can be further exploited during the factorization by applying, e.g., the multilevel nested dissection reordering to  $A_S$ , similarly to what is done to the upper leftmost block  $B$ . We implemented this idea at the first permutation level, using ILU factorization as local solver and selecting the same values of  $p$  and  $n_{lev}$  for each matrix problem. We tuned the drop tolerance in the ILU factorization to have roughly the same memory costs in different runs. We varied  $n_{levAS}$  from 0 to 3 ( $n_{levAS} = 0$  means that only the diagonal blocks of the upper-left block  $B$  are permuted). The results reported

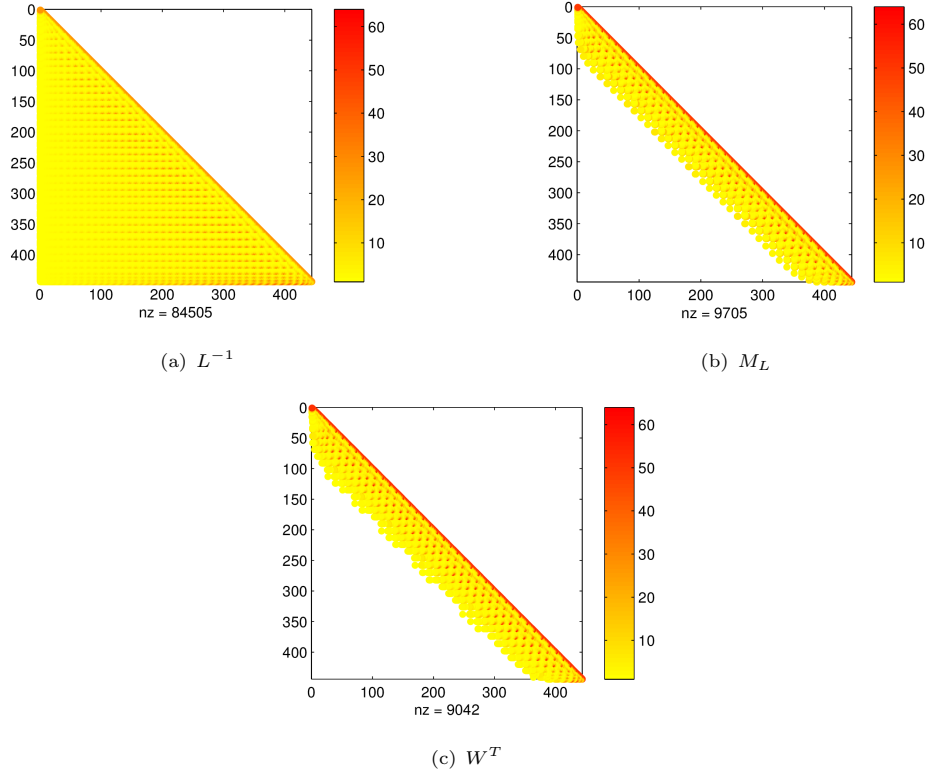


Figure 5: The exact (left) and approximate (middle and right) inverse lower triangular factors of the *cz40948* matrix.

in Table 5 show that the simultaneous permutation of both the diagonal blocks of  $B$  and of the Schur complement can make the preconditioner more robust. We adopted this strategy in the experiments illustrated in the coming sections, selecting in each run the value of  $n_{levAS}$  that minimized the total solution cost.

#### 4.5. Comparison against other solvers

We compared the performance of the AMES preconditioner against other three popular algebraic preconditioners for solving linear systems, that are the ILUPACK solver developed by Bollhöfer and Saad [5], the Algebraic Recursive Multilevel Method (ARMS) proposed by Saad and Suchomel [44], and the SParse Approximate Inverse preconditioner (SPAI) introduced by Grote and

