SIAM J. MATRIX ANAL. APPL.
Vol. 27, No. 4, pp. 998–1018

AGGREGATION-BASED ALGEBRAIC MULTILEVEL PRECONDITIONING*

YVAN NOTAY†

Abstract. We propose a preconditioning technique that is applicable in a "black box" fashion to linear systems arising from second order scalar elliptic PDEs discretized by finite differences or finite elements with nodal basis functions. This technique is based on an algebraic multilevel scheme with coarsening by aggregation. We introduce a new aggregation method which, for the targeted class of applications, produces semicoarsening effects whenever desirable, while the number of nodes is decreased by a factor of about 4 at each level, regardless of the problem at hand. Moreover, the number of nonzero entries per row in the successive coarse grid matrices remains approximately constant, ensuring small set up cost and modest memory requirements.

This aggregation technique can be used in an algebraic multigrid (AMG)-like framework, but better results are obtained with an algebraic multilevel scheme based on a block approximate factorization of the matrix. In this scheme, the block pivot corresponding to fine grid nodes is approximated by a modified incomplete LU (MILU) factorization. To enhance robustness and avoid any potential breakdown, the coarsening process is refined by recasting as "coarse" fine grid nodes for which the corresponding pivot in this MILU factorization would be negative or too small.

Numerical results display the efficiency, the scalability, and the robustness of the resulting preconditioner on a wide set of discrete scalar PDE problems, ranging from the two-dimensional Poisson equation to three-dimensional convection-diffusion problems with high Reynolds number and strongly varying convection.

Key words. algebraic multigrid, multilevel, aggregation, preconditioning

AMS subject classification. 65F15

DOI. 10.1137/04061129X

1. Introduction.

We consider large sparse, possibly nonsymmetric, linear systems

$$A\,\mathbf{u} \;=\; \mathbf{b} \tag{1.1}$$

arising from the discretization of second order scalar elliptic PDEs. To solve such systems, algebraic multigrid (AMG) and algebraic multilevel methods are now quite popular [5, 6, 7, 8, 9, 10, 11, 12, 17, 19, 24, 26, 30, 34]. They follow the paradigm of classical "geometric" multigrid or multilevel methods (e.g., [2, 3, 22, 16, 33]) but often do not require any information besides the matrix structure and the matrix entries. They can thus be applied to problems without any particular structure, and, somewhat sacrificing efficiency on "easy" cases, they aim at achieving robustness on a large class of problems without need for any specific tuning.

Most of these methods are based on the recursive use of a two-level scheme, and basically the same two ingredients enter the definition of these two-level schemes.

The first of these ingredients is a partitioning of the unknowns (or nodes) into fine grid nodes and coarse grid nodes, according to which the system matrix is permuted

*Received by the editors July 8, 2004; accepted for publication (in revised form) by E. Chow May 6, 2005; published electronically March 17, 2006. This research was supported by the "Fonds National de la Recherche Scientifique," Maître de recherches.

http://www.siam.org/journals/simax/27-4/61129.html

†Service de Métrologie Nucléaire (C.P. 165/84), Université Libre de Bruxelles, 50, Av. F.D. Roosevelt, B-1050 Brussels, Belgium (ynotay@ulb.ac.be).

(in general only implicitly) and written in the 2×2 block form

$$A = \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} . \tag{1.2}$$

Here and in the following, F corresponds to the F set, i.e., the set of fine grid nodes, and C to the C set, i.e., the set of coarse grid nodes.

The second ingredient is an "interpolation" matrix J_{FC} that interpolates on the F set a vector defined on the C set. With this matrix, one associates the prolongation

$$p = \begin{pmatrix} J_{FC} \\ I \end{pmatrix} \tag{1.3}$$

and the Galerkin coarse grid matrix

$$\widehat{A}_C = p^T A p = A_{CC} + A_{CF} J_{FC} + J_{FC}^T A_{FC} + J_{FC}^T A_{FF} J_{FC} . \tag{1.4}$$

There are several ways to define these ingredients. Among them, aggregation-based schemes play a particular role. On the one hand, they are appealing because they are simple and cheap. Indeed, each fine grid node is then associated with a unique coarse grid node; that is, J_{FC} has only one nonzero entry per row. It is therefore as sparse as possible, as is the associated coarse grid matrix $\widehat{A}_C$. Consequently, resources needed to compute and store the preconditioner are guaranteed to be low.

On the other hand, aggregation-based schemes do not in general lead to robust and scalable performances, except when modified in one way or another, in which case they lose their original simplicity [8, 32, 34]. However, this observation is essentially true for schemes that follow the AMG paradigm. In [24], an algebraic multilevel method based on a block approximate factorization process has been found scalable and robust while using coarse grid matrices computed by an aggregation rule.

Consider the exact block factorization of the matrix (1.2):

$$A = \begin{pmatrix} I & \\ A_{CF} A_{FF}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{FF} & \\ & S_A \end{pmatrix} \begin{pmatrix} I & A_{FF}^{-1} A_{FC} \\ & I \end{pmatrix} , \tag{1.5}$$

where $S_A = A_{CC} - A_{CF} A_{FF}^{-1} A_{FC}$ is the Schur complement. The latter kind of method, in its basic two-level variant, approximates this factorization with

$$B = \begin{pmatrix} I & \\ A_{CF} P_{FF}^{-1} & I \end{pmatrix} \begin{pmatrix} P_{FF} & \\ & S \end{pmatrix} \begin{pmatrix} I & P_{FF}^{-1} A_{FC} \\ & I \end{pmatrix} , \tag{1.6}$$

where P_{FF} is some approximation to A_{FF}, and S some approximation to S_A, for which one may take the (rescaled) coarse grid matrix $\widehat{A}_C$.

Here, we present a method based on these principles, introducing two major enhancements to the method from [24].

