SIAM J. Matrix Anal. Appl.
Vol. 39, No. 1, pp. 123–147

A ROBUST MULTILEVEL APPROXIMATE INVERSE PRECONDITIONER FOR SYMMETRIC POSITIVE DEFINITE MATRICES*

ANDREA FRANCESCHINI†, VICTOR ANTONIO PALUDETTO MAGRI†, MASSIMILIANO FERRONATO‡, AND CARLO JANNA§

Abstract. The use of factorized sparse approximate inverse (FSAI) preconditioners in a standard multilevel framework for symmetric positive definite (SPD) matrices may pose a number of issues as to the definiteness of the Schur complement at each level. The present work introduces a robust multilevel approach for SPD problems based on FSAI preconditioning, which eliminates the chance of algorithmic breakdowns independently of the preconditioner sparsity. The multilevel FSAI algorithm is further enhanced by introducing descending and ascending low-rank corrections, thus giving rise to the multilevel FSAI with low-rank corrections (MFLR) preconditioner. The proposed algorithm is investigated in a number of test problems. The numerical results show that the MFLR preconditioner is a robust approach that can significantly accelerate the solver convergence rate preserving a good degree of parallelism. The possibly large set-up cost, mainly due to the computation of the eigenpairs needed by low-rank corrections, makes its use attractive in applications where the preconditioner can be recycled along a number of linear solves.

Key words. preconditioning, approximate inverses, parallel computing, iterative methods

AMS subject classifications. 65F08, 65F10, 65F50, 65Y05

DOI. 10.1137/16M1109503

1. Introduction.

The solution of a large sparse linear system of equations in the form

$$A\mathbf{x} = \mathbf{b} \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$, $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$, and A is symmetric positive definite (SPD), typically requires the use of preconditioned iterative methods based on Krylov subspaces. As is well known, the key for accelerating the solver convergence is the definition of the preconditioner, i.e., the operator that approximates the action of A^{-1}.

There are several different approaches for building an efficient preconditioner, either physics-based or purely algebraic. Among the algebraic algorithms, the most popular categories are incomplete factorizations, multigrid methods, and sparse approximate inverses [2, 11], though different preconditioners can be composed to form new ones, e.g., blending domain decomposition approaches with incomplete factorizations or approximate inverses, or using nested Krylov methods [7, 18, 19, 24]. Combinations of physics-based and purely algebraic approaches are also denoted as "gray-box" solvers [2, 5]. For example, some physics-based pattern selection strategies

*Received by the editors December 27, 2016; accepted for publication (in revised form) by D. Orban August 21, 2017; published electronically January 24, 2018.
http://www.siam.org/journals/simax/39-1/M110950.html
Funding: The work of the authors was supported by the ISCRA project "SCAIP: Scalable approximate inverse preconditioners."
†Department ICEA, University of Padova, 35121 Padova, Italy (franc90@dmsa.unipd.it, victor.magri@dicea.unipd.it).
‡M^{3}E s.r.l., 35129 Padova, Italy, and Department ICEA, University of Padova, 35121 Padova, Italy (massimiliano.ferronato@unipd.it).
§Corresponding author. M^{3}E s.r.l., 35129 Padova, Italy, and Department ICEA, University of Padova, 35121 Padova, Italy (carlo.janna@unipd.it).

can be used for algebraic sparse approximate inverse preconditioners [5] and block variants of multilevel incomplete factorizations [17, 6].

Compared to incomplete factorizations, sparse approximate inverses are generally more robust and more appropriate for parallel computational architectures. In particular, the factorized sparse approximate inverse (FSAI) [21] is an algebraic preconditioner for SPD problems that proves effective in a wide range of applications, especially in its dynamically adaptive variants [16, 20]. This algorithm provides a factorized approximation of A^{-1}:

$$G^T G \simeq A^{-1}, \tag{1.2}$$

where G is a lower triangular matrix explicitly computed in the set-up phase so as to approach the inverse of the lower Cholesky factor of A in the sense of the Frobenius norm. One of the most attractive features of FSAI for modern computers is its intrinsic high degree of parallelism. In fact, each row of G can be fully formed independently of the others, with the parallelization trivially accomplished by evenly subdividing the rows among threads and/or processes. The degree of parallelism is even redundant, as the number of rows is much larger than the available number of computing cores.

The central idea of this work is to introduce some sequentiality in the FSAI computation, in order to use the information extracted from the earlier set-up stages for the remaining rows. This concept, which is the basis for incomplete factorizations, has been already introduced in the context of approximate inverses in [8, 25], and more recently in [12], where both a block tridiagonal and a domain decomposition approach have been used to improve the FSAI performance. In the present paper, we develop a more general multilevel framework that describes the former approaches just by selecting a proper unknown ordering.

One of the difficulties arising in the multilevel generalization of the FSAI preconditioner is related to the accuracy in the computation of the Schur complement at each level. If the earlier levels are not well approximated, the resulting Schur complement can be inaccurate, with a consequent degradation of the solver performance. This issue has been recently addressed in the context of multilevel incomplete factorizations with the aid of low-rank corrections [29]. Low-rank compression algorithms are gaining increasing attention especially in direct linear solution methods, e.g., [1, 14, 28, 30, 31, 32], with the basic idea of taking advantage of data sparsity instead of structural sparsity. During the factorization process, the off-diagonal blocks, which in discretized-PDE problems are usually characterized by low-rank properties, are decomposed by SVD and compressed by neglecting those components corresponding to the smallest singular values. There are several schemes to carry out the compression on the entire matrix, ranging from $\mathcal{H}$-matrices Hermitian and skew-Hermitian splitting methods to block low rank tree structured compression. Typically a drop tolerance is set below which the singular values are dropped. If this tolerance is sufficiently small, the matrix factorization is cheaper but can still be directly applied to a right-hand side to get the system solution. With large drop tolerances, the low-rank approach gives an approximation of the exact factorization and can be used as a preconditioner, e.g., [1]. To our knowledge, the first attempt to directly use low-rank representations in preconditioning is discussed in [23] in the context of a divide and conquer strategy.

Within the FSAI multilevel framework presented in this work, low-rank corrections are introduced for both enhancing the preconditioner quality at the earlier levels (descending low-rank, DLR) and improving the accuracy in the Schur complement computation (ascending low-rank, ALR). The paper is organized as follows.

Algorithm 2.1. Multilevel Factorization Set-up.

1. **Function** ML_SetUp(n_l,A)
2. Set $A_0 = A$;
3. **for all** $l = 0, \ldots, n_l - 2$ **do**
4. Partition A_l as $\begin{bmatrix} K & B \\ B^T & C \end{bmatrix}$;
5. Compute $\widetilde{L}$ such that $\widetilde{L}\widetilde{L}^T \simeq K$;
6. Compute $\widetilde{H}$ such that $\widetilde{H} \simeq L_K^{-1}B$;
7. Compute $\widetilde{S}$ such that $\widetilde{S} \simeq C - B^T K^{-1} B$;
8. Form $M_l = \begin{bmatrix} \widetilde{L} & 0 \\ \widetilde{H}^T & I \end{bmatrix}$;
9. Set $A_{l+1} = \widetilde{S}$;
10. **end for**
11. Compute M_{n_l-1} such that $M_{n_l-1} M_{n_l-1}^T \simeq A_{n_l-1}$;
12. Set $M = \{M_0, M_1, \ldots, M_{n_l-1}\}$;

The Multilevel FSAI (MF) preconditioner is first derived proving its robustness. It is then demonstrated that the MF quality can be improved if the approximated Schur complement computed at any level is corrected in order to approximate the preconditioner instead of the original matrix Schur complement. Low-rank corrections are finally introduced to further improve the MF preconditioner. The properties of this approach are numerically investigated, and finally a few considerations close the work.

2. Multilevel FSAI preconditioning. Consider a standard multilevel approach applied to the SPD matrix A for a total number n_l of levels. The matrix A_l obtained at any level $l \in [0, n_l - 1]$ is the Schur complement of the previous level. Following for instance [17], we can partition A_l in four blocks and perform the factorization:

$$A_l = \begin{bmatrix} K & B \\ B^T & C \end{bmatrix} = \begin{bmatrix} L_K & 0 \\ B^T L_K^{-T} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} L_K^T & L_K^{-1}B \\ 0 & I \end{bmatrix}, \tag{2.1}$$

where $L_K L_K^T$ is the exact Cholesky decomposition of K, $K \in \mathbb{R}^{n_1 \times n_1}$, and $S = C - B^T K^{-1} B$, $S \in \mathbb{R}^{n_2 \times n_2}$, is the Schur complement of A_l with respect to the partition (n_1, n_2), i.e., $A_{l+1} \equiv S$. The partitioning of each level can follow some physics-based criterion, if any. As our goal is to get a preconditioner, we can replace each block on the right-hand side of (2.1) with an approximation:

$$\widetilde{L} \simeq L_K, \qquad \widetilde{H} \simeq L_K^{-1}B, \qquad \widetilde{S} \simeq S \tag{2.2}$$

The recursive computation of (2.1) with the approximations (2.2) provides the general framework for a multilevel preconditioner M of A, made by the list of factors M_l, $l \in [0, n_l]$ (Algorithm 2.1). The preconditioner application stage, i.e., the computation of $\mathbf{w} = M^{-1}\mathbf{v}$ for some known vector $\mathbf{v}$, is provided in Algorithm 2.2.

Because of the recursion in the computation of M, we can restrict to the two-level case with no loss of generality. Hence, A_l in (2.1) coincides with A and $n_1 + n_2 = n$. A popular choice is to use an incomplete factorization as the main kernel for the approximations in (2.2), i.e., setting $\widetilde{L}$ equal to the incomplete Cholesky (IC) factor of K as in [26, 22, 17, 4]. In the same framework, the use of FSAI as the main kernel

Algorithm 2.2. Multilevel Factorization Application.

Function $\mathbf{w} = \text{ML\_APPLY}(n_l, M, \mathbf{v})$
Set $lev_{start} = 1$;
for all $l = 0, \ldots, n_l - 2$ **do**
  Solve $M_l \mathbf{z} = \mathbf{v}$;
  Set $lev_{end} = lev_{start} + n_1 - 1$;
  Form $\mathbf{w}(lev_{start} : lev_{end}) = \mathbf{v}(1 : n_1)$;
  Set $\mathbf{v} = \mathbf{v}(n_1 + 1 : n)$;
  Update $lev_{start} = lev_{end} + 1$;
end for
Solve $(M_{n_l-1} M_{n_l-1}^T)\mathbf{z} = \mathbf{v}$;
Form $\mathbf{w}(lev_{start} : n) = \mathbf{z}$;
Set $lev_{end} = lev_{start} - 1$;
for all $l = n_l - 2, \ldots, 0$ **do**
  Set $lev_{start} = lev_{end} - n_1 + 1$;
  Retrieve $\mathbf{w}_l = \mathbf{w}(lev_{start} : n)$;
  Solve $M_l^T \mathbf{z} = \mathbf{w}_l$;
  Form $\mathbf{w}(lev_{start} : n) = \mathbf{w}_l$;
  Update $lev_{end} = lev_{start} - 1$;
end for

is straightforward. After computing G such that $G^T G \simeq K$, the approximations in (2.2) become

$$\widetilde{L} \simeq G^{-1}, \qquad \widetilde{H} \simeq GB, \qquad \widetilde{S} \simeq C - \widetilde{H}^T \widetilde{H}. \tag{2.3}$$

In contrast to what typically happens using an incomplete factorization as the main kernel, no dropping is necessary to compute efficiently $\widetilde{H}$ and $\widetilde{S}$, because G is usually very sparse. In fact, recall that the sparsity of G is controlled by the user, while the one of $\widetilde{L}^{-1}$ is not.