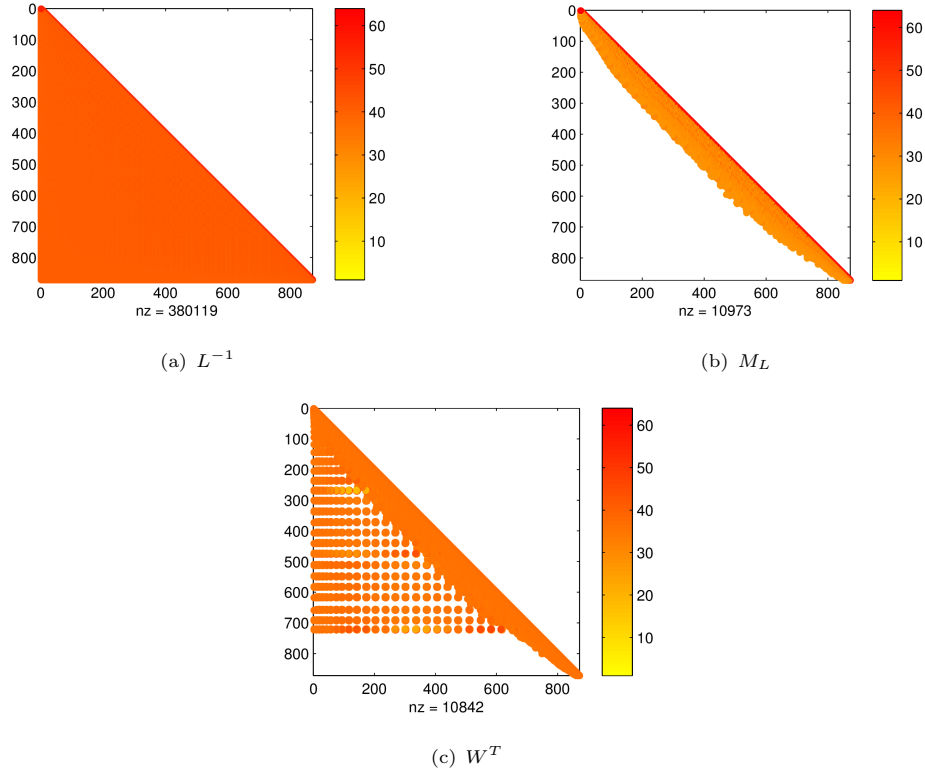


Figure 6: The exact (left) and approximate (middle and right) inverse lower triangular factors of the *ABACUS\_shell.ud* matrix.

Huckle [21]. As in the previous experiments, for each run we recorded the CPU time from the start of the solution until the initial residual was reduced by 12 orders of magnitude or until the process failed. We declared a solver failure  
450 when no convergence was achieved after 5000 iterations of the restarted GMRES method. We selected the parameters carefully to have a fair comparison between different methods. In AMES, following our conclusions from Section 4.2, we selected the number of blocks  $B_i$  at the first level so that their average size is almost equal to the size of the Schur complement. For every problem we  
455 tested different combinations of number of levels  $n_{lev}$  of recursive factorization and different values for the dropping threshold parameter *droptol* for the

Matrix	$p$	$Its$	$t_p$	$t_f$	$t_s$	$t_{per\_it}$	$\frac{sizeB}{sizeA_S}$
cz20468	15	151	0.4	0.3	3.0	0.020	4.3
	20	139	0.4	0.3	2.8	0.020	2.5
	30	131	0.5	0.4	2.5	0.019	1.1
	40	137	0.5	0.6	2.6	0.019	0.6
ABACUS_shell_ud	4	258	0.5	1.2	9.1	0.035	7.8
	6	242	0.5	1.2	8.3	0.034	3.8
	12	213	0.5	2.0	6.8	0.032	1.0
	15	253	0.5	2.1	8.7	0.034	0.7
cz40948	15	219	1.1	0.5	10.8	0.049	8.7
	30	212	1.1	0.7	10.0	0.047	2.2
	45	198	1.1	1.2	9.2	0.046	0.9
	50	207	1.1	1.9	9.8	0.047	0.5

Table 3: The best performance of the multilevel sparse approximate inverse preconditioner are observed when  $\frac{sizeB}{sizeA_S} \approx 1$ .

factorization of the last level blocks  $B_i$  and of the Schur complements. We chose the best combination in terms of memory and time to solution costs for the given problem. Then we tuned the value of the dropping threshold in the ILUPACK, ARMS, SPAI and AINV solvers to have roughly equal memory costs as in AMES, setting the other parameters equal to their default values defined in those packages. The performance of these methods is rather sensitive to the dropping threshold parameter. For example, on the *rma10* problem, ILUPACK converged in only 9 iterations using the default value  $droptol = 0.01$ , but the computation costed  $\frac{nnz(M)}{nnz(A)} = 8.9$  and  $t_f = 45s$ ; ARMS converged in 26 iterations with the default  $droptol = 0.001$ , costing  $\frac{nnz(M)}{nnz(A)} = 33.9$  and  $t_f = 1111s$ ; and SPAI could not converge in 5000 iterations with  $\frac{nnz(M)}{nnz(A)} = 0.19$ , using the default value  $droptol = 0.6$ . The number of levels of recursive factorization in the multilevel methods ILUPACK and ARMS are calculated automatically by the original codes developed by their authors. We point out that the performance