First, the coarsening process considered in [24] was essentially the one from the standard AMG method [26, 32], with an a posteriori assignment of each fine grid node to a neighboring coarse grid node. This process therefore inherits the drawbacks of that approach: sensitivity to the strong/weak coupling threshold and slow coarsening in case of low connectivity, severe anisotropy, or dominating convection. For instance,

if the (main part of the) matrix originates from a two-dimensional (2D) five point or three-dimensional (3D) nine point stencil, the F/C partitioning is essentially a red-black one, which is not very desirable. Note also that aggressive coarsening, as one may use with standard AMG [32, section A.7.1.2], is not very convenient in the context of an aggregation scheme, since the a posteriori assignment of fine grid nodes is not easy when some of them are not directly connected to any coarse grid node.

Here we develop another approach, in which aggregates are directly formed by grouping the nodes pairwise. Since the resulting coarsening would be too slow, this process is repeated once; that is, we form quadruplets by grouping pairwise the pairs resulting from the first pass; we call this "double pairwise aggregation." The number of nodes is then decreased by a factor of about 4 at each level, regardless of whether, for instance, the underlying PDE is 2D or 3D, with or without anisotropy. On the other hand, pairs are formed by following the strongest negative coupling. As will be seen below, this makes the process essentially insensitive to the strong/weak coupling threshold.

The second enhancement concerns the approximation P_{FF} to A_{FF}. As in [24], we rely on modified incomplete LU (MILU) factorizations, because they possess all the desirable properties (see below). However, they are not always well defined for general matrices. Most often, the latter fact is harmless because the submatrix A_{FF} resulting from the coarsening process is fairly well conditioned, and (M)ILU factorizations of well-conditioned matrices tend to be breakdown-free. Nevertheless, there is an obvious danger in using this approach without safeguard. Here, we introduce a "dynamic" MILU algorithm that reallocates to the C set the nodes for which the corresponding pivot in the factorization of A_{FF} would be negative or too small (relative to the corresponding diagonal entry in A_{FF}). Note that we are comfortable in doing so since, on account of the fast coarsening resulting from the double pairwise aggregation strategy, there is plenty of room for moving nodes from F to C before one gets in trouble with the coarsening speed or the set-up cost.

To conclude this introduction, we briefly mention some other approaches that can be followed to build a preconditioner based on a two-level block factorization of the form (1.6). First, as shown in [25], basically any (algebraic) multigrid-like scheme can be used to define the F/C partitioning and the coarse grid matrix that approximates the Schur complement. Here we have proposed an aggregation-based scheme because we found it simple, cheap, and nevertheless robust.

Alternatively, the approximate Schur complement can be computed in an ILU-like fashion, that is, formed by incompletely eliminating the fine grid unknowns, using, for instance, some threshold to drop small entries. This idea is followed in, e.g., [28, 30, 31]. In these works, the F/C partitioning is based on finding a (block) independent set in the matrix graph of A, so that A_{FF} is (block) diagonal. Its inverse is therefore sparse, allowing us to compute the approximate Schur complement by applying some dropping rule to S_A. Consequently, the method inherits the versatility and the wide range of applications of ILU-based methods. However, the coarsening is actually much slower than with our approach. Moreover, the number of nonzero entries per row tends to increase in the successive coarse grid matrices, and most often the multilevel reduction has to be stopped before the coarse grid matrix is sufficiently small to be factorized exactly. The resulting method is then likely to be not competitive with multigrid-based schemes on very large PDE problems.

The remainder of this paper is organized as follows. In section 2, we introduce our double pairwise aggregation strategy. Its use in the context of an AMG-like algorithm is briefly discussed in section 3. In section 4, we present the multilevel block

approximate factorization preconditioner with more details, including a complete description and discussion of our dynamic MILU factorization algorithm. The results of numerical experiments are reported in section 5.

2. Double pairwise aggregation.

2.1. Simple pairwise aggregation.

There are two ways to define an aggregation algorithm. One way consists of defining an F/C partitioning and next associating each F node with a unique C node, so as to form aggregates. The other way consists of forming these aggregates directly. This is, for instance, the approach followed in [8, 34]. In these works, the coarse level unknowns are treated as purely algebraic entities, and no F or C set is defined explicitly. This is possible, however, only with an AMG-like algorithm. If one wants to use a block two-level preconditioner of the form (1.6), it is mandatory to further select one C node in each aggregate, the other nodes forming the F set.

Our aggregation algorithm given below (Algorithm 2.1) forms aggregates directly by grouping the nodes pairwise. One picks up one unmarked node i at a time, according to some priority rule designed so as to favor a regular covering of the matrix graph (see below). A tentative aggregate is then formed by grouping this node with the unmarked node with which it is most strongly negative coupled, that is, (one of) the unmarked node(s) j for which a_{ij} is minimal. The strong/weak coupling threshold β then acts as a filter. That is, essentially as in AMG [32, p. 473], one defines the set of nodes S_i to which i is strongly negative coupled by

$$S_i = \left\{ j \neq i \mid a_{ij} < -\beta \max_{a_{ik}<0} |a_{ik}| \right\} ,$$

and the tentative pair is rejected if and only if $j \notin S_i$. In the latter case the node i initially picked up forms a singleton, that is, a C node without any associated F node. Note, however, that this may occur only if all nodes k for which a_{ik} is minimal have already been marked (or do not correspond to a negative coupling because all couplings in row i are positive). For the kind of applications targeted here, this occurs only incidentally, for instance near boundaries. In general, the algorithm progresses smoothly along the direction of strongest negative coupling, and most nodes picked up are effectively associated with (one of) the neighbor(s) with which they are most strongly negative coupled. Hence, the strong/weak coupling threshold has only a slight influence on the coarsening process. This justifies the relatively large size of the proposed default value in Algorithm 2.1, protecting against unsafe associations without dramatic impact on the coarsening speed.

Once a pair has been formed, we always put the node initially picked up in F and its companion in C. This guarantees that each node in F is strongly negative coupled to the associated node in C, whereas the converse is not necessarily true and is even usually not true when the matrix is nonsymmetric.