Unfortunately, this straightforward implementation of the MF preconditioner is very prone to breakdowns. In fact, the Schur complement approximation $\widetilde{S} = C - \widetilde{H}^T \widetilde{H}$ is computed as the difference between two SPD matrices and can be indefinite. Such an experience is quite common even in relatively well-conditioned problems. The reason for this resides in the poor approximation of the leftmost eigenvalues usually obtained by FSAI. Figure 1 compares the eigenspectra of the `bcsstk16` matrix (structural problem, $n = 4,884$ with 290,378 nonzero entries) from the University of Florida sparse matrix collection [9] with its approximations by IC and FSAI, $\widetilde{L}\widetilde{L}^T$ and $(G^T G)^{-1}$, respectively, where $\widetilde{L}$ and G are computed with the same number of nonzeros. Both IC and FSAI are not able to capture the smallest eigenvalues of a matrix, though IC is generally better. The smallest eigenvalues of K are the largest of K^{-1}, and therefore control the most significant entries of $B^T K^{-1} B$. As a consequence, $\widetilde{H}^T \widetilde{H}$ computed as in (2.3) often fails in approximating accurately the largest entries of $B^T K^{-1} B$, leading to the appearance of negative eigenvalues in $\widetilde{S}$ and causing the breakdown of the procedure. This occurrence can happen using IC as a kernel in a multilevel preconditioning framework as well. To address this issue some stabilization techniques, e.g., based on diagonal shifts [17, 27], have been successfully introduced. Unfortunately, these strategies do not provide satisfactory results when used for the computation of $\widetilde{S}$ with the aid of FSAI as a kernel.

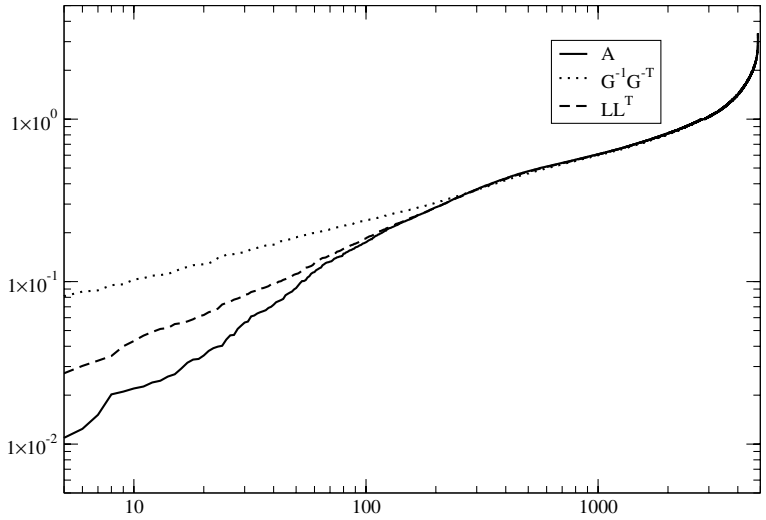

FIG. 1. *Comparison between the eigenspectra of A, $\widetilde{L}\widetilde{L}^T$, and $(G^TG)^{-1}$ for the* `bcsstk16` *matrix from the University of Florida sparse matrix collection.*

The robustness of the MF preconditioner can be ensured by computing $\widetilde{S}$ so as to be SPD independently of G. To this aim, we can use the following result.

THEOREM 2.1. *Let $A \in \mathbb{R}^{n\times n}$ be an SPD (2×2)-block matrix,*

$$A = \begin{bmatrix} K & B \\ B^T & C \end{bmatrix}, \tag{2.4}$$

with $K \in \mathbb{R}^{n_1\times n_1}$, $B \in \mathbb{R}^{n_1\times n_2}$, and $C \in \mathbb{R}^{n_2\times n_2}$, and let $V \in \mathbb{R}^{n\times n_2}$ and $D \in \mathbb{R}^{n_2\times n}$ be the 2-block rectangular matrices

$$V = \begin{bmatrix} F^T \\ I \end{bmatrix}, \qquad D = \begin{bmatrix} 0 & Z \end{bmatrix}, \tag{2.5}$$

with $F \in \mathbb{R}^{n_2\times n_1}$ and $Z \in \mathbb{R}^{n_2\times n_2}$, such that the Frobenius norm $\|D - V^TL\|_F$ is minimum for any Z, L being the lower Cholesky factor of A. Then, $S = V^TAV$ is the Schur complement of A with respect to the partition (n_1, n_2).

Proof. Recalling (2.1), the lower Cholesky factor of A reads

$$L = \begin{bmatrix} L_K & 0 \\ B^TL_K^{-T} & L_S \end{bmatrix}, \tag{2.6}$$

where L_S is the lower Cholesky factor of the Schur complement of A, i.e., $L_SL_S^T = C - B^TK^{-1}B$. The matrix $(D - V^TL)$ is therefore

$$D - V^TL = \begin{bmatrix} FL_K + B^TL_K^{-T}, & Z - L_S \end{bmatrix}, \tag{2.7}$$

whose Frobenius norm is minimum for any Z if $FL_K + B^TL_K^{-T} = 0$; i.e.,

$$F = -B^TK^{-1}. \tag{2.8}$$

The matrix $S = V^TAV$ reads

$$S = FKF^T + B^TF^T + FB + C. \tag{2.9}$$

Introducing (2.8) into (2.9) provides $V^TAV = C - B^TK^{-1}B$. ∎

Remark 2.2. The matrix F of Theorem 2.1 is generally dense. If (2.8) is enforced only for the entries located in a prescribed set of positions $\widetilde{\mathcal{S}} \subset \mathcal{S} = \{(i,j) : 1 \leq i \leq n_1, 1 \leq j \leq n_2\}$, the sparsity of F can be retained at a workable level. This definition for F coincides with the block FSAI preconditioner introduced in [18] and [16], where the nonzero pattern $\widetilde{\mathcal{S}}$ is defined either statically or dynamically during the computation of F. Using a sparse F in (2.9) produces an approximation $\widetilde{S}$ of the exact Schur complement S of A.

COROLLARY 2.3. *The Schur complement approximation* $\widetilde{S}$ *computed with* (2.9) *and a sparse block FSAI* F *is SPD.*

Proof. The expression of $\widetilde{S}$ can be easily rearranged by adding and subtracting $B^T K^{-1} B$:

$$\widetilde{S} = C - B^T K^{-1} B + (F + B^T K^{-1}) K (K^{-T} B + F^T) = S + W^T K W. \tag{2.10}$$

The result immediately follows by noting that $W^T K W$ is SPD. ∎

Based on these results, the MF preconditioner is built as follows. The zero-level preconditioner M_0^{-1} is made by two factors:

$$M_0^{-1} = P_b P_a. \tag{2.11}$$

An explicit approximation of K^{-1} is computed as $G^T G$ using an adaptive FSAI procedure [20] and introduced in P_a:

$$P_a = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix}. \tag{2.12}$$

Then, the preconditioned matrix $P_a A P_a^T$ is computed,

$$P_a A P_a^T = \begin{bmatrix} GKG^T & GB \\ B^T G^T & C \end{bmatrix}, \tag{2.13}$$

and the adaptive block FSAI [16] of $P_a A P_a^T$ is computed for the second factor:

$$P_b = \begin{bmatrix} I & 0 \\ F & I \end{bmatrix}. \tag{2.14}$$

The zero-level preconditioned matrix $M_0^{-1} A M_0^{-T}$ reads

$$M_0^{-1} A M_0^{-T} = \begin{bmatrix} I & 0 \\ F & I \end{bmatrix} \begin{bmatrix} GKG^T & GB \\ B^T G^T & C \end{bmatrix} \begin{bmatrix} I & F^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} GKG^T & -R_F^T \\ -R_F & \widetilde{S} \end{bmatrix}, \tag{2.15}$$

where $R_F = -F(GKG^T) - B^T G^T$ is the residual on F i.e., R_F approaches the null matrix as the accuracy in the computation on F increases. Finally, the (2,2) block of $M_0^{-1} A M_0^{-T}$ is the approximation of the first-level Schur complement

$$\widetilde{S} = C + FGB + B^T G^T F^T + FGKG^T F^T \tag{2.16}$$

that becomes the new matrix for the next level. As $\widetilde{S}$ in (2.16) is SPD for any F and G (see Corollary 2.3), no breakdown is possible. The operations required for building the robust MF preconditioner are provided in Algorithm 2.3.

Algorithm 2.3. FSAI-based Multilevel Preconditioner Set-up.

1. **Function** MF_SetUp(n_l,A)
2. Set $A_0 = A$;
3. **for all** $l = 0, \ldots, n_l - 2$ **do**
4. Partition A_l as $\begin{bmatrix} K & B \\ B^T & C \end{bmatrix}$;
5. Compute the adaptive FSAI approximation G of K such that $G^T G \simeq K^{-1}$;
6. Set $P_a = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix}$;
7. Compute $P_b = \begin{bmatrix} I & 0 \\ F & I \end{bmatrix}$ the adaptive block FSAI approximation of $P_a A_l P_a^T$;
8. Compute $\widetilde{S} = C + FGB + B^T G^T F^T + FGKG^T F^T$;
9. Form $M_l^{-1} = P_b P_a$;
10. Set $A_{l+1} = \widetilde{S}$;
11. **end for**
12. Compute $M_{n_l-1}^{-1}$ as the adaptive FSAI approximation of A_{n_l-1};
13. Set $M^{-1} = \left\{ M_0^{-1}, M_1^{-1}, \ldots, M_{n_l-1}^{-1} \right\}$;

2.1. Theoretical properties. In this section, we will obtain some theoretical bounds on the eigenspectrum of the preconditioned matrix according to the different approximations introduced in the MF computation. At every level of the MF preconditioner set-up, the partial factorization of the approximated Schur complement (2.16) is computed. This operation introduces additional approximations level after level, potentially yielding to a Schur complement quite different from the exact one. Consider the zero-level preconditioner M_0^{-1} of (2.11):

$$M_0^{-1} = \begin{bmatrix} G & 0 \\ FG & I \end{bmatrix}. \tag{2.17}$$

The natural choice for the next level preconditioner is

$$\widetilde{M}_1^{-1} = \begin{bmatrix} I & 0 \\ 0 & L_{\widetilde{S}}^{-1} \end{bmatrix}, \tag{2.18}$$

where $L_{\widetilde{S}} L_{\widetilde{S}}^T = \widetilde{S}$. However, $\widetilde{S}$ is an approximated Schur complement. Should S be available, one could use

$$M_1^{-1} = \begin{bmatrix} I & 0 \\ 0 & L_S^{-1} \end{bmatrix} \tag{2.19}$$

with $L_S L_S^T = S$. Using either $\widetilde{M}_1^{-1}$ or M_1^{-1} as the next level preconditioner of A leads to a different performance. The two propositions that follow provide a theoretical upper bound for the eigenvalues of the preconditioned matrices $\widetilde{M}^{-1} A \widetilde{M}^{-T}$ and $M^{-1} A M^{-T}$, in order to allow for an a priori assessment of the preconditioner quality.