Matrix	$n_{lev}$	$Its$	$t_p$	$t_f$	$t_s$	$t_{per\_it}$
cz20468	1	113	0.5	0.4	1.9	0.017
	2	80	0.5	0.5	2.3	0.028
	3	71	0.6	0.5	4.1	0.058
ABACUS_shell.ud	1	388	0.5	1.7	17.6	0.045
	2	381	0.5	1.9	21.9	0.057
	3	294	0.6	1.9	22.7	0.077
cz40948	1	198	1.1	1.2	9.2	0.046
	2	154	1.2	1.3	10.4	0.068
	3	133	1.3	1.3	17.3	0.13

Table 4: The number of iterations of the multilevel approximate inverse preconditioner can be reduced by increasing the number of reduction levels  $n_{lev}$  for the diagonal blocks, at roughly equal memory costs.

comparison between AMES and the other solvers at fixed memory occupation may be a little penalizing for the AINV, FSAI and SPAI preconditioners as one-level approximate inverses inherently need more memory; the ARMS method is a multilevel solver, but it factorizes the diagonal blocks without any permutation.

475 In Table 6, we show the complete results of our experiments. These include number of iterations ( $Its$ ), density ratio ( $\frac{nnz(M_L+M_U)}{nnz(A)}$ ), time costs for the reordering ( $t_p$ ), factorization ( $t_f$ ) and solve phase ( $t_s$ ). We also tested the unpreconditioned GMRES for these matrices problems, and no convergence is achieved. We clearly see the good potential of the multilevel mechanism  
480 incorporated in the AMES preconditioner to reduce the number of iterations of Krylov methods, also in comparison to other multilevel solvers at low to moderate memory costs. In our examples, AMES was more robust than these solvers especially at low memory ratios.

Matrix	$n_{levAS}$	$Its$	$t_p$	$t_f$	$t_s$
cz20468	0	331	0.5	0.2	8.2
	1	228	0.4	1.3	5.6
	2	209	0.4	1.3	4.8
	3	181	0.4	1.3	4.0
ABACUS_shell_ud	0	576	0.5	1.8	35.0
	1	485	0.5	1.8	29.5
	2	414	0.5	1.4	24.4
	3	393	0.5	1.6	22.2
cz40948	0	183	1.9	0.5	23.7
	1	166	1.9	6.4	16.6
	2	157	1.9	6.1	14.8
	3	152	1.8	6.1	14.3

Table 5: At roughly equal memory costs, larger reduction levels for the Schur complement can improve the convergence rate.

#### 4.6. Effect of overlapping

485

We solved several problems from Table 1 combining the AMES method with overlapping after the first level of reordering in (6). In these runs, we set  $n_{lev} = 2$ , and we tuned the *droptol* parameter to have roughly the same memory costs in the experiments with and without overlapping. In the last two columns of

490 Table 7 we give the effect of overlapping on the change in size and in number of nonzeros for the overlapped system. The number of iteration ( $Its$ ) are almost the same after overlapping for problems *cz20468* and *cz40948*, while for problems *sme3Db*, *ABACUS\_shell\_ud* and *raefsky3* we observed a consistent reduction of the number of iterations  $Its$  by a factor between 9.5% and 23.8% and of the

495 solving time  $t_s$  by a factor between 21.4% and 29.9%. This is in agreement

(a) cz20468

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	1.26	187	0.3	0.2	4.2
ILUPACK	1.24	2500	-	0.4	40.3
ARMS	1.16	+5000	-	0.1	+6.5
SPAI	1.64	+5000	-	4.0	+8.0

(b) raefsky3

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	0.54	235	2.4	3.7	10.0
ILUPACK	0.55	1224	-	2.8	25.2
ARMS	2.38	+5000	-	2.4	+23.5
SPAI	1.83	+5000	-	5040	+243.0