In Algorithm 2.1, we also include an optional check for strongly diagonally dominant rows, for which the corresponding node is put in F without any companion node in C (that is, the corresponding row in J_{FC} is zero). This guarantees a proper treatment of finite element matrices in which Dirichlet boundary conditions have been imposed by adding a big number to the corresponding diagonal elements. More generally, it is heuristically clear that, when a row is strongly diagonally dominant enough, a fast reduction of the error at the corresponding node can be obtained without multilevel enhancement.

ALGORITHM 2.1 (pairwise aggregation).

Input: $n \times n$ matrix $A = (a_{ij})$;
strong/weak coupling threshold β (default: $\beta = \mathbf{0.75}$);
logical parameter *CheckDD*.
Output: partitioning $(F\,,\,C)$ of $[1\,,\,n]$;
subset (aggregates) G_i, $i \in C$, such that $G_i \cap C = \{i\}$
and $G_i \cap G_j = \emptyset$ for $i \neq j$.
Definition: for all i: $S_i = \left\{\, j \neq i \mid a_{ij} \,<\, -\beta \max_{a_{ik}<0} |a_{ik}| \right\}$.
Initialization: if (*CheckDD*): $F = \left\{ i \mid a_{ii} \,>\, 3 \sum_{j\neq i} |a_{ij}| \right\}$, otherwise: $F = \emptyset$;
$C = \emptyset$;
$U = [1\,,\,n] \backslash F$;
for all i: $m_i = |\left\{\, j \in U \mid i \in S_j \,\right\}|$.
Algorithm: while $U \neq \emptyset$, do
1. select $i \in U$ with minimal m_i
2. select $j \in U$ such that $a_{ij} = \min_{k\in U} a_{ik}$
3. if $j \in S_i$:
   3a. $C = C \cup \{j\}$, $F = F \cup \{i\}$, $G_j = \{i, j\}$, $U = U \backslash \{i, j\}$
   3b. update: $m_k = m_k - 1$ for $k \in S_i$, and $m_k = m_k - 1$ for $k \in S_j$
   otherwise:
   3a′. $C = C \cup \{i\}$, $G_i = \{i\}$, $U = U \backslash \{i\}$
   3b′. update: $m_k = m_k - 1$ for $k \in S_i$

Eventually we discuss item 1 of Algorithm 2.1. We give priority to the node(s) i with minimal m_i, where m_i is the number of unmarked nodes that are strongly negative coupled to i (that is, m_i is the number of sets S_j to which i belongs and that correspond to an unmarked node j). As stated above, this favors a regular covering of the matrix graph. Probably other rules could work as well. We consider this one because we observe that in some nonsymmetric examples, at any stage of the process, there is at least one node j for which $m_j = 0$. Then, if this node is not selected, it has many chances to become a singleton, since no unmarked node is strongly negative coupled to it.

As output of the algorithm, we have, besides the F/C partitioning, also the aggregates G_j, $j \in C$, that is, the set of F nodes associated with j, plus j itself. The interpolation matrix J_{FC} may then formally be defined by

$$\forall\, i \in F\,,\, j \in C\,:\; (J_{FC})_{ij} \;=\; \begin{cases} 1 & \text{if } i \in G_j, \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

whereas the prolongation p satisfies

$$\forall\, i \in [1\,,\,n]\,,\, j \in C\,:\; (p)_{ij} \;=\; \begin{cases} 1 & \text{if } i \in G_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

However, there is no need to form these matrices explicitly, and the Galerkin coarse grid matrix (1.4) is more easily computed with

$$\forall\, i\,,\, j \in C\,:\; \left(\widehat{A}_C\right)_{ij} \;=\; \sum_{k\in G_i} \sum_{\ell \in G_j} a_{k\ell}\ . \tag{2.3}$$

That is, the entries in the coarse grid matrix are obtained by summing the entries in A that connect the different aggregates.

2.2. Forming pairs of pairs. Coarsening by simple pairwise aggregation is relatively slow, which does not favor optimal performance of multilevel methods. A faster coarsening can be obtained by repeating the process, defining aggregates by forming pairs of pairs—more precisely, by forming pairs of aggregates from the first pass, some of which are singletons.

Forming pairs of pairs has already been considered in [8], where a strategy is proposed that is based on the connectivity pattern in the resulting aggregates. This favors a coarsening that mimics the geometric coarsening well. However, as is well known, geometric coarsening is not desirable in all cases, and good algebraic coarsening algorithms should, for instance, automatically provide a semicoarsening like partitioning in the presence of strong anisotropy or dominating convection.

We therefore propose another strategy, based on the repeated use of our pairwise aggregation algorithm. That is, we compute the coarse grid matrix resulting from a first application of Algorithm 2.1, and apply this algorithm again to the latter matrix, forming pairs of C nodes from the first pass, which implicitly define pairs of pairs. Details are given in Algorithm 2.2 below. For the class of applications considered in this paper, the resulting aggregates are mostly quadruplets, with some triplets, pairs, and singletons left. Consequently, the coarsening ratio n/n_c (where $n_c = |C|$ is the number of C nodes) is slightly less than 4, regardless of the problem at hand.

On 2D model problems, the procedure mimics geometric coarsening well if the coefficients are isotropic, whereas strong anisotropy induces a nonstandard semicoarsening $h \to 4\,h$ in the direction of strong coupling. We apply the same procedure in 3D cases as well. On model anisotropic problems, this induces a standard planewise semicoarsening or an $h \to 4\,h$ linewise semicoarsening, according to the case at hand. It is more difficult to figure out what comes out in isotropic cases, but the numerical experiments show that the approach is robust (see section 5 below).

ALGORITHM 2.2 (double pairwise aggregation).

Input: $n \times n$ matrix $A = (a_{ij})$;
strong/weak coupling threshold β (default: $\beta = \mathbf{0.75}$);

Ouput: partitioning $(F\,,\,C)$ of $[1\,,\,n]$;
subset (aggregates) G_i, $i \in C$, such that $G_i \cap C = \{i\}$ and $G_i \cap G_j = \emptyset$ for $i \neq j$.