Proposition 2.4. *The eigenvalues λ of the preconditioned matrix $\widetilde{M}^{-1} A \widetilde{M}^{-T}$, with $\widetilde{M}^{-1} = \widetilde{M}_1^{-1} M_0^{-1}$ as defined in* (2.17) *and* (2.18)*, satisfy*

$$|\lambda - 1| \leq \frac{\|E_K\| + \sqrt{\|E_K\|^2 + 4\|\widetilde{Q}^T\|\|\widetilde{Q}\|}}{2}, \tag{2.20}$$

where $\widetilde{Q} = G(KG^TF^T+B)L_{\widetilde{S}}^{-T} = -R_F^TL_{\widetilde{S}}^{-T}$ *and* $E_K = GKG^T-I$*, for any consistent matrix norm.*

Proof. The preconditioned matrix $\widetilde{M}^{-1}A\widetilde{M}^{-T}$ reads

(2.21)
$$\widetilde{M}^{-1}A\widetilde{M}^{-T} = \begin{bmatrix} GKG^T & G(KG^TF^T+B)L_{\widetilde{S}}^{-T} \\ L_{\widetilde{S}}^{-1}(FGK+B^T)G^T & I \end{bmatrix} = \begin{bmatrix} I+E_K & \widetilde{Q} \\ \widetilde{Q}^T & I \end{bmatrix}.$$

Its eigenpairs $(\lambda, \mathbf{w})$, $\mathbf{w} = [\mathbf{u}, \mathbf{v}]^T$, satisfy by definition the relationship

$$\begin{bmatrix} I+E_K & \widetilde{Q} \\ \widetilde{Q}^T & I \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix} = \lambda \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix}, \tag{2.22}$$

which is equivalent to

$$\begin{bmatrix} E_K & \widetilde{Q} \\ \widetilde{Q}^T & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix} = (\lambda-1) \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix}. \tag{2.23}$$

Taking consistent norms at both sides of the first and second set of equations we have

$$\begin{cases} \|E_K\|\|\mathbf{u}\| + \|\widetilde{Q}\|\|\mathbf{v}\| \geq |\lambda-1|\|\mathbf{u}\|, \\ \|\widetilde{Q}^T\|\|\mathbf{u}\| \geq |\lambda-1|\|\mathbf{v}\|, \end{cases} \tag{2.24}$$

which, by setting $t = \|\mathbf{v}\|/\|\mathbf{u}\|$, can be rearranged as

$$\begin{cases} |\lambda-1| \leq \|E_K\| + \|\widetilde{Q}\|t, \\ |\lambda-1| \leq \|\widetilde{Q}^T\|/t. \end{cases} \tag{2.25}$$

If $\|\mathbf{u}\| = 0$, then trivially $\mathbf{v} \in \mathrm{Ker}(\widetilde{Q})$ and $\lambda = 1$, thus satisfying the inequality (2.20). The right-hand side of the first and second inequality in (2.25) increases and decreases monotonically with t, respectively (Figure 2). The intersection point is

$$\bar{t} = \frac{-\|E_K\| + \sqrt{\|E_K\|^2 + 4\|\widetilde{Q}^T\|\|\widetilde{Q}\|}}{2\|\widetilde{Q}\|}, \tag{2.26}$$

and thus for any t we have

$$|\lambda-1| \leq \frac{\|E_K\| + \sqrt{\|E_K\|^2 + 4\|\widetilde{Q}^T\|\|\widetilde{Q}\|}}{2}. \tag{2.27}$$

∎

PROPOSITION 2.5. *The eigenvalues* λ *of the preconditioned matrix* $M^{-1}AM^{-T}$*, with* $M^{-1} = M_1^{-1}M_0^{-1}$ *as defined in* (2.11) *and* (2.19)*, satisfy*

(2.28)
$$|\lambda-1| \leq \frac{\|E_K\| + \|Q^T\|\|\left(I+E_K\right)^{-1}\|\|Q\| + \sqrt{(\|E_K\| - \|Q^T\|\|\left(I+E_K\right)^{-1}\|\|Q\|)^2 + 4\|Q^T\|\|Q\|}}{2},$$

where $Q = G(KG^TF^T+B)L_S^{-T} = -R_F^TL_S^{-T}$ *and* $E_K = GKG^T-I$*, for any consistent matrix norm.*

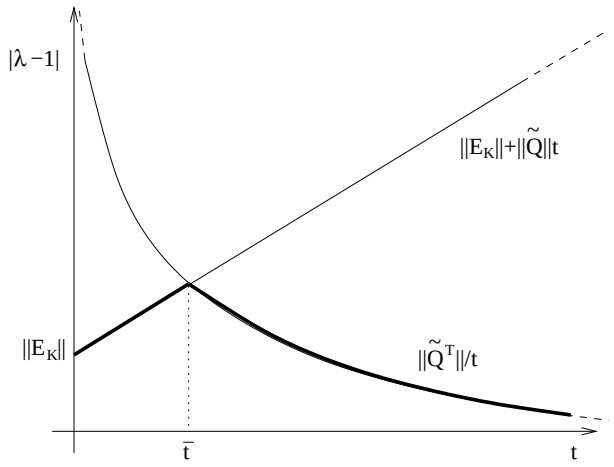

FIG. 2. *Schematic representation of the system of inequalities* (2.25).

Proof. Operating as in the proof of Proposition 2.4, it can be shown that the eigenpairs of $M^{-1}AM^{-T}$ satisfy

$$\begin{bmatrix} I+E_K & Q \\ Q^T & I+Q^T\left(I+E_K\right)^{-1}Q \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix} = \lambda \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix}, \tag{2.29}$$

which is equivalent to

$$\begin{bmatrix} E_K & Q \\ Q^T & Q^T\left(I+E_K\right)^{-1}Q \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix} = (\lambda-1) \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix}. \tag{2.30}$$

Again, taking consistent norms on both sides of the first and second set of equations, and setting $t = \|\mathbf{v}\|/\|\mathbf{u}\|$, we get

$$\begin{cases} |\lambda-1| \le \|E_K\| + \|Q\|t, \\ |\lambda-1| \le \|Q^T\|/t + \|Q^T\|\,\|\left(I+E_K\right)^{-1}\|\,\|Q\|. \end{cases} \tag{2.31}$$

If $\|\mathbf{u}\| = 0$, then trivially $\mathbf{v} \in \mathrm{Ker}(Q)$ and $\lambda = 1$, thus satisfying the inequality (2.28). Otherwise, denoting by $\bar{t}$ the intersection point between the right-hand sides of (2.31),

(2.32)
$$\bar{t} = \frac{-\|E_K\| + \|Q^T\|\,\|\left(I+E_K\right)^{-1}\|\,\|Q\| + \sqrt{\left(\|E_K\| - \|Q^T\|\,\|\left(I+E_K\right)^{-1}\|\,\|Q\|\right)^2 + 4\|Q^T\|\|Q\|}}{2\|Q\|}$$

for any t, we have

(2.33)
$$|\lambda-1| \le \frac{\|E_K\| + \|Q^T\|\,\|\left(I+E_K\right)^{-1}\|\,\|Q\| + \sqrt{\left(\|E_K\| - \|Q^T\|\,\|\left(I+E_K\right)^{-1}\|\,\|Q\|\right)^2 + 4\|Q^T\|\|Q\|}}{2}.$$

□

Now, to understand which preconditioner, either $\widetilde{M}^{-1}$ or M^{-1}, is expected to ensure a faster convergence, we compare the upper bounds provided in Propositions 2.4 and 2.5. These bounds depend on the norms of E_K and Q or $\widetilde{Q}$, which in turn

are controlled by the accuracy in the computation of G and F, respectively. As G^TG approaches K^{-1}, $\|E_K\| \to 0$, and in the limit the bounds (2.20) and (2.28) read

$$|\lambda - 1| \leq \sqrt{\|\widetilde{Q}\|\|\widetilde{Q}^T\|} \tag{2.34}$$

and

$$|\lambda - 1| \leq \frac{1}{2}\left[\|Q\|\|Q^T\| + \sqrt{\|Q\|\|Q^T\|\left(4 + \|Q\|\|Q^T\|\right)}\right], \tag{2.35}$$

respectively. Similarly, as F approaches $-B^TK^{-1}G^{-1}$, $\|Q\|, \|\widetilde{Q}\| \to 0$ and the bounds (2.20) and (2.28) trivially provide

$$|\lambda - 1| \leq \|E_K\|. \tag{2.36}$$

The previous relationships hold true for any consistent matrix norm. However, to compare the bounds in an easier way, we restrict our attention to the matrix norm induced by the 2-norm of vectors. In this case we have

(2.37)
$$\|E_K\|_2 = \lambda_1(E_K) = \epsilon_1, \quad \|Q\|_2 = \sqrt{\lambda_1(Q^TQ)} = \eta_1, \quad \|\widetilde{Q}\|_2 = \sqrt{\lambda_1(\widetilde{Q}^T\widetilde{Q})} = \tilde{\eta}_1,$$
$$\|(I + E_K)^{-1}\|_2 = \lambda_n^{-1}(GKG^T) = \kappa_n^{-1},$$

and the bounds (2.20) and (2.28), respectively, become

$$|\lambda - 1| \leq \frac{\epsilon_1 + \sqrt{\epsilon_1^2 + 4\tilde{\eta}_1^2}}{2}, \tag{2.38}$$

$$|\lambda - 1| \leq \frac{\epsilon_1 + \eta_1^2\kappa_n^{-1} + \sqrt{\left(\epsilon_1 - \eta_1^2\kappa_n^{-1}\right)^2 + 4\eta_1^2}}{2}. \tag{2.39}$$

Remark 2.6. When $\epsilon_1 = 0$, it is easy to prove that the maximum and minimum eigenvalues of $\widetilde{M}^{-1}A\widetilde{M}^{-T}$ are $1 + \tilde{\eta}_1$ and $1 - \tilde{\eta}_1$, respectively, and the maximum and minimum eigenvalues of $M^{-1}AM^{-T}$ are $1 + (\eta_1^2 + \sqrt{\eta_1^4 + 4\eta_1^2})/2$ and $1 + (\eta_1^2 - \sqrt{\eta_1^4 + 4\eta_1^2})/2$, respectively.

The following results suggest the use of $\widetilde{M}^{-1}$ instead of M^{-1} as the MF preconditioner of A.

THEOREM 2.7. *For any choice of G and F in* (2.17)*, the bound* (2.38) *is narrower than or equal to the bound* (2.39).