(c) ABACUS\_shell\_ud

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	1.79	453	0.3	0.8	22.1
ILUPACK	1.82	1411	-	0.5	26.6
ARMS	1.88	+5000	-	0.2	+7.6
SPAI	2.41	+5000	-	11.0	+12.0

(d) sme3Db

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	0.85	407	3.5	8.4	39.3
ILUPACK	0.74	1210	-	4.1	41.4
ARMS	5.61	+5000	-	39.0	+54.9
SPAI	1.23	+5000	-	3360	+123.0

with our analysis of Section 3. In Table 8, for each problem we studied the sparsity pattern of block  $F$  and the size of blocks  $B$  and  $C$  before and after overlapping is applied at the first reordering level. The quantity  $Sp_F$  denotes the ratio between the number of nonzero elements and the size of  $F$ , that is

(e) viscoplastic2

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	3.07	78	0.9	14.3	3.9
ILUPACK	4.00	2500	-	1.6	70.0
ARMS	3.02	+5000	-	0.9	+10.9
SPAI	3.37	+5000	-	244.0	+24.0

(f) cz40948

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	1.41	170	0.7	0.4	7.4
ILUPACK	1.48	1627	-	1.0	51.1
ARMS	1.70	+5000	-	0.9	+21.8
SPAI	1.64	+5000	-	8.5	+17.2

(g) rma10

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	2.33	164	3.9	13.1	34.5
ILUPACK	2.27	1242	-	8.6	82.9
ARMS	14.30	+5000	-	203.9	+111.3
SPAI	4.84	+5000	-	11280	+180

(h) finan512

Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	0.59	9	0.8	0.5	0.8
ILUPACK	0.62	11	-	0.7	0.1
ARMS	0.58	36	-	0.4	0.5
SPAI	0.61	7	-	4.2	0.2



(i) helm2d03					
Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	0.88	6	6.1	4.3	4.6
ILUPACK	0.91	7	-	3.7	0.4
ARMS	0.93	12	-	1.4	1.5
SPAI	0.87	15	-	100.7	2.7
(j) parabolic_fem					
Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
AMES	0.75	4	4.7	5.7	1.3
ILUPACK	0.68	10	-	5.3	0.5
ARMS	0.76	12	-	2.0	2.0
SPAI	0.77	4	-	175.3	0.8

Table 6: Performance comparison of the multilevel approximate inverse preconditioner against other iterative solvers, both one-level and multilevel.

the sparsity degree  $\frac{nnz(F)}{size(F)}$ . As we can see, the *cz20468* and *cz40948* problems have the smallest relative size of the separator  $C$  and also the smallest value of  $Sp_F$ ; this means that less information is added to the subdomains. Following the analysis reported in Section 3, the overlapping technique is less likely to help on these two matrices, and this is also confirmed by the numerical results. Differently, problems *sme3Db* and *raefsky3* show larger values of  $size_C$  and  $Sp_F$  and in fact overlapping has a better effect on convergence for these two problems. In our experiments we found that a small number of independent clusters  $p$  is recommended to use when overlapping.

Matrix	Method	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_s$	$\frac{n(A_{overlapped})}{n(A)}$	$\frac{nnz(A_{overlapped})}{nnz(A)}$
cz20468	overlapping	1.33	147	4.3	1.005	1.004
	without overlapping	1.32	149	4.5	-	-
raefsky3	overlapping	0.56	218	14.3	1.134	1.135
	without overlapping	0.56	286	20.3	-	-
ABACUS_shell_ud	overlapping	3.06	238	13.6	1.020	1.019
	without overlapping	3.03	263	17.4	-	-
sme3Db	overlapping	0.91	389	49.1	1.588	1.639
	without overlapping	0.91	495	62.5	-	-
cz40948	overlapping	1.42	175	12.0	1.006	1.004
	without overlapping	1.40	172	17.5	-	-

Table 7: Experiments on the effect of block overlapping on the performance of the multilevel sparse approximate inverse.