Algorithm:

1. Apply Algorithm 2.1 to A with threshold β and *CheckDD* = *True*.
   Output: $(F_1\,,\,C_1)$ and $G_i^{(1)}$, $i \in C_1$.
2. Compute the auxiliary matrix $A_1 = (a_{ij}^{(1)})$, $i, j \in C_1$, with
$$a_{ij}^{(1)} \;=\; \sum_{k \in G_i^{(1)}} \sum_{\ell \in G_j^{(1)}} a_{k\ell}\ .$$
3. Apply Algorithm 2.1 to A_1 with threshold to β and *CheckDD* = *False*.
   Output: $(F_2\,,\,C_2)$ and $G_i^{(2)}$, $i \in C_2$.
4. $C = C_2$, $F = F_1 \cup F_2$, $G_i = \cup_{j \in G_i^{(2)}} G_j^{(1)}$, $i \in C$.

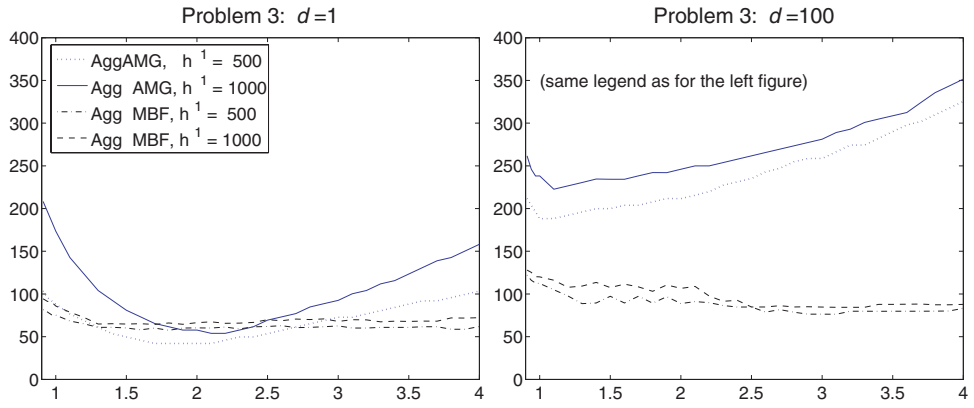

FIG. 1. *Relative solution cost (total number of flops divided by the number of flops for* 1 *unpreconditioned CG iteration) as a function of the inverse of the scaling parameter.*

3. Aggregation-based AMG. With the ingredients defined in the preceding section, it is not difficult to set up an AMG scheme. One needs only to choose a smoother and a cycling strategy. Since this approach is not central in our work, we refer to, e.g., [32] for more details.

Indeed, the combined use of AMG with simple aggregation is not advised in general, as explained in some detail in [32, section A.9]. To summarize in a few words, consider a model 2D example for which our strategy mimics well the geometric coarsening. Despite this similarity, two major differences appear between a geometric and an aggregation-based multigrid scheme. Firstly, the interpolation defined by (2.1) is zero order accurate, whereas standard geometric multigrid uses (at least) first order accurate interpolation, as required by the theory [16, sections 3.5 and 6.3.2]. Next, the coarse grid matrices defined by (2.3) resemble their geometric counterparts but appear to be badly scaled.

This latter point is best illustrated by the following numerical experiment. We consider the AMG scheme resulting from the algorithms given in section 2, except that the coarse grid matrices are rescaled. We use V-cycle and symmetric Gauss–Seidel smoothing with one pre- and one postsmoothing step at each level; the coarse grid matrix at the coarsest level is factorized exactly, and the number of levels is fixed as indicated in section 4.3; the resulting AMG scheme is used as a preconditioner for the conjugate gradient (CG) method. We consider Problem 3 that is fully described in section 5. This is a five point approximation of a 2D Laplace-like equation, with some Neumann boundary conditions and anisotropies and discontinuities of order d, where d is a parameter; for $d = 1$, we thus have a model isotropic constant coefficient problem.

Figure 1 displays the relative solution cost against the inverse of the scaling parameter, that is, the number by which all coefficients in the successive coarse grid matrices have been divided. One sees that for the model problem ($d = 1$), dividing all coefficients by 2 delivers near-optimal results and scalable performances. This was already observed in [8]. However, performances deteriorate as d increases, and it seems better not to scale the coarse grid matrices then. Generally speaking, results appear sensitive to the scaling parameter, and it is not obvious how to find a rule that could be applied uniformly.

Figure 1 also displays the results obtained with our multilevel block approximate factorization (MBF) preconditioner, considering the method as described in the next section, except that all coarse grid matrices are uniformly scaled according to the indicated parameter (instead of the more sophisticated default rule (4.4) that is justified below). On the model problem, this approach is roughly equivalent to aggregation-based AMG with appropriate scaling. But it appears fairly insensitive to the scaling parameter and much more robust with respect to anisotropies and discontinuities in the coefficients of the underlying PDE.

4. Aggregation-based multilevel block factorization. As already stated, this approach consists of using as a building block a two-level preconditioner of the form (1.6) with

$$S = \alpha \widehat{A}_C, \tag{4.1}$$

where $\widehat{A}_C$ is the coarse grid matrix computed from (2.3) and α some scaling factor.

There are several ways to explain why this may be efficient despite the mitigated results obtained with AMG. First, applying AMG requires effectively using the interpolation (2.1) to transfer information from the coarse grid to the fine grid. With the block approximate factorization approach, the interpolation is used as only a tool to build the coarse grid matrix, and plays otherwise no role in the algorithm. Hence, it does not matter if this interpolation resembles a zero order one, as long as the resulting coarse grid matrix mimics well those obtained with more sophisticated schemes. More generally, one may observe that if the F/C partitioning resembles a geometric one and if the coarse grid matrix mimics well a discretization on the coarse grid, then the multilevel block approximate factorization preconditioner is close to its "geometric" counterpart as defined, for instance, in [22], where optimal convergence properties are proved for the symmetric case (see also [23] for the nonsymmetric case).