Proof. Using the arguments of Corollary 2.3, it follows that $\widetilde{S} = S + \widetilde{H}$, with $\widetilde{H}$ a symmetric positive semidefinite matrix, hence $\|S\widetilde{S}^{-1}\|_2 \leq 1$. In particular,

$$\begin{aligned}\widetilde{S} &= C + FGKG^TF^T + FGB + B^TG^TF^T \\ &= C - B^TK^{-1}B + (F + B^TK^{-1}G^{-1})(GKG^T)(F + B^TK^{-1}G^{-1})^T \\ &= S + R_F(GKG^T)^{-1}R_F^T. \end{aligned} \tag{2.40}$$

Moreover, the matrix $\widetilde{Q}^T\widetilde{Q}$ is similar to $L_S^T\widetilde{S}^{-1}L_SQ^TQ$. In fact, recalling that $Q = -R_F^TL_S^{-T}$ and $\widetilde{Q} = -R_F^TL_{\widetilde{S}}^{-T}$, we obtain

$$R_F^TR_F = L_SQ^TQL_S^T = L_{\widetilde{S}}\widetilde{Q}^T\widetilde{Q}L_{\widetilde{S}}^T, \tag{2.41}$$

from which the similarity follows. Hence

$$\|\widetilde{Q}^T\widetilde{Q}\|_2 = \|L_S^T\widetilde{S}^{-1}L_SQ^TQ\|_2 \leq \|L_S^T\widetilde{S}^{-1}L_S\|_2\|Q^TQ\|_2 \leq \|Q^TQ\|_2 \tag{2.42}$$

because $L_S^T\widetilde{S}^{-1}L_S$ is similar to $S\widetilde{S}^{-1}$. As a consequence, $\tilde{\eta}_1 = \alpha\eta_1$ for some $\alpha \leq 1$. The thesis of the theorem reads

$$\frac{\epsilon_1 + \sqrt{\epsilon_1^2 + 4\tilde{\eta}_1^2}}{2} \leq \frac{\epsilon_1 + \eta_1^2\kappa_n^{-1} + \sqrt{\left(\epsilon_1 - \eta_1^2\kappa_n^{-1}\right)^2 + 4\eta_1^2}}{2}. \tag{2.43}$$

Introducing $\tilde{\eta}_1 = \alpha\eta_1$ in (2.43), after some algebra we obtain

$$\alpha^2 \leq 1 + \frac{\sqrt{\left(\epsilon_1 - \eta_1^2\kappa_n^{-1}\right)^2 + 4\eta_1^2} - \left(\epsilon_1 - \eta_1^2\kappa_n^{-1}\right)}{2}, \tag{2.44}$$

which holds true for any F and G. ∎

Theorem 2.7 suggests that the use of $\widetilde{S}$ in the MF preconditioner is likely to be more appropriate than the exact Schur complement S. In particular, in the theoretical case of $G^TG = K^{-1}$ it is possible to compute explicitly the ratio between the conditioning numbers of the preconditioned matrices $M^{-1}AM^{-T}$ and $\widetilde{M}^{-1}A\widetilde{M}^{-T}$ as follows.

THEOREM 2.8. *If $E_K = 0$, the ratio between the conditioning number of the preconditioned matrices* (2.29) *and* (2.21) *is*

$$r(\eta_1) = \frac{\left(1 + \dfrac{\eta_1^2 + \sqrt{\eta_1^4 + 4\eta_1^2}}{2}\right)^2}{1 + 2\eta_1^2 + 2\sqrt{\eta_1^4 + \eta_1^2}}. \tag{2.45}$$

Proof. If $E_K = 0$, it can be easily verified from (2.40) that $\widetilde{S} = S + R_FR_F^T$ and

$$L_S^{-1}\widetilde{S}L_S^{-T} = I + L_S^{-1}R_FR_F^TL_S^{-T} = I + Q^TQ. \tag{2.46}$$

Recalling from the proof of Theorem 2.7 that $\widetilde{Q}^T\widetilde{Q}$ is similar to $L_S^T\widetilde{S}^{-1}L_SQ^TQ$ we have

$$\|\widetilde{Q}^T\widetilde{Q}\|_2 = \|(I + Q^TQ)^{-1}Q^TQ\|_2. \tag{2.47}$$

Trivially, $(I + Q^TQ)^{-1}Q^TQ$ is symmetric positive definite and has the same eigenvectors as Q^TQ. Denoting by λ an eigenvalue of Q^TQ, the norm $\|(I + Q^TQ)^{-1}Q^TQ\|_2$ is the maximum of the function

$$f(\lambda) = \frac{\lambda}{1 + \lambda}. \tag{2.48}$$

As $f(\lambda)$ monotonically increases with λ, its maximum value is attained for the largest eigenvalue of Q^TQ, i.e., $\|Q^TQ\|_2 = \eta_1^2$. Hence

$$\|\widetilde{Q}^T\widetilde{Q}\|_2 = \frac{\|Q^TQ\|_2}{1 + \|Q^TQ\|_2} \Rightarrow \tilde{\eta}_1 = \sqrt{\frac{\eta_1^2}{1 + \eta_1^2}}. \tag{2.49}$$

The proof is completed by introducing (2.49) in the results of Remark 2.6. ∎

Remark 2.9. The ratio $r(\eta_1)$ monotonically increases with η_1 and takes value 1 for $\eta_1 = 0$, i.e., $F = -B^TK^{-1}G^{-1}$ and $\widetilde{S} = S$. For $\eta_1 \to \infty$, r monotonically diverges to infinity as the second power of η_1. Hence, the sparser or more inaccurate F, the more important is using $\widetilde{S}$ instead of S in the MF preconditioner.

3. Improving the MF performance with low-rank corrections. The framework developed in section 2 provides the formulation for a robust multilevel FSAI preconditioner that can be computed in a stable way for any choice of the fill-in degree used for the basic kernels. Using low fill-in degrees obviously keeps the computational cost for the preconditioner set-up and application under control, but may yield a poor convergence. The quality of the MF preconditioner can be improved by using low-rank corrections. The idea of using low-rank corrections in a multilevel framework has been already introduced in [29], giving rise to the robust multilevel Schur complement-based low-rank (MSLR) preconditioner. The basic concept can be briefly recalled as follows. Define the matrix

$$Y = L_C^{-1} B^T K^{-1} B L_C^{-T} = L_C^{-1}(C - S)L_C^{-T}, \tag{3.1}$$

where L_C is the exact lower factor of C, i.e., $C = L_C L_C^T$. It is easily recognized that the eigenvalues σ_i of Y are such that

$$0 \leq \sigma_{n_2} \leq \cdots \leq \sigma_1 < 1. \tag{3.2}$$

The separation of the eigenvalues θ_i of $X = L_C^T(S^{-1} - C^{-1})L_C$ is larger than that of Y, because

$$\begin{aligned} \theta_i &= \frac{\sigma_i}{1-\sigma_i}, \qquad i = 1, \ldots, n_2, \\ \theta_i - \theta_{i+1} &= \frac{\sigma_i - \sigma_{i+1}}{(1-\sigma_i)(1-\sigma_{i+1})}, \qquad i = 1, \ldots, n_2 - 1. \end{aligned} \tag{3.3}$$

The separation of the eigenvalues of $L_C^{-T} X L_C^{-1} = S^{-1} - C^{-1}$ has a stronger impact on the performance of the MSLR preconditioner [29], however studying X is easier and the main results for X are to some extent still valid for $S^{-1} - C^{-1}$. Equation (3.3) suggests that approximating with a low-rank matrix $(S^{-1} - C^{-1})$ is easier than $(S - C)$ because of the faster eigenvalue decay. A better approximation of S^{-1} can be computed as

$$S^{-1} \simeq C^{-1} + W_k \Theta_k W_k^T \tag{3.4}$$

with $W_k \Theta_k W_k^T$ a rank-k approximation of $L_C^{-T} X L_C^{-1}$ which can be obtained from the eigendecomposition of Y. In fact, by retaining the k largest eigenvalues and corresponding eigenvectors of Y we can write

$$Y \simeq U_k \Sigma_k U_k^T. \tag{3.5}$$

Noting that

$$S^{-1} - C^{-1} = L_C^{-T}[(I - Y)^{-1} - I]L_C^{-1} = L_C^{-T}[Y(I - Y)^{-1}]L_C^{-1}, \tag{3.6}$$

the rank-k correction to $S^{-1} - C^{-1}$ is found by setting

$$\Theta_k = \Sigma_k (I - \Sigma_k)^{-1} \tag{3.7}$$

and $W_k = L_C^{-T} U_k$.

In the original formulation outlined above, the low-rank corrections are used to make the action of C^{-1} closer to that of S^{-1}. By distinction, we use low-rank corrections to improve the action of $\widetilde{S}^{-1}$. A consequence of Corollary 2.3 is that the eigenvalues σ_i of the matrix

$$\overline{Y} = L_{\widetilde{S}}^{-1}(\widetilde{S} - S)L_{\widetilde{S}}^{-T} \tag{3.8}$$

satisfy the condition (3.2). Thus, following the procedure outlined above for C, we can compute the k largest eigenpairs of $\overline{Y}$

$$\overline{Y} \simeq \overline{U}_k \overline{\Sigma}_k \overline{U}_k^T, \tag{3.9}$$

and get the expression of the corrected Schur complement inverse:

$$S^{-1} \simeq \widetilde{S}^{-1} + \overline{W}_k \overline{\Theta}_k \overline{W}_k^T, \tag{3.10}$$

where $\overline{\Theta}_k = \overline{\Sigma}_k(I - \overline{\Sigma}_k)^{-1}$ and $\overline{W} = L_{\widetilde{S}}^{-T}\overline{U}$. However, the idea of correcting the application of $\widetilde{S}^{-1}$ so as to better resemble the one of S^{-1} is not that good. First, the computation of (3.9) may be quite expensive, since every multiplication by S requires a solution of a linear system with K. Second, according to Theorems 2.7 and 2.8, the use of S^{-1} in our multilevel framework is not optimal.

The low-rank corrections can be more effectively implemented in another way. Since we are working in a multilevel framework, $\widetilde{S}^{-1}$ will not be used exactly. Rather, a new approximation, say $\widehat{S}^{-1} \simeq \widetilde{S}^{-1}$, will be computed. Thus, $\widehat{S}$ will be the new target of the low-rank correction. Moreover, as shown by (2.20), reducing $\|E_K\|$ is also useful for improving the convergence. Also this task can be performed by a low-rank correction. Therefore, we use two low-rank correction techniques for the preconditioner set-up:

- *Descending low-rank corrections*: computed at each level, from the first to the last, to reduce $\|E_K\|$;
- *Ascending low-rank corrections*: computed at each level, from the last to the first, to reduce the gap between $\widehat{S}$ and $\widetilde{S}$.