#### 4.7. Utilizing direct solvers in the AMES framework

The results of previous sections indicate that the multilevel mechanism can be effective to reduce the memory burden but, at least in our implementation, tends to increase the cost per iteration. As an attempt of a possible remedy, we

Matrix problem	Method	$size_B$	$size_C$	$Sp_F$
cz20468	original	20405	63	$1.1e - 4$
	after overlapping	20450	116	$4.3e - 5$
raefsky3	original	19776	1424	$1.1e - 4$
	after overlapping	21184	2864	$5.1e - 4$
ABACUS_shell_ud	original	23184	228	$1.3e - 4$
	after overlapping	23412	458	$6.1e - 5$
sme3Db	original	19956	9111	$9.2e - 4$
	after overlapping	25932	20214	$3.4e - 4$
cz40948	original	40825	123	$5.2e - 5$
	after overlapping	40925	250	$2.4e - 5$

Table 8: Effect of overlapping on matrix blocks size and sparsity.

performed some runs setting the dropping threshold parameter *droptol* equal  
515 to zero, and using a sparse direct solver, namely the routine **MA38** from the  
HSL Mathematical Software Library [22], as a local solver. No approximation  
is introduced and the Schur complements are exact. Therefore in each problem  
we can obtain convergence in one or two iterations, and the solving phase is  
much cheaper. This can be observed in Table 9 on selected matrix problems.  
520 Comparing against the results with inexact inversion, we see that using a direct  
solver as local component can save computational time at only moderate extra  
storage cost.

## 5. Conclusions

In this paper a recursive multilevel implementation of factorized sparse  
525 approximate inverse preconditioners for Krylov subspace methods was discussed.  
We used recursive combinatorial techniques and overlapping strategies as an  
attempt to remedy two typical drawbacks of explicit preconditioning, that are  
lack of robustness and high construction cost. The numerical experiments show

Matrix	$\frac{nnz(M_L+M_U)}{nnz(A)}$	$Its$	$t_p$	$t_f$	$t_s$
<b>cz20468</b>	1.28	2	0.7	0.4	1.5
<b>raefsky3</b>	2.74	1	3.4	11.1	1.3
<b>cz40948</b>	1.87	2	1.2	0.3	0.2
<b>rma10</b>	3.01	1	5.2	11.6	0.8

Table 9: Using the AMES factorization as a direct solver.

that these strategies can improve the performance of conventional approximate  
530 inverse methods, yielding iterative solutions that can compete favourably  
against other popular solvers in use today. Parallelism can be exploited at  
various levels in our method, alongside other code optimization. Fine-grained  
blocking, filtering, postfiltering, adaptive pattern selection strategies have been  
shown to be promising approaches in other contexts [26, 18, 25, 13, 12, 11],  
535 and these can be considered also in our setting. In a distributed memory  
implementation, it will be natural to split the oct-tree by assigning the local  
problems to different processors. An efficient use of recursive combinatorial  
algorithms may reduce considerably the size of the Schur complements, hence  
the amount of inter-node communications. Memory demands, an important  
540 bottleneck of modern algorithms, are also limited, but this does not penalize  
much the overall numerical efficiency of the solver, as illustrated by the  
experiments of Tables 6 and 9. Overlapping does not destroy the sparsity  
structure of the matrix and can reduce further the interconnections between  
subdomains and separator set. Hence it is worthwhile considering it in a parallel  
545 setting as well. However, the parallel implementation of a fully distributed Schur  
complement formulation may not be trivial and will be considered in a separate  
study.

## 6. Acknowledgements

550 The work of the first and of the third authors is funded by the Ubbo  
Emmius scholarship of the University of Groningen. The work of the last author  
is supported by NSFC (61170309), the Fundamental Research Funds for the  
Central Universities (ZYGX2013Z005). We are grateful to Miroslav Tůma for  
motivating discussion and for supplying the AINV package for our numerical  
555 experiments. Finally, we would like to thank the anonymous reviewers for their  
valuable and thorough comments that we believe helped improve significantly  
the presentation and the quality of the paper.