Next, as we saw in the preceding section, the main problem with aggregation-based coarse grid matrices is their bad scaling. We also saw on an example that this issue is much less critical with multilevel block factorization preconditioning. To understand why, observe that the eigenvalues of $B^{-1}A$ with B defined by (1.6) are the same as those of $\widetilde{B}^{-1}\widetilde{A}$, where

$$\widetilde{A} = \begin{pmatrix} I & \\ -A_{CF}\,P_{FF}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{FF} & A_{FC} \\ A_{CF} & A_{CC} \end{pmatrix} \begin{pmatrix} I & -P_{FF}^{-1}\,A_{FC} \\ & I \end{pmatrix}$$

and

$$\widetilde{B} = \begin{pmatrix} P_{FF} & \\ & S \end{pmatrix}.$$

The scaling of S thus plays exactly the same role as the scaling of one of the diagonal blocks in a 2×2 block-diagonal preconditioning. From a theoretical point of view, this scaling is near-optimal when, considering both matrices $P_{FF}^{-1}\widetilde{A}_{FF}$ and $S^{-1}\widetilde{A}_{CC}$, the spectrum of the better conditioned one is contained in the spectrum of the worse conditioned one. In practice, one does not have to expect large variations in the performances as long as this condition approximately holds. This explains the observations from Figure 1: on the one hand, it is worthwhile to rescale the coarse grid matrices, but, on the other hand, results are about the same with any scaling factor whose inverse lies between 2.5 and 3.5.

Now, to complete the description of our multilevel block factorization preconditioner, we need to consider the following points: (1) the choice of the approximation P_{FF} to A_{FF} and the associated refinement of the F/C partitioning; (2) the scaling factor α for which a rule has still to be provided; (3) the cycling strategy used to turn the two-level preconditioner based on (1.6) into a multilevel scheme. These points are discussed in the next three subsections.

4.1. Definition of P_{FF} and refinement of the F/C partitioning. In general, A_{FF} as resulting from our coarsening algorithm is fairly well conditioned, and it is therefore not difficult to set up close easy-to-invert approximations P_{FF}. However, from the theoretical developments and numerical results in [20, 25], one knows that P_{FF} may *not* be any approximation. Roughly speaking, it has to be such that

$$\widetilde{J}_{FC} \;=\; -\,P_{FF}^{-1}\,A_{FC} \tag{4.2}$$

also defines a "correct" interpolation for (algebraically) smooth vectors. Without going into the details, this requirement may be understood from the relation

$$\begin{aligned} B^{-1} &= \begin{pmatrix} I & -P_{FF}^{-1}A_{FC} \\ & I \end{pmatrix} \begin{pmatrix} P_{FF}^{-1} & \\ & S^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -A_{CF}\,P_{FF}^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} I \\ 0 \end{pmatrix} P_{FF}^{-1} \begin{pmatrix} I & 0 \end{pmatrix} \;+\; \begin{pmatrix} -P_{FF}^{-1}A_{FC} \\ I \end{pmatrix} S^{-1} \begin{pmatrix} -A_{CF}\,P_{FF}^{-1} & I \end{pmatrix}. \end{aligned}$$

This preconditioner acts therefore as an additive two-level scheme, with prolongation from the coarse variables space defined by

$$\widetilde{p} \;=\; \begin{pmatrix} -P_{FF}^{-1}A_{FC} \\ I \end{pmatrix}.$$

If A is a symmetric M-matrix with nonnegative row-sum, theoretical and numerical results show that MILU factorizations of A_{FF} possess all the desired properties [20, 22, 25]. These factorizations are computed so as to satisfy the row-sum criterion

$$P_{FF}\,\mathbf{e}_F \;=\; A_{FF}\,\mathbf{e}_F,$$

where $\mathbf{e}_F = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix}^T$. Note that if $A_{FF}\,\mathbf{e}_F + A_{FC}\,\mathbf{e}_C \approx 0$, this rule ensures that the interpolation (4.2) is a good approximation for the constant vector. This is precisely the main requirement for a "correct" interpolation for the class of applications targeted in this paper (that is, second order scalar elliptic PDEs discretized by finite differences or finite elements with nodal basis functions). We therefore decided to use MILU factorizations as well for non–M- or nonsymmetric matrices. We stress, however, that this approach is likely to be insufficient for other classes of problems like, for instance, coupled systems of PDEs, which are the subject of further investigations.

Now, MILU factorizations are not necessarily well defined if A_{FF} is not an M-matrix with nonnegative row-sum. Of course, instabilities (small or negative pivots) are much less likely than when factorizing the whole matrix, since the restriction to F nodes strengthens the relative weight of the diagonal elements. Nevertheless, as is, the method is not breakdown-free. More generally, there is no guarantee that the conditioning of A_{FF} is good enough so that a cheap MILU factorization will deliver a nice approximation P_{FF}.

This leads us to consider a "dynamic" factorization algorithm that reallocates to the C set the nodes for which the corresponding pivot would be too small relative

ALGORITHM 4.1 (adaptive MILU factorization of A_{FF}).

Input: $n \times n$ matrix $A = (a_{ij})$;
stability threshold γ (default: $\gamma = \frac{3}{5}$);
partitioning $(F\,,\,C)$ of $[1\,,\,n]$ (tentative F/C partitioning),
associated aggregates G_i, $i \in C$;
Boolean matrix (η_{ij}) indicating where fill-in is allowed.