3.1. Descending low-rank corrections. The aim of this correction is to enhance the approximation of the inverse of K. We can define the matrix

$$\overline{Y} = G[(G^TG)^{-1} - K]G^T = I - GKG^T \tag{3.11}$$

obtained from (3.8), where $(G^TG)^{-1}$ and K replace $\widetilde{S}$ and S, respectively, and compute its rank-k approximation:

$$\overline{Y} \simeq \overline{U}_k \overline{\Sigma}_k \overline{U}_k^T. \tag{3.12}$$

Note that the computation of $\overline{U}_k$ and $\overline{\Sigma}_k$ is less expensive than in (3.9), because both G and K are explicitly known. The enhanced preconditioner for K reads

$$K^{-1} \simeq G^TG + \overline{W}_k \overline{\Theta}_k \overline{W}_k^T. \tag{3.13}$$

The eigenvalues of $\overline{Y}$ are bounded from above by 1 as GKG^T is positive definite, but there is not a lower bound in this case. Actually this is not a problem, as we are mainly interested in the computation of the eigenvalue $\overline{\sigma}_i$ of $\overline{Y}$ closest to 1. From the

implementation point of view, it is better to dispose of a symmetrically split operator. Hence, we define

$$\widetilde{G} = (I + \overline{U}_k \overline{\Psi}_k \overline{U}_k^T) G \tag{3.14}$$

in order to have

$$\widetilde{G}^T \widetilde{G} = G^T (I + \overline{U}_k \overline{\Psi}_k \overline{U}_k^T)(I + \overline{U}_k \overline{\Psi}_k \overline{U}_k^T) G = G^T G + \overline{W}_k \overline{\Theta}_k \overline{W}_k^T. \tag{3.15}$$

Recalling that $\overline{W}_k = G^T \overline{U}_k$, the diagonal $k \times k$ matrix $\overline{\Psi}_k$ is simply found by solving

$$I + 2\overline{U}_k \overline{\Psi}_k \overline{U}_k^T + \overline{U}_k \overline{\Psi}_k^2 \overline{U}_k^T = I + \overline{U}_k \overline{\Theta}_k \overline{U}_k^T. \tag{3.16}$$

Using (3.3), the entries of $\overline{\Psi}_k$ read

$$\overline{\psi}_i = -1 + \sqrt{\frac{1}{1 - \overline{\sigma}_i}}, \qquad i = 1, \ldots, k. \tag{3.17}$$

Since GKG^T is positive definite, $\overline{\sigma}_i < 1$ and $\overline{\psi}_i$ is real for any i.

This correction can significantly reduce $\|E_K\|$. However, the update in $\widetilde{G}$ propagates in the other blocks of the preconditioned matrix, potentially shattering the overall procedure efficiency:

(3.18)

$$P_a A P_a^T = \begin{bmatrix} \widetilde{G}K\widetilde{G}^T & \widetilde{G}B \\ B^T\widetilde{G}^T & C \end{bmatrix} = \begin{bmatrix} GKG^T & GB \\ B^TG^T & C \end{bmatrix} + \begin{bmatrix} \overline{U}_k\overline{\Psi}_k\overline{U}_k^T GKG^T + GKG^T\overline{U}_k\overline{\Psi}_k\overline{U}_k^T + \overline{U}_k\overline{\Psi}_k\overline{U}_k^T GKG^T\overline{U}_k\overline{\Psi}_k\overline{U}_k^T & \overline{U}_k\overline{\Psi}_k\overline{U}_k^T GB \\ B^TG^T\overline{U}_k\overline{\Psi}_k\overline{U}_k^T & C \end{bmatrix}.$$

Actually, this is not the case. In fact, the block FSAI $\widetilde{F}$ computed as the (2,1) block of P_b when using $\widetilde{G}$ is the approximate solution of the multiple right-hand-side system:

$$\widetilde{F}^T \simeq -(\widetilde{G}K\widetilde{G}^T)^{-1}\widetilde{G}B = -\widetilde{G}^{-T}K^{-1}B. \tag{3.19}$$

By expanding (3.19) with $\widetilde{G}$ definition, we note that $\widetilde{F}$ can be easily found as:

$$\widetilde{F} = F(I + \overline{U}_k \overline{\Psi}_k \overline{U}_k^T)^{-1}, \tag{3.20}$$

where F is the standard block FSAI computed using G. Moreover, from (3.20) and (3.14) we notice that $FG = \widetilde{F}\widetilde{G}$. This implies that the approximate Schur complement (2.16) is not affected by the use of $\widetilde{F}$ and $\widetilde{G}$:

(3.21)

$$\widetilde{S} = C + \widetilde{F}\widetilde{G}B + B^T\widetilde{G}^T\widetilde{F}^T + \widetilde{F}\widetilde{G}K\widetilde{G}^T\widetilde{F}^T = C + FGB + B^TG^TF^T + FGKG^TF^T.$$

As a consequence, descending low-rank corrections on G have only a local impact with no changes for the following levels.

3.2. Ascending low-rank corrections. We define the matrix $\widehat{Y}$ and compute its rank-k approximation:

$$\widehat{Y} = I - \widehat{G}\widetilde{S}\widehat{G}^T \simeq \widehat{U}_k \widehat{\Sigma}_k \widehat{U}_k^T, \tag{3.22}$$

where $\widehat{G}$ is the lower inverse factor of $\widehat{S}$; i.e., $(\widehat{G}^T\widehat{G})^{-1} = \widehat{S}$, which is explicitly available by the approximation of lower levels. The computation of (3.22) is relatively cheap, because the explicit expression of every matrix is known. The new approximation to $\widetilde{S}$ is given by:

$$\widetilde{S}^{-1} \simeq \widehat{G}^T\widehat{G} + \widehat{W}_k\widehat{\Theta}_k\widehat{W}_k^T, \tag{3.23}$$

where $\widehat{W}_k = \widehat{G}^T\widehat{U}_k$ and $\widehat{\Theta}_k = \widehat{\Sigma}_k(I - \widehat{\Sigma}_k)^{-1}$. Notice that during the set-up, we use a split update, as done for the descending low-rank corrections, because it is operatively necessary to compute $\widehat{G}\widetilde{S}\widehat{G}^T$. However, during the preconditioner application, the use of (3.23) is more efficient as it only requires one update.

4. Numerical results. The multilevel FSAI preconditioner with low-rank corrections (MFLR) can be implemented and applied through a recursive function (Algorithms 4.1 and 4.2, respectively). The key parameters for the MFLR computation and application are the matrix A, the number of levels n_l, the current level index l, and the size of the ascending and descending low-rank corrections, alr and dlr. Other threshold parameters are actually needed by the inner kernels, such as those required

Algorithm 4.1. MFLR Set-up.

1. **Recursive Function** MFLR_SetUp(l,n_l,A,alr,dlr)
2. **if** $l < (n_l - 1)$ **then**
3. Partition A_l as $\begin{bmatrix} K & B \\ B^T & C \end{bmatrix}$;
4. Compute the adaptive FSAI approximation G of K such that $G^TG \simeq K^{-1}$;
5. Set $P_a = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix}$;
6. Compute $P_b = \begin{bmatrix} I & 0 \\ F & I \end{bmatrix}$ as the adaptive block FSAI approximation of $P_aA_lP_a^T$;
7. Compute $\widetilde{S} = C + FGB + B^TG^TF^T + FGKG^TF^T$;
8. Compute a rank-k approximation $\overline{U}_k\overline{\Sigma}_k\overline{U}_k^T$ of $\overline{Y} = I - GKG^T$;
9. Set $\widetilde{G} = (I + \overline{U}_k\overline{\Psi}_k\overline{U}_k^T)G$ with $\overline{\Psi}_k = (I - \overline{\Sigma}_k)^{-1/2} - I$;
10. Set $P = \begin{bmatrix} \widetilde{G} & 0 \\ FG & I \end{bmatrix}$;
11. $[Q_{alr}, Q_M]$ = MFLR_SetUp($l+1$,n_l,$\widetilde{S}$,alr,dlr);
12. Use Q_{alr} and Q_M to compute the rank-k correction $\widehat{W}_k$ and $\widehat{\Theta}_k$ as in (3.23);
13. Set $\widetilde{P} = \begin{bmatrix} I & 0 \\ 0 & I + \widehat{W}_k\widehat{\Theta}_k\widehat{W}_k^T \end{bmatrix}$;
14. Push $\widetilde{P}$ in the head of Q_{alr};
15. Push P in the head of Q_M;
16. **return** Q_{alr}, Q_M;
17. **else**
18. Compute the adaptive FSAI approximation G of A such that $G^TG \simeq A^{-1}$;
19. Set $Q_{alr} = \emptyset$;
20. Set $Q_M = \{G\}$;
21. **return** Q_{alr}, Q_M;
22. **end if**

Algorithm 4.2. MFLR Application.

1. **Recursive Function** MFLR_Apply(l,n_l,Q_{alr},Q_M,$\mathbf{x}$)
2. **if** $l < (n_l - 1)$ **then**
3. Pop P from the head of Q_M;
4. Pop $\widetilde{P}$ from the head of Q_{alr};
5. Compute $\mathbf{y} = P\mathbf{x}$;
6. Partition $\mathbf{y}$ into $\mathbf{y}_1 = \mathbf{y}(1 : n_1)$ and $\mathbf{y}_2 = \mathbf{y}(n_1 + 1 : n_1 + n_2)$;
7. Compute $\mathbf{z}_2$ = MFLR_Apply($l + 1$,n_l,Q_{alr},Q_M,$\mathbf{y}_2$);
8. Form $\mathbf{z}$ with $\mathbf{y}_1$ and $\mathbf{z}_2$;
9. Compute $\mathbf{y} = \widetilde{P}\mathbf{z}$;
10. Update $\mathbf{y} \leftarrow P^T\mathbf{y}$;
11. Push P in the head of Q_M;
12. Push $\widetilde{P}$ in the head of Q_{alr};
13. **return** $\mathbf{y}$;
14. **else**
15. Pop G from the head of Q_M;
16. Compute $\mathbf{y} = G^T G\mathbf{x}$;
17. Push G in the head of Q_M;
18. **return** $\mathbf{y}$;
19. **end if**

by the computation of G and F, and the level-of-fill control of the matrix-matrix products. For the sake of readability, these parameters are dropped from the list of arguments of the MFLR_SetUp and MFLR_Apply functions. The preconditioner is stored using the lists Q_{alr} and Q_M, and the parallelization is performed equally partitioning each level among the available cores using the OpenMP directives. The reason for this choice, which limits the algorithm scalability and the problem size, is that the aim of this work is mainly the analysis of the MFLR preconditioner behavior. The MPI implementation is an ongoing task.

The MFLR preconditioner behavior is investigated on a set of test cases. First, its theoretical properties are verified in a small problem. Second, the MFLR sensitivity to the user-specified parameters that control the preconditioner quality and density are analyzed with an extensive numerical experimentation on a medium-size matrix. Finally, the performance in a few large-size problems is considered.

4.1. Theoretical properties. We analyze the `bcsstk38` matrix from the University of Florida Sparse Matrix Collection [9]. This SPD matrix has 8,032 rows and 355,460 nonzeroes, and has been scaled so as to have a unitary diagonal. Its eigenspectrum is provided in the leftmost frame of Figure 3. The matrix is uniformly partitioned into two levels ($n_1 = n_2 = 4,016$). The approximate Schur complement $\widetilde{S}$ is computed using (2.9), where F is the block FSAI of A obtained using the dynamic strategy introduced in [16] and a variable number $k_{F,\max}$ of entries retained per row. The nonzero eigenspectrum of $(\widetilde{S} - S)$ is shown in the rightmost frame of Figure 3 and is strictly positive for any F, as expected from Corollary 2.3.

The analysis is performed by introducing step-by-step new ingredients to the MFLR preconditioner. In particular, (i) the exact inverse of K is used to evaluate the effect of using either $\widetilde{S}$ or S, (ii) the approximation $G^T G \simeq K^{-1}$ is introduced, and (iii) low-rank corrections are added to improve the preconditioner.