- [1] M. Benzi, Preconditioning techniques for large linear systems: A survey, *J. Comp. Phys.* 182 (2002) 418–477.
- 560 [2] M. Benzi, J. Marín, M. M. Tůma, A two-level parallel preconditioner based  
on sparse approximate inverses, in: D. Kincaid, A. Elster (eds.), *Iterative  
Methods in Scientific Computation IV*, IMACS Series, 1999.
- [3] M. Benzi, C. Meyer, M. Tůma, A sparse approximate inverse preconditioner  
for the conjugate gradient method., *SIAM J. Scientific Computing* 17 (1996)  
565 1135–1149.
- [4] M. Benzi, M. Tůma, A sparse approximate inverse preconditioner for  
nonsymmetric linear systems., *SIAM J. Scientific Computing* 19 (1998)  
968–994.
- [5] M. Bollhoefer, Y. Saad, O. Schenk, ILUPACK - preconditioning software  
570 package, available online at <http://ilupack.tu-bs.de/>. Release V2.3,  
December 2010. (2010).
- [6] M. Bollhöfer, A robust and efficient ILU that incorporates the growth of  
the inverse triangular factors, *SIAM J. Sci. Comput.* 25 (1) (2003) 86–103.  
URL <http://dx.doi.org/10.1137/S1064827502403411>
- 575 [7] M. Bollhöfer, Y. Saad, Multilevel preconditioners constructed from inverse-  
based ILUs, *SIAM J. Scientific Computing* 27 (5) (2006) 1627–1650.

- [8] B. Carpentieri, Fast iterative solution methods in electromagnetic scattering, *Progress in Electromagnetic Research* 79 (2007) 151–178.
- [9] B. Carpentieri, M. Bollhöfer, Symmetric inverse-based multilevel ILU preconditioning for solving dense complex non-hermitian systems in electromagnetics, *Progress In Electromagnetics Research* 128 (2012) 55–74.
- [10] B. Carpentieri, I. Duff, L. Giraud, G. Sylvand, Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations, *SIAM J. Scientific Computing* 27 (3) (2005) 774–792.
- [11] B. Carpentieri, J. Liao, M. Sosonkina, VBARMS: A variable block algebraic recursive multilevel solver for sparse linear systems, *Journal of Computational and Applied Mathematics* 259 (A) (2014) 164–173.
- [12] E. Chow, A priori sparsity patterns for parallel sparse approximate inverse preconditioners., *SIAM J. Scientific Computing* 21 (5) (2000) 1804–1822.
- [13] E. Chow, Parallel implementation and practical use of sparse approximate inverses with a priori sparsity patterns., *Int. J. High Perf. Comput. Appl.* 15 (2001) 56–74.
- [14] E. Chow, Y. Saad., Approximate inverse preconditioners via sparse-sparse iterations., *SIAM J. Scientific Computing* 19 (3) (1998) 995–1023.
- [15] T. Davis, Sparse matrix collection, available at the URL: <http://www.cise.ufl.edu/research/sparse/matrices> (1994).
- [16] I. S. Duff, J. Koster, The design and use of algorithms for permuting large entries to the diagonal of sparse matrices, *SIAM J. Matrix Analysis and Applications* 20 (4) (1999) 889–901.
- [17] I. S. Duff, G. A. Meurant, The effect of ordering on preconditioned conjugate gradient., *BIT* 29 (1989) 635–657.

- [18] M. Ferronato, C. Janna, G. Pini, A generalized Block FSAI preconditioner  
605 for nonsymmetric linear systems, *J. Comput. Appl. Math.* 256 (2014) 230–  
241.
- [19] A. George, J. W. Liu, *Computer Solution of Large Sparse Positive Definite  
Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.
- [20] L. Grigori, F. Nataf, L. Qu, Overlapping for preconditioners based on  
610 incomplete factorizations and nested arrow form, *Numerical Linear Algebra  
with Applications* 22 (1) (2015) 48–75.  
URL <http://dx.doi.org/10.1002/nla.1937>
- [21] M. Grote, T. Huckle, Parallel preconditionings with sparse approximate  
inverses., *SIAM J. Scientific Computing* 18 (1997) 838–853.
- [22] HSL, A collection of Fortran codes for large scale scientific computation,  
615 <http://www.numerical.rl.ac.uk/hsl> (2000).
- [23] T. Huckle, A. Kallischko, A. Roy, M. Sedlacek, T. Weinzierl, An efficient  
parallel implementation of the MSPAI preconditioner, *Parallel Computing*  
36 (5-6) (2010) 273–284.
- [24] T. Huckle, M. Sedlacek, Smoothing and regularization with modified sparse  
620 approximate inverses, *Journal of Electrical and Computer Engineering*  
2010 (2010) 1–16.  
URL [http://downloads.hindawi.com/journals/jece/2010/930218.  
pdf](http://downloads.hindawi.com/journals/jece/2010/930218.pdf)
- [25] C. Janna, M. Ferronato, Adaptive pattern research for block FSAI  
625 preconditioning, *SIAM J. Sci. Comput.* 33 (6) (2011) 3357–3380.
- [26] C. Janna, M. Ferronato, G. Gambolati, A block FSAI-ILU parallel  
preconditioner for symmetric positive definite linear systems, *SIAM J. Sci.  
Comput.* 32 (5) (2010) 2468–2484.