Output: partitioning $(F\,,\,C)$ of $[1\,,\,n]$ (final F/C partitioning),
associated aggregates G_i, $i \in C$;
MILU factorization $P_{FF} = L_{FF}\,Q_{FF}^{-1}U_{FF}$ of A_{FF}
(where $\mathrm{diag}(L_{FF}) = \mathrm{diag}(U_{FF}) = Q_{FF}$) such that $Q_{FF} \geq \gamma\,\mathrm{diag}(A_{FF})$($Q_{FF} = (q_{ij})$, $L_{FF} = (l_{ij})$, $U_{FF} = (u_{ij})$).

Initialization: *Repeat = True.*

Algorithm:

1. *Repeat = False.*
2. (re)initialize:
   $Q_{FF} = \mathrm{diag}(A_{FF})$, $L_{FF} = \mathrm{lower}(A_{FF})$, $U_{FF} = \mathrm{upper}(A_{FF})$.
3. for $k = 1, \ldots, n$, $k \in F$:
   if $q_{kk} \geq \gamma\,a_{kk}$:
   (eliminate row and column k in A_{FF} according to the MILU algorithm)
   $\forall\ i > k$, $i \in F$:
   $q_{ii} \;=\; q_{ii} - \frac{l_{ik}\,u_{ki}}{q_{kk}}$
   $\forall\ j > k$, $j \in F$, $j \neq i$:
   if $\eta_{ij} = 0$: $q_{ii} \;=\; q_{ii} - \frac{l_{ik}\,u_{kj}}{q_{kk}}$
   if $\eta_{ij} = 1$ and $j > i$: $u_{ij} \;=\; u_{ij} - \frac{l_{ik}\,u_{kj}}{q_{kk}}$
   if $\eta_{ij} = 1$ and $j < i$: $l_{ij} \;=\; l_{ij} - \frac{l_{ik}\,u_{kj}}{q_{kk}}$
   otherwise: $F = F\backslash\{k\}$, $C = C \cup \{k\}$;
   for $i \in C$ such that $k \in G_i$: $G_i = G_i\backslash\{k\}$;
   $G_k = \{k\}$;
   Repeat = True.
4. If (*Repeat*), GoTo 1, possibly decreasing the value of γ.

to the corresponding diagonal element in A_{FF}. The detailed procedure is given in Algorithm 4.1. In this algorithm, γ is the threshold for this reallocation. That is, letting $P_{FF} = L_{FF}\,Q_{FF}^{-1}U_{FF}$ with $\mathrm{diag}(L_{FF}) = \mathrm{diag}(U_{FF}) = Q_{FF}$ be the tentative MILU factorization of A_{FF}, a node i is reallocated to the C set if and only if $(Q_{FF})_{ii} < \gamma(A_{FF})_{ii}$.

We iterate on the factorization because removing a posteriori a row and a column disturbs the relation $(P_{FF}\,\mathbf{e}_F)_i = (A_{FF}\,\mathbf{e}_F)_i$ on previously computed rows i. Hence, if some nodes are moved to F, the factorization has to be recomputed in a second pass. In principle, during this refactorization, some more nodes may be moved, in which case a third factorization would be necessary, and so on. To maintain low set-up cost, it is mandatory to keep small the number of such refactorizations. That is why we suggest decreasing the threshold γ from one pass to the next (see step 4 of Algorithm 4.1). There is, of course, some tradeoff here between set-up cost and stability issues. We could, however, hardly recommend any particular strategy because, in the numerical

experiments we have performed so far, when some nodes were moved to C during the first pass, *the second factorization always satisfied the stability criterion* $Q_{FF} \geq \gamma\,\mathrm{diag}(A_{FF})$ *for the value of the threshold* γ *used during the first pass.* Hence, any possible strategy would anyway accept the F/C partitioning resulting from the first pass.

We now discuss the proposed default value $\gamma = \frac{3}{5}$. This may look relatively severe, but remember that one of the goals is to guarantee the good conditioning of $P_{FF}^{-1}A_{FF}$. In [1, Lemma 10.2] it is proven that, in the symmetric case, if

$$\lambda_{\min}\Big(Q_{FF} - \xi\big(A_{FF} - (L_{FF} - Q_{FF}) - (U_{FF} - Q_{FF})\big)\Big) \;\geq\; 0 \tag{4.3}$$

holds for some $\xi > \frac{1}{2}$, then

$$\lambda_{\max}\left(P_{FF}^{-1}A_{FF}\right) \;\leq\; \frac{1}{2 - \xi^{-1}}\;,$$

whereas $\lambda_{\min}\left(P_{FF}^{-1}A_{FF}\right) \geq 1$ holds if A_{FF} is an M-matrix.[1] If one discards all fills (that is, if one uses Algorithm 4.1 with $\eta_{ij} = 0$ for all $i\,,\,j$), one has $A_{FF} - (L_{FF} - Q_{FF}) - (U_{FF} - Q_{FF}) = \mathrm{diag}(A_{FF})$, and condition (4.3) reduces to $Q_{FF} \geq \xi\,\mathrm{diag}(A_{FF})$, which is precisely the one targeted by the algorithm with $\xi = \gamma$. The default threshold then ensures $\lambda_{\max}\left(P_{FF}^{-1}A_{FF}\right) \leq 3$.

In our experiments we do not allow fill-in, but we accept the update of entries already nonzero in A_{FF} (that is, we set $\eta_{ij} = 1 \iff a_{ij} \neq 0$). Then, it is difficult to know whether (4.3) still holds for some $\xi > \frac{1}{2}$, and what could be the relationship between ξ and γ. We nevertheless observe that the proposed default threshold is efficient to improve the conditioning of $P_{FF}^{-1}A_{FF}$, especially in highly nonsymmetric cases for which there is no supporting theory.

In most cases, this improvement is obtained nearly for free, only a fairly small amount of nodes being moved from F to C. The effect is more dramatic on some difficult highly nonsymmetric problems, for which n_c was increased by up to 20% at some coarser levels. Note that we are anyway comfortable in this respect, since, on account of the fast coarsening resulting from our double pairwise aggregation strategy, there is plenty of room for moving nodes from F to C before recovering a prohibitively slow coarsening.