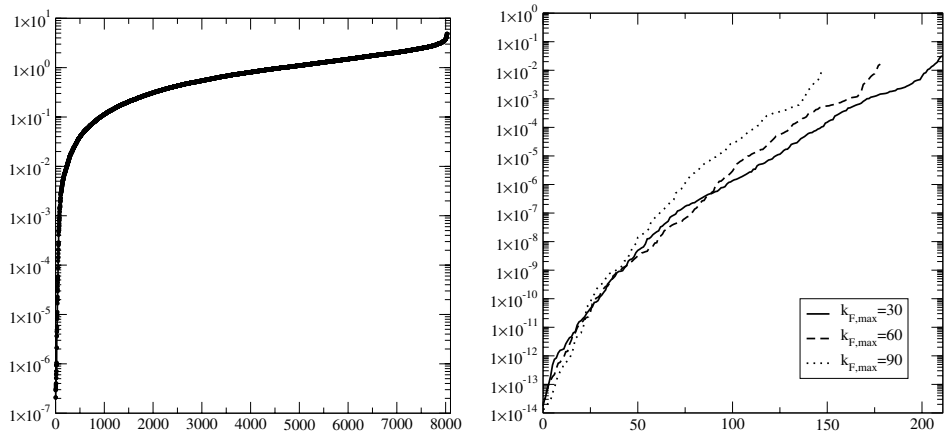

FIG. 3. `bcsstk38` *test case: eigenspectrum of A (left) and* $(\widetilde{S} - S)$ *(right) for different block FSAI F.*

TABLE 1

`bcsstk38` *test case: comparison between the conditioning numbers of* (2.29) *and* (2.21) *varying* F *with* $\|E_K\| = 0$. λ *and* $\tilde{\lambda}$ *denote the eigenvalues of* $M^{-1}AM^{-T}$ *and* $\widetilde{M}^{-1}A\widetilde{M}^{-T}$*, respectively.*

$k_{F,\max}$	η_1	$\tilde{\eta}_1$	$[\lambda_n, \lambda_1]$	$[\tilde{\lambda}_n, \tilde{\lambda}_1]$	$r(\eta_1)$
30	4.395	0.975	[0.0470, 21.273]	[0.0249, 1.975]	5.709
60	2.867	0.944	[0.0988, 10.121]	[0.0558, 1.944]	2.939
90	2.248	0.914	[0.1448, 6.9069]	[0.0863, 1.914]	2.153

First, we want to verify that the main claim of section 2, i.e., the use of $\widetilde{S}$ is more appropriate than that of S in the presented multilevel framework, is actually correct using the exact inverse of K. The ratio between the conditioning numbers of $M^{-1}AM^{-T}$ and $\widetilde{M}^{-1}A\widetilde{M}^{-T}$, as obtained in Theorem 2.8, is reported in Table 1 for different block FSAI F. Should F be computed exactly as a full matrix, there would be no difference between $\widetilde{M}^{-1}$ and M^{-1}. By distinction, decreasing the quality of F means increasing $\eta_1 = \|Q\|_2$ and $\tilde{\eta}_1 = \|\widetilde{Q}\|_2$. As a consequence, the effectiveness of $\widetilde{M}^{-1}$ improves with respect to M^{-1}; i.e., the sparser F, the more important it is to use $\widetilde{S}$ instead of S. For this test problem, the ratio $r(\eta_1)$ between the conditioning numbers of (2.29) and (2.21) increases up to 5.709.

Theorem 2.8 no longer holds if we introduce the approximation G^TG for K^{-1}. However, it is still true that the theoretical eigenvalue bounds for $\widetilde{M}^{-1}$ are tighter than those for M^{-1}. The matrix G is computed as the FSAI of K using the adaptive strategy implemented in the FSAIPACK software package [20]. The quality of G is controlled by the maximum number of entries $k_{G,\max}$ retained per row. First of all, notice that $\widetilde{S}$ computed as in (2.3) might be indefinite if G is not accurate enough. For instance, even with $k_{G,\max} = 250$ the approximate Schur complement $(C - B^TG^TGB)$ still has one negative eigenvalue. Changing the fill-in degree of G modifies the values obtained from the bounds (2.38) and (2.39), which become tighter and tighter as $k_{G,\max}$ increases. Such bounds along with the actual eigenvalue intervals are reported in Table 2 for different choices of G and F. It can be observed that the role played by ϵ_1 and $\tilde{\eta}_1$, i.e., a measure of the quality of G and F, respectively, has a similar impact on the actual eigenvalue distribution of $\widetilde{M}^{-1}A\widetilde{M}^{-T}$, as expected from the right-hand side of inequality (2.38). Hence, one may argue that improving both G and F is

TABLE 2
`bcsstk38` *test case: eigenvalue distribution of* (2.29) *and* (2.21) *varying* F *and* G. *The same notation as in Table* 1 *is used.*

$k_{G,\max}$	$k_{F,\max}$	ϵ_1	η_1	$\tilde{\eta}_1$	$[\lambda_n, \lambda_1]$	$[\tilde{\lambda}_n, \tilde{\lambda}_1]$	bound (2.39)	bound (2.38)
30	30	1.076	4.505	0.812	[1.8e-04, 47.3]	[1.7e-04, 2.1]	19473.3	2.512
30	60	1.076	3.181	0.790	[1.8e-04, 33.2]	[1.8e-04, 2.1]	9707.6	2.494
30	90	1.076	2.617	0.648	[1.8e-04, 29.1]	[1.8e-04, 2.1]	6570.8	2.380
60	30	1.240	4.393	0.718	[3.9e-04, 40.0]	[3.9e-04, 2.2]	8291.3	2.569
60	60	1.240	3.037	0.641	[4.0e-04, 29.5]	[4.0e-04, 2.2]	3963.8	2.512
60	90	1.240	2.466	0.589	[4.0e-04, 25.8]	[4.0e-04, 2.2]	2613.6	2.475
90	30	1.092	4.300	0.738	[6.6e-04, 36.0]	[6.5e-04, 2.1]	5353.9	2.464
90	60	1.092	2.935	0.651	[6.8e-04, 26.8]	[6.7e-04, 2.1]	2495.6	2.396
90	90	1.092	2.434	0.620	[6.9e-04, 23.3]	[6.8e-04, 2.1]	1716.8	2.372

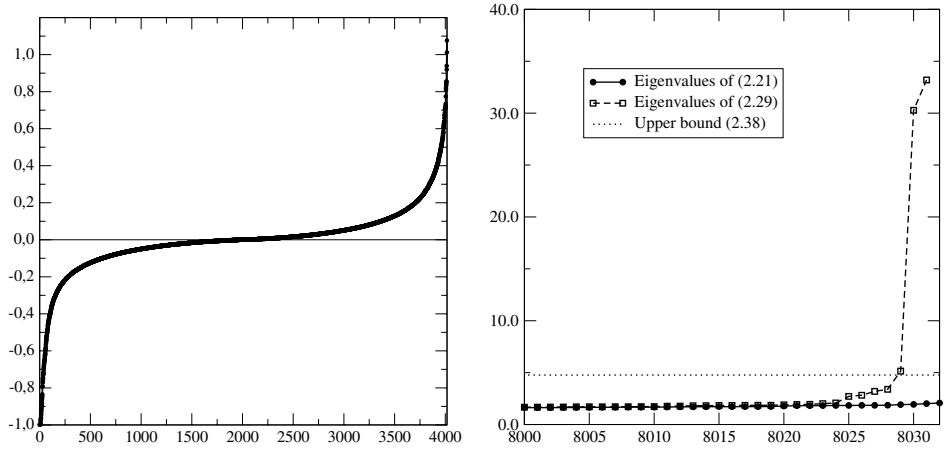

FIG. 4. `bcsstk38` *test case: eigenspectrum of* E_K *(left) and rightmost eigenvalues of the preconditioned matrices* (2.29) *and* (2.21)*(right) for* $k_{G,max} = 30$ *and* $k_{F,max} = 60$. *The upper bound* (2.38) *is also shown.*

essential for a better MF performance. On the other hand, the bound (2.39) is less significant because of the presence of κ_n^{-1}, which can be quite large. Nonetheless, the actual eigenvalue distribution of $M^{-1}AM^{-T}$ is in any case worse than that of $\widetilde{M}^{-1}A\widetilde{M}^{-T}$.

As an example, Figure 4 shows the eigenvalue distribution of E_K for $k_{G,\max} = 30$, and the preconditioned matrices (2.29) and (2.21) for the same $k_{G,\max}$ and $k_{F,\max} = 60$. The matrix E_K is indefinite, with the eigenvalues almost equally distributed between negative and positive values approximately in the interval $[-1, 1]$. The matrix $\widetilde{M}^{-1}A\widetilde{M}^{-T}$ has a more favorable eigenvalues distribution than $M^{-1}AM^{-T}$.

Finally, we add to the frame descending and ascending low-rank corrections. We fix $k_{G,\max} = 30$ and $k_{F,\max} = 60$, and change the number of eigenpairs computed to correct either G (*dlr*) or $\widehat{S}^{-1}$ (*alr*). Their effect on the variation of the conditioning number of $\widetilde{M}^{-1}A\widetilde{M}^{-T}$ is shown in Table 3. Using both low-rank strategies greatly helps reduce the conditioning number. For example, in this case retaining 20 eigenpairs for both corrections yields a reduction of the conditioning number of about two orders of magnitude, i.e., from about 24,000 to 500. By distinction, notice that correcting $\widetilde{S}^{-1}$ towards S^{-1} is not as effective, as expected from the MFLR theoretical

TABLE 3

`bcsstk38` *test case: effect of descending and ascending low-rank corrections, dlr and alr, respectively, with* $k_{G,max} = 30$ *and* $k_{F,max} = 60$. *The target of ALR correction is* $\widetilde{S}^{-1}$ *and* S^{-1} *in the upper and lower table, respectively.*

alr target	*alr*	*dlr*	$[\lambda_n, \lambda_1]$	λ_1/λ_n
$\widetilde{S}^{-1}$	0	0	[1.682e-04, 2.337e+00]	2.389e+04
	0	10	[4.391e-04, 2.337e+00]	5.321e+03
	0	20	[4.438e-04, 2.337e+00]	5.265e+03
	10	0	[1.682e-04, 2.076e+00]	1.234e+04
	10	10	[4.391e-04, 2.077e+00]	4.731e+03
	10	20	[4.438e-04, 2.137e+00]	4.816e+03
	20	0	[1.744e-04, 2.076e+00]	1.190e+04
	20	10	[3.380e-03, 2.077e+00]	6.145e+02
	20	20	[4.413e-03, 2.137e+00]	4.843e+02
S^{-1}	10	0	[1.753e-04, 1.467e+01]	8.371e+04
	10	10	[4.342e-03, 1.482e+01]	3.414e+03
	10	20	[6.308e-03, 1.493e+01]	2.367e+03
	20	0	[1.772e-04, 3.220e+01]	1.817e+05
	20	10	[5.475e-03, 3.270e+01]	5.973e+03
	20	20	[9.065e-03, 3.290e+01]	3.630e+03

properties. It is also interesting to observe that the largest eigenvalue is the most sensitive to the selection of the target matrix for the ALR corrections, while it is insensitive to the DLR corrections, which mainly affect the smallest eigenvalue.