- 630 [27] C. Janna, M. Ferronato, G. Gambolati, Enhanced block FSAI preconditioning using domain decomposition techniques, *SIAM J. Sci. Comput.* 35 (5) (2013) S229–S249.
- [28] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM J. Sci. Comput.* 20 (1999) 359–392.
- 635 [29] L. Y. Kolotilina, A. Y. Yeremin, Factorized sparse approximate inverse preconditionings. I: Theory, *SIAM J. Matrix Analysis and Applications* 14 (1993) 45–58.
- [30] L. Y. Kolotilina, A. Y. Yeremin, Factorized sparse approximate inverse preconditionings. II: Solution of 3D FE systems on massively parallel computers., *Int J. High Speed Computing* 7 (1995) 191–215.
- 640 [31] L. Y. Kolotilina, A. Y. Yeremin, A. Nikishin, Factorized sparse approximate inverse preconditionings. IV: Simple approaches to rising efficiency., *Numerical Linear Algebra with Applications* 6 (1999) 515–531.
- [32] L. Y. Kolotilina, A. Y. Yeremin, A. Nikishin, Factorized sparse approximate inverse preconditionings. III: Iterative construction of preconditioners., *Journal of Mathematical Sciences* 101 (2000) 3237–3254, originally published in Russian in *Zap. Nauchn. Semin. POMI*, 248:17–48, 1998.
- 645 [33] R. Li, Y. Saad, GPU-accelerated preconditioned iterative linear solvers, *The Journal of Supercomputing* 63 (2) (2013) 443–466.
- [34] M. Manguoglu, A domain-decomposing parallel sparse linear system solver, *Journal of Computational and Applied Mathematics* 236 (3) (2011) 319 – 325.
- 650 [35] J. A. Meijerink, H. A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix., *Mathematics of Computation* 31 (1977) 148–162.
- 655



- [36] M. M. monga Made, R. Beauwens, G. Warzee, Preconditioning of discrete Helmholtz operators perturbed by a diagonal complex matrix., Communications in Numerical Methods in Engineering 11 (2000) 801–817.
- [37] X.-M. Pan, X.-Q. Sheng, Improved algebraic preconditioning for MoM  
660 solutions of large-scale electromagnetic problems, Antennas and Wireless Propagation Letters, IEEE 13 (2014) 106–109.
- [38] X.-M. Pan, X.-Q. Sheng, Sparse approximate inverse preconditioner for multiscale dynamic electromagnetic problems, Radio Science 49 (2014) 1041–1051.  
665 URL <http://dx.doi.org/10.1002/2014RS005387>
- [39] A. Quarteroni, A. Valli, Domain decomposition methods for partial differential equations., Clarendon Press Oxford, 1999.
- [40] Y. Saad, Iterative Methods for Sparse Linear Systems., SIAM Publications, second edition, 2003.
- [41] Y. Saad, Multilevel ILU with reorderings for diagonal dominance, SIAM J. Scientific Computing 27 (2005) 1032–105.  
670
- [42] Y. Saad, M. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems., SIAM J. Scientific and Statistical Computing 7 (1986) 856–869.
- [43] Y. Saad, A. Soulaïmani, R. Touihri, Variations on algebraic recursive multilevel solvers (ARMS) for the solution of CFD problems, Applied Numerical Mathematics 51 (2004) 305–327.  
675
- [44] Y. Saad, B. Suchomel, ARMS: An algebraic recursive multilevel solver for general sparse linear systems, Numerical Linear Algebra with Applications 9 (5) (2002) 359–378.  
680
- [45] A. Yeremin, A. Nikishin, Factorized-sparse-approximate-inverse preconditionings of linear systems with unsymmetric matrices, Journal of Mathematical Sciences 121 (2004) 2448–2457.