4.2. The scaling factor α. We already saw that coarse grid matrices computed by the aggregation rule (2.3) are somewhat overweight. We also saw that it is not necessary to finely tune the scaling parameter α in (4.1). For instance, in light of Figure 1, taking uniformly $\alpha = \frac{1}{3}$ seems enough.

The main goal of this subsection is therefore to report our observation that in a few cases results tend to be (slightly) improved by taking somewhat larger values for α. These cases are more precisely those where a significant number of nodes are moved from F to C by Algorithm 4.1. One should not be surprised at this observation: considering the formula (2.3), clearly, fewer nodes on average in the aggregates implies less overweighting. Therefore, instead of applying uniformly $\alpha = \frac{1}{3}$, we propose the default rule $\alpha = \frac{4\,n_C}{3\,n}$, that is

$$S \;=\; \frac{4\,n_C}{3\,n}\;\widehat{A}_C \tag{4.4}$$

(observe that $\alpha = \frac{1}{3}$ when $\frac{n}{n_C} = 4$).

[1] $\lambda_{\min}(C)$ and $\lambda_{\max}(C)$ stand for, respectively, the smallest and the largest eigenvalues of C.

4.3. From two- to multilevel. To apply the two-level preconditioner (1.6), one needs to solve a system

$$B\,\mathbf{v} \;=\; \mathbf{g} \tag{4.5}$$

at each iteration. This is performed with the following steps:

1. $\mathbf{y}_F \;=\; P_{FF}^{-1}\,\mathbf{g}_F$,
2. $\mathbf{y}_C \;=\; \mathbf{g}_C - A_{CF}\,\mathbf{y}_F$,
3. solve $S\,\mathbf{v}_C \;=\; \mathbf{y}_C$,
4. $\mathbf{v}_F \;=\; P_{FF}^{-1}\,(\mathbf{g}_F - A_{FC}\,\mathbf{v}_C)$.

A multilevel scheme is obtained when the solution to

$$S\,\mathbf{v}_C = \mathbf{y}_C \tag{4.6}$$

at step 3 is computed only approximately, using a preconditioner B_S for S that is defined with the same algorithms and rules as those used to define B. This principle is then applied recursively until the coarse grid matrix is sufficiently small so that it is more advantageous to factorize it exactly with some standard software.

The simplest approach consists of using $\mathbf{v}_C \;=\; B_S^{-1}\mathbf{y}_C$, leading to a so-called V-cycle. This is unfortunately not optimal with this kind of preconditioner. We therefore consider a more sophisticated approach, in which an approximate solution to (4.6) is computed by running a few steps of a Krylov subspace method with B_S as preconditioner. This idea has already been followed in [24] and traces back to [4]. It is important to recall that the approach is stable, in the sense that a small error in the solution to (4.6) cannot be amplified and entails a large error in the solution to (4.5). More precisely, as shown in [24], the residual in (4.5) is the residual in (4.6) padded with zero for the F entries; in the symmetric case, one has further that the energy norm of the error is the same in (4.5) and in (4.6).

Concerning the practical details, we made essentially the same choices as in [24]. We review them below, with a few comments.

- Except at the coarsest level, the so defined preconditioner is slightly variable from step to step. We therefore select Krylov subspace methods explicitly designed to accommodate variable or "inexact" preconditioning, namely the flexible CG method [15, 21] in the symmetric case and FGMRES [27, 29] in the nonsymmetric one. In all cases, the initial approximation is the zero vector.
- Inner iterations are exited when the residual $\mathbf{r}_C$ in (4.6) satisfies $\|\mathbf{r}_C\| \leq 0.35\|\mathbf{y}_C\|$, or when the number of iterations reaches its maximal value, set equal to $int[nz(A)/nz(S)]$, where $int[\cdot]$ and $nz(\cdot)$ mean the integer part of and the number of nonzero entries in, respectively. As discussed in [24], the latter criterion ensures that the global cost of one application of the preconditioner is bounded independently of the number of levels; with the choices made for the coarsening process, this maximal number of iterations is most often equal to 3.
- The coarse grid matrix is factorized exactly when the estimated cost for this exact factorization represents less than a fraction of one unpreconditioned CG iteration for the whole system. This fraction is set equal to 1 in the symmetric case and 0.2 in the nonsymmetric one (for which the real cost may be larger than the estimated one, because of the need for pivoting).
- We use the CERFACS implementation of FGMRES [14], whereas the coarse grid matrices at the coarsest level are factorized using routines of the MA27 and MA28 suites from the HSL archive [18].

5. Numerical results. We consider the following test problems.

PROBLEM 1 (model 2D anisotropic problem). *Linear system resulting from the five point finite difference approximation of*

$$-a_x \frac{\partial^2 u}{\partial x^2} - a_y \frac{\partial^2 u}{\partial y^2} \;=\; 1 \qquad in \quad \Omega = (0,1)\times(0,1)$$

with constant coefficients a_x, a_y and boundary conditions

$$\left\{ \begin{array}{rcll} u & = & 0 & on \;\; x = 1\,,\, 0 \le y \le 1, \\ \frac{\partial u}{\partial n} & = & 0 & elsewhere \; on \;\; \partial\Omega. \end{array} \right.$$

We use a uniform mesh with constant mesh size h in both directions.

PROBLEM 2 (model 2D anisotropic problem, bilinear finite elements). *Linear system resulting from the* bilinear finite element *approximation of*

$$-a_x \frac{\partial^2 u}{\partial x^2} - a_y \frac{\partial^2 u}{\partial y^2} \;=\; 1 \qquad in \quad \Omega = (0,1)\times(0,1)$$

with constant coefficients a_x, a_y and homogeneous Dirichlet boundary conditions everywhere on $\partial\Omega$. We use a uniform mesh with constant mesh size h in both directions.