4.2. Sensitivity to user-specified parameters. To test the influence of the user-specified parameters that control the MFLR behavior, we use the medium-size matrix `Cube` arising from a homogeneous and uniformly discretized structural problem by P1 finite elements, with 190,581 unknowns and 7,531,389 nonzero entries. The linear system (1.1) is solved by using an MFLR-preconditioned conjugate gradient (PCG) method, with an exit tolerance on the relative residual equal to 10^{-8}. The right-hand-side $\mathbf{b}$ is such that the solution $\mathbf{x}$ is the vector with components $x_j = j+1$, $j = 1, \ldots, n$. All tests reported here are obtained using a machine equipped with Intel(R) Xeon(R) E5-2680 v2 processors at 2.80 GHz and 256 Gbyte of RAM. Each CPU has 10 cores. For these preliminary tests, just one thread is used. The MFLR preconditioner is implemented in Fortran90, with the code compiled by the Intel Fortran compiler using the -O3 optimization level. We also used BLAS and LAPACK routines from the Intel Math Kernel Library. For the computation of the eigenpairs needed by the low-rank corrections, we used the Laneig software, that is part of the Filtlan package [10].

The user-specified parameters that control the MFLR quality and performance are as follows:

1. n_l: number of levels. Therefore, $(n_l - 1)$ is the number of computed Schur complements;
2. ϵ_G: tolerance for the adaptive computation of G, see [20];
3. ϵ_F: tolerance for the adaptive computation of F, see [16];
4. *dlr*: DLR correction size, i.e., number of eigenpairs used to enrich G. This is a local improvement;
5. *alr*: ALR correction size, i.e., number of eigenpairs used to enrich $\widehat{S}^{-1}$. This is a global improvement, in the sense that the Schur complement size grows up to the size of the zero-level partition.

TABLE 4
`Cube` *test case: MFLR performance varying* n_l *and* ϵ_G ($\epsilon_F = 10^{-2}$, $dlr = alr = 0$).

ϵ_G	n_l	n_{it}	ρ	T_p	T_s	ϵ_G	n_l	n_{it}	ρ	T_p	T_s
10^{-1}	10	930	0.43	14.5	26.8	10^{-2}	10	640	0.77	34.5	21.9
	20	843	0.73	132.0	34.3		20	591	1.17	265.9	29.3
	50	664	1.47	349.9	49.1		50	503	2.04	441.7	41.6
	100	612	2.78	515.1	79.2		100	464	3.67	612.9	67.0
10^{-3}	10	446	1.97	268.4	24.0	10^{-4}	10	362	3.83	2232.9	37.3
	20	416	2.53	881.2	29.5		20	338	4.67	3741.6	35.7
	50	386	3.45	829.9	41.8		50	329	5.74	2404.2	47.4
	100	367	5.71	956.9	66.3		100	318	8.88	2053.8	73.1

The `Cube` test matrix is reordered with the Reverse Cuthill McKee algorithm and uniformly partitioned into equal-size levels. The results are evaluated in terms of number of iterations, n_{it}, time needed to compute the preconditioner, T_p, the time spent in the PCG iterations, T_s, and the preconditioner density, ρ, defined as

$$\rho = \frac{1}{\mathrm{nnz}(A)} \sum_{i=0}^{n_l-1} \left(\mathrm{nnz}(Q^i_{alr}) + \mathrm{nnz}(Q^i_M) \right), \tag{4.1}$$

where $\mathrm{nnz}(\cdot)$ gives the number of nonzero entries stored for the operator $(\cdot)$ and all the remaining symbols are defined as in Algorithm 4.1. For these preliminary tests, the matrix-matrix products are fully computed, i.e., any use of thresholds and/or levels of fill is avoided.

Similarly to the previous section, the analysis is carried out step by step, adding one ingredient at a time. First, the performance of the robust MF preconditioner with no low-rank corrections is investigated by changing the fill-in degree of G and F. Then, low-rank corrections are added to the frame to identify their effect. It is worth mentioning that only the robust MF preconditioner is addressed here, because the standard multilevel FSAI algorithm breaks down, giving rise to indefinite Schur complements.

Table 4 shows the results obtained by varying the number of levels n_l and the tolerance ϵ_G. The other tolerance ϵ_F is set to 10^{-2} and no low-rank corrections are applied, i.e., $dlr = alr = 0$. As expected, the iteration count decreases progressively as ϵ_G decreases, i.e., G is more accurate, and n_l grows. The preconditioner density also increases, so that the set-up burden and the cost per iteration grows. Notice that, in particular, the set-up time can become very heavy. Although such a cost could be reduced by introducing thresholds and level-of-fills in the matrix-matrix computations with no substantial loss in the PCG acceleration, it appears that the proposed multilevel approach is of interest whenever the preconditioner can be reused several times, e.g., in eigensolvers or in some transient simulations, so that its set-up cost can be properly amortized and set apart in the present analysis. Hence, we focus on the solution time T_s only. With the most efficient ϵ_G value, i.e., 10^{-2}, we vary ϵ_F and n_l. The results, provided in Table 5, show that the MFLR preconditioner appears to be less sensitive to the quality of F than of G. In fact, the iteration count is more stable than in Table 4.

With $\epsilon_G = 10^{-2}$, $\epsilon_F = 10^{-2}$, and $n_l = 10$, we test the effects of corrections. Table 6 shows the impact of using either DLR and ALR corrections. Increasing the rank size, the number of iterations decreases for both approaches, but ALR corrections have a stronger impact on the solution time T_s. This is somewhat expected because ALR corrections have a global effect on the overall preconditioner, while

TABLE 5
`Cube` *test case: MFLR performance varying* n_l *and* ϵ_F *(*$\epsilon_G = 10^{-2}$*,* $dlr = alr = 0$*).*

ϵ_F	n_l	n_{it}	ρ	T_p	T_s	ϵ_F	n_l	n_{it}	ρ	T_p	T_s
10^{-1}	10	660	0.65	29.1	22.8	10^{-2}	10	640	0.77	34.5	21.9
	20	632	0.94	216.8	29.5		20	591	1.17	265.9	29.3
	50	601	1.61	369.8	47.8		50	503	2.04	441.7	41.6
	100	579	2.81	498.0	78.7		100	464	3.67	612.9	67.0
10^{-3}	10	621	1.13	85.7	27.0	10^{-4}	10	585	2.304	1163.3	35.1
	20	544	1.79	639.2	32.9		20	460	3.445	4513.5	39.3
	50	403	3.21	1129.2	43.0		50	309	5.550	4020.9	43.6
	100	348	6.98	2278.7	71.2		100	255	11.192	6029.8	68.6

TABLE 6
`Cube` *test case: MFLR performance varying either dlr or alr (*$\epsilon_G = \epsilon_F = 10^{-2}$*,* $n_l = 10$*).*

dlr	n_{it}	ρ	T_p	T_s	alr	n_{it}	ρ	T_p	T_s
5	534	0.90	71.3	29.3	5	289	1.35	95.1	17.0
10	485	1.01	66.1	29.4	10	213	1.92	78.0	15.1
20	450	1.24	65.9	25.5	20	196	3.06	88.1	14.1
50	405	1.92	75.0	30.1	50	186	6.48	214.5	36.7

TABLE 7
`Cube` *test case: MFLR performance varying both dlr and alr (*$\epsilon_G = \epsilon_F = 10^{-2}$*,* $n_l = 10$*).*

dlr	alr	n_{it}	ρ	T_p	T_s
1	1	525	0.92	76.3	34.4
1	10	204	1.95	87.7	14.8
1	20	188	3.09	112.5	18.5
10	1	381	1.13	71.7	25.0
10	10	148	2.15	83.6	10.9
10	20	134	3.29	100.0	11.3
20	1	341	1.35	86.6	23.9
20	10	132	2.38	100.5	10.4
20	20	123	3.52	100.1	10.8

DLR corrections improve locally the current level approximation of K^{-1}. Finally, the combined effect of both corrections is investigated in Table 7. This allows for obtaining the best result in terms of both iteration count and solution time T_s decrease. In particular, the latter is more than halved with respect to the case with no corrections with at least $dlr = alr = 10$. The solver acceleration is paid with a bigger set-up cost, which makes the MFLR approach interesting when the preconditioner can be recycled.

Although the present analysis cannot be thoroughly exhaustive, it is possible to observe that dlr and alr appear to be the most sensitive user-specified parameters. The number of levels n_l strongly depends on the size of the matrix and should be selected such that each level is not too small. In the `Cube` test case it can be seen that with more than 20 levels, i.e., less than 10,000 unknowns per level, the preconditioner application cost grows quickly and is no longer compensated by the iteration count reduction. By distinction, the selection of the tolerances ϵ_G and ϵ_F does not appear to be overly difficult. In fact, the MFLR performance does not change much with respect to a variation of these parameters in the interval $[10^{-3}, 10^{-1}]$.

4.3. Preconditioner performance. The computational performance of the MFLR preconditioner is finally evaluated in a set of large size SPD matrices taken from the University of Florida Sparse Matrix Collection [9]. The main properties of

TABLE 8
Test matrices.

	Size	Number of nonzeroes
af_shell3	504,855	17,588,875
af_shell8	504,855	17,579,155
Emilia_923	923,136	40,373,538
Geo_1438	1,437,960	60,236,322
StocF_1465	1,465,137	21,005,389

TABLE 9
Computational performance of the adaptive FSAI and MFLR preconditioners for the test matrices of Table 8.

	Adaptive FSAI				MFLR			
	n_{it}	ρ	T_p [s]	T_s [s]	n_{it}	ρ	T_p [s]	T_s [s]
af_shell3	963	0.89	88.8	63.9	126	4.50	237.8	31.5
af_shell8	1033	0.86	81.2	68.0	131	3.22	174.8	28.6
Emilia_923	1513	0.11	5.0	128.5	575	0.25	82.8	80.4
Geo_1438	491	0.31	59.4	89.6	456	0.49	159.8	110.4
StocF_1465	937	0.93	51.3	133.4	936	1.66	260.4	222.2

the selected test cases are provided in Table 8, while the best results obtained by the MFLR preconditioner in terms of iteration CPU time T_s are given in Table 9. As a benchmark, in the same table the performance provided by the native adaptive FSAI algorithm is also shown. All the CPU times are obtained using 1 thread on the computer defined in section 4.2. It is worth mentioning that the native adaptive FSAI is used as a benchmark because the standard multilevel FSAI generally gives rise to indefinite Schur complements. Hence, only the robust MF algorithm presented here can be effectively used, with or without low-rank corrections. The latter may improve the convergence rate but have no effect as to the preconditioner robustness.

As already observed, the MFLR preconditioner set-up can be quite expensive, especially because of the computation of the eigenpairs needed by the low-rank correction procedures. However, its effectiveness in the iteration count and CPU time can be quite significant. For instance, in the af_shell3 and af_shell8 test cases, n_{it} is approximately reduced by a factor 10 and T_s is more than halved. Hence, the use of the MFLR preconditioner can be of great interest whenever the set-up time can be amortized along several linear solves. On the other hand, if the reduction of the number of iterations is marginal, such as in the Geo_1438 and StocF_1465 test cases, the adaptive FSAI proves more efficient than the MFLR preconditioner.

For the sake of a comparison, the computational performance obtained by other multilevel packages, such as ILUPACK [3] and the Trilinos_ML [13], is provided in Table 10. The ILUPACK parameters are set so as to obtain approximately the same memory footprint as MFLR, i.e., a density close to the one provided in Table 9. By distinction, the default parameters are retained for Trilinos_ML. The results show that the ILUPACK performance in terms of solution time T_s is close to MFLR for af_shell3 and af_shell8, much better for StocF_1465, and much lower for Emilia_923 and Geo_1438. In particular, the ILU density for the Emilia_923 test case must be increased up to 3.8 to attain convergence in just 62 iterations, otherwise the convergence is not reached after 3,000 iterations. Recalling that ILUPACK cannot thoroughly exploit a growing parallelism of the computational architecture, the comparison with MFLR appears to be quite satisfactory. As far as the results with Trilinos_ML are concerned, it can be observed that with the default parameters

TABLE 10
Computational performance of the test matrices of Table 8 with ILUPACK and Trilinos_ML.

	ILUPACK – GMRes				Trilinos_ML – CG			
	n_{it}	ρ	T_p [s]	T_s [s]	n_{it}	ρ	T_p [s]	T_s [s]
af_shell3	79	4.12	64.0	24.6	356	1.01	1.2	80.1
af_shell8	78	4.12	62.2	24.4	309	1.02	1.2	57.4
Emilia_923	> 3000	0.40	34.0	–	586	1.15	6.5	367.0
Geo_1438	878	0.50	57.3	385.8	118	1.15	9.6	93.7
StocF_1465	108	1.50	49.2	33.3	759	1.16	3.8	278.0

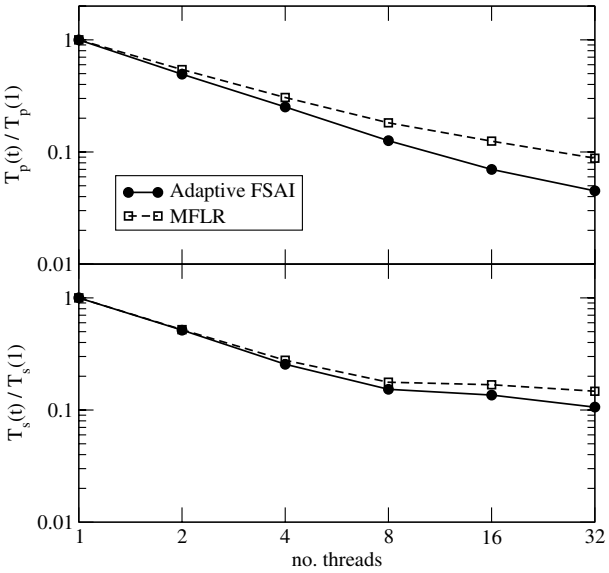

FIG. 5. *Strong scalability test for a regular* 300^3 *Laplacian.*

the MFLR preconditioner is always superior, with the exception of the Geo_1438 test case, where the two performances are roughly equivalent.

Finally, to show the potential parallelism of the MFLR preconditioner, a scalability test has been carried out on a discrete Laplacian computed over a regular $300 \times 300 \times 300$ grid. The numerical experiment is performed on the Marconi cluster at the CINECA Center of High Performance Computing, Bologna, Italy. The cluster is still under construction and, at the moment of writing this work, it consists of 1,512 nodes with 128 Gbyte of RAM memory each. Every node is equipped with 2 Intel Xeon E5-2697v4 Broadwell processors at 2.3 GHz with 36 cores. This preliminary implementation of MFLR makes use of shared-memory parallelism through the OpenMP directives, hence in the strong scalability test provided in Figure 5, a single node only is used with up to 32 threads. The strong scalability of the MFLR preconditioner is compared to that of the Adaptive FSAI algorithm as far as both the set-up and the iteration time, T_p and T_s, are concerned. In particular, Figure 5 provides the speed-up,

$$S_p = \frac{T_p(t)}{T_p(1)}, \qquad S_s = \frac{T_s(t)}{T_s(1)}, \tag{4.2}$$

with $T_p(t)$ and $T_s(t)$ the set-up and iteration wall-clock times measured with t threads. It has been already verified that the native Adaptive FSAI practically has an ideal

speed-up [20], according to the hardware properties of the specific computational architecture used for the numerical experiments. In the case of the multi-core processors of the Marconi cluster, it is well-known that it is virtually impossible to obtain ideal speed-ups with iterative solvers as these algorithms are bandwidth limited, being characterized by a low bit per flop ratio. The MFLR preconditioner is theoretically less parallel than FSAI, because of the intrinsic sequentiality introduced by the multilevel framework. Nevertheless, Figure 5 shows that the strong scalability is only marginally affected by such a sequentiality and the MFLR preconditioner still preserves a good degree of parallelism.

5. Conclusions. The development of a multilevel framework is often useful in several applications. However, using the FSAI preconditioner as the basic kernel in a standard multilevel approach may give rise to some difficulties related with the approximations introduced in the computation of the Schur complement at each level. With SPD problems, such a Schur complement might be indefinite, thus causing a breakdown of the multilevel algorithm.

The present work develops a robust multilevel framework for SPD matrices based on the use of adaptive FSAI as the main kernel. An alternative way of computing the Schur complement is introduced so as to guarantee its positive definiteness independent of the preconditioner sparsity. A theoretical analysis is formulated with the aim of providing appropriate bounds for the eigenspectrum of the preconditioned matrix. The multilevel FSAI preconditioner is further enhanced by introducing low-rank corrections at both a local and a global level, namely DLR and ALR corrections, respectively, thus producing the MFLR preconditioning framework.

The MFLR preconditioner has been investigated in a set of test problems to analyze (i) the relative influence of the user-specified parameters controlling the algorithm set-up, and (ii) the computational performance and potential scalability in a parallel environment. The numerical results show that the proposed approach is generally able to significantly accelerate the solver convergence rate still preserving a good degree of parallelism. At the present time, the solver acceleration is paid off by a large set-up cost, which is mainly due to the computation of the eigenpairs needed by the low-rank corrections. The increase of the cost for building the preconditioner with respect to the native adaptive FSAI makes this approach attractive especially for those applications where the preconditioner can be effectively recycled along a number of linear solves.

The main goal of this work was to prove the robustness and effectiveness of a multilevel framework in explicit preconditioning. Further investigations will be devoted in the development of a faster set-up stage by using, for instance, randomized approaches while computing low-rank corrections [15], and enforcing sparsity in the resulting factors.

REFERENCES

[1] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J. Y. L'Excellent, and C. Weisbecker, *Improving multifrontal methods by means of block low-rank representations*, SIAM J. Sci. Comp., 37 (2015), pp. A1451–A1474.

[2] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[3] M. Bollhöfer, J. I. Aliaga, A. F. Martin, and E. S. Quintana-Ortí, ILUPACK, in *Encyclopedia of Parallel Computing*, D. Padua, ed., Springer, New York, 2011, pp. 917–926.

[4] Y. Bu, B. Carpentieri, Z. Shen, and T. Z. Huang, *A hybrid recursive multilevel incomplete factorization preconditioner for solving general linear systems*, Appl. Numer. Math., 104 (2016), pp. 141–157.

[5] B. Carpentieri, I. S. Duff, L. Giraud, and G. Sylvand, *Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations*, SIAM J. Sci. Comput., 27 (2005), pp. 774–792.
[6] B. Carpentieri, J. Liao, and M. Sosonkina, *VBARMS: A variable block algebraic recursive multilevel solver for sparse linear systems*, J. Comp. Appl. Math., 259 (2014), pp. 164–173.
[7] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, Acta Numer., 3 (1994), pp. 61–143.
[8] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.
[9] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), pp. 1–25.
[10] H. Fang and Y. Saad, *A filtered Lanczos procedure for extreme and interior eigenvalue problems*, SIAM J. Sci. Comput., 34 (2012), pp. 2220–2246
[11] M. Ferronato, *Preconditioning for sparse linear systems at the dawn of the* 21*st century: History, current developments, and future perspectives*, ISRN Appl. Math., (2012), doi:10.5402/2012/127647.
[12] A. Franceschini, V. A. Paludetto Magri, C. Janna, and M. Ferronato, *Multilevel approaches for FSAI preconditioning*, Numer. Linear Algebra Appl., submitted.
[13] M. W. Gee, C. M. Siefert, J. J. Hu, R. S. Tuminaro, and M. G. Sala, *ML* 5.0 *Smoothed Aggregation User's Guide*, 2006–2649 Sandia National Laboratories, 2006.
[14] W. Hackbusch, *A sparse matrix arithmetic based on H-matrices. Part* I*: Introduction to H-matrices*, Computing, 62 (1999), pp. 89–108.
[15] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[16] C. Janna and M. Ferronato, *Adaptive pattern research for block FSAI preconditioning*, SIAM J. Sci. Comput. 33 (2011), pp. 3357–3380.
[17] C. Janna, M. Ferronato, and G. Gambolati, *Multilevel incomplete factorizations for the iterative solution of non-linear finite element problems*, Internat. J. Numer. Methods Engrg., 80 (2009), pp. 651–670.
[18] C. Janna, M. Ferronato, and G. Gambolati, *A Block FSAI-ILU parallel preconditioner for symmetric positive definite linear systems*, SIAM J. Sci. Comput., 32 (2010), pp. 2468–2484.
[19] C. Janna, M. Ferronato, and G. Gambolati, *Enhanced block FSAI preconditioning using domain decomposition techniques*, SIAM J. Sci. Comput., 35 (2013), pp. S229–S249.
[20] C. Janna, M. Ferronato, F. Sartoretto, and G. Gambolati, *FSAIPACK: A software package for high performance FSAI preconditioning*, ACM Trans. Math. Software, 41 (2015), Art. 10.
[21] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditioning* I*. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[22] Z. Li, Y. Saad, and M. Sosonkina, *pARMS: a parallel version of the algebraic recursive multilevel solver*, Numer. Lin. Alg. Appl., 10 (2003), pp. 485–509.
[23] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A2069–A2095.
[24] L. C. McInnes, B. Smith, H. Zhang, and R. T. Mills, *Hierarchical Krylov and nested Krylov methods for extreme-scale computing*, Parallel Comput., 40 (2014), 17–31.
[25] P. Raghavan and K. Teranishi, *Parallel hybrid preconditioning: Incomplete factorization with selective sparse approximate inversion*, SIAM J. Sci. Comput. 32 (2010), pp. 1323–1345.
[26] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl. 9 (2002), pp. 359–378.
[27] J. A. Scott and M. Tůma, *On positive semidefinite modification schemes for incomplete Cholesky factorization*, SIAM J. Sci. Comput., 36 (2014), pp. A609–A633.
[28] S. Wang, X. S. Li, J. Xia, Y. Situ, and M. V. de Hoop, *Efficient scalable algorithms for solving dense linear systems with hierarchically semiseparable structures*, SIAM J. Sci. Comp., 35 (2013), pp. C519–C544.
[29] Y. Xi, R. Li and Y. Saad, *An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 235–259.
[30] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.
[31] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.
[32] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.