PROBLEM 3 (2D problem with anisotropy and discontinuity). *Linear system resulting from the five point finite difference approximation of*

$$-\frac{\partial}{\partial x}\left(a_x \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(a_y \frac{\partial u}{\partial y}\right) \;=\; f \qquad in \quad \Omega = (0,1)\times(0,1)$$

with boundary conditions

$$\left\{ \begin{array}{rcll} u & = & 0 & on \;\; y = 1\,,\, 0 \le x \le 1, \\ \frac{\partial u}{\partial n} & = & 0 & elsewhere \; on \;\; \partial\Omega \end{array} \right.$$

and coefficients given by

$$\left\{ \begin{array}{llll} a_x = 1, & a_y = d, & f = 0 & in \;\; (0.65\,,\, 0.95)\times(0.05\,,\, 0.65), \\ a_x = d, & a_y = 1, & f = 0 & in \;\; (0.25\,,\, 0.45)\times(0.25\,,\, 0.45), \\ a_x = d, & a_y = d, & f = 1 & in \;\; (0.05\,,\, 0.25)\times(0.65\,,\, 0.95), \\ a_x = 1, & a_y = 1, & f = 0 & elsewhere, \end{array} \right.$$

where d is a parameter. We use a uniform mesh with constant mesh size h in both directions.

PROBLEM 4 (model 3D anisotropic problem). *Linear system resulting from the seven point finite difference approximation of*

$$-a_x \frac{\partial^2 u}{\partial x^2} - a_y \frac{\partial^2 u}{\partial y^2} - a_z \frac{\partial^2 u}{\partial y^2} \;=\; 1 \qquad in \quad \Omega = (0,1)\times(0,1)\times(0,1)$$

with constant coefficients a_x, a_y, a_z and boundary conditions

$$\left\{ \begin{array}{rcll} u & = & 0 & on \;\; x = 1\,,\, 0 \le y\,,\, z \le 1, \\ \frac{\partial u}{\partial n} & = & 0 & elsewhere \; on \;\; \partial\Omega. \end{array} \right.$$

We use a uniform mesh with constant mesh size h in all directions.

PROBLEM 5 (3D problem with discontinuity and local refinement). *Linear system resulting from the seven point finite difference approximation of*

$$-\frac{\partial}{\partial x}\left(a_x \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(a_y \frac{\partial u}{\partial y}\right) - \frac{\partial}{\partial z}\left(a_z \frac{\partial u}{\partial z}\right) \;=\; f \qquad in \;\; \Omega = (0,1)\times(0,1)\times(0,1)$$

with boundary conditions

$$\left\{ \begin{array}{rcll} u & = & 0 & \text{on } \; z=1\,,\, 0 \le x\,,\, y \le 1, \\ \frac{\partial u}{\partial n} & = & 0 & \text{elsewhere on } \; \partial\Omega \end{array} \right.$$

and coefficients given by

$$\left\{ \begin{array}{rcl} a_x \;=\; a_y \;= a_z \;=\; d, \quad f=1 & & \text{in } \; \left(\frac{1}{4},\frac{3}{4}\right) \times \left(\frac{1}{4},\frac{3}{4}\right) \times \left(\frac{1}{4},\frac{3}{4}\right), \\ a_x \;=\; a_y \;= a_z \;=\; 1, \quad f=0 & & \text{elsewhere}, \end{array} \right.$$

where d is a parameter. We use either a uniform mesh *with constant mesh size h in all directions or a* nonuniform mesh*, structurally regular (Cartesian) but refined near material interfaces, with mesh sizes $h_x = h(x)$, $h_y = h(y)$, $h_z = h(z)$ varying according to*

$$h(t) \;=\; \left\{ \begin{array}{ll} \frac{1}{400} & \text{if } \; 0.2 \le t \le 0.3 \;\; \text{or} \;\; 0.7 \le t \le 0.8, \\ \frac{1}{25} & \text{otherwise} \end{array} \right.$$

to define a $101 \times 101 \times 101$ grid, or half these values to define a $201 \times 201 \times 201$ grid.

PROBLEM 6 (2D convection-diffusion equation). *Linear system resulting from the five point finite difference approximation (upwind scheme) of*

$$-\nu\,\Delta\,u \;+\; \overline{v}\,\overline{\nabla} u \;=\; 0 \qquad \text{in } \; \Omega = (0,1) \times (0,1)$$

with boundary conditions

$$\left\{ \begin{array}{rcll} u & = & 1 & \text{on } \; y=1\,,\, 0 \le x \le 1, \\ u & = & 0 & \text{elsewhere on } \; \partial\Omega \end{array} \right.$$

and convective flow given by

$$\overline{v}(x\,,\,y) = \begin{pmatrix} x(1-x)(2\,y-1) \\ -(2\,x-1)y(1-y) \end{pmatrix} \;;$$

$\nu = \infty$ corresponds to the Laplace equation (no convection). We use either a uniform mesh *with constant mesh size h in both directions or a* stretched mesh*, structurally regular (Cartesian) but refined in the neighborhood of the boundaries, in such a way that the ratio of maximum mesh size to minimum mesh size is equal to* 200*, the ratio of subsequent mesh sizes being kept constant.*

PROBLEM 7 (3D convection-diffusion equation). *Linear system resulting from the seven point finite difference approximation (upwind scheme) of*

$$-\nu\,\Delta\,u \;+\; \overline{v}\,\overline{\nabla} u \;=\; 0 \qquad \text{in } \; \Omega = (0,1) \times (0,1) \times (0,1)$$

with boundary conditions

$$\left\{ \begin{array}{rcll} u & = & 1 & \text{on } \; z=1\,,\, 0 \le x\,,\, y \le 1, \\ u & = & 0 & \text{elsewhere on } \; \partial\Omega \end{array} \right.$$

and convective flow given by

$$\overline{v}(x\,,\,y\,,\,z) = \begin{pmatrix} 2\,x(1-x)(2\,y-1)z \\ -(2\,x-1)y(1-y) \\ -(2\,x-1)(2\,y-1)z(1-z) \end{pmatrix} \;;$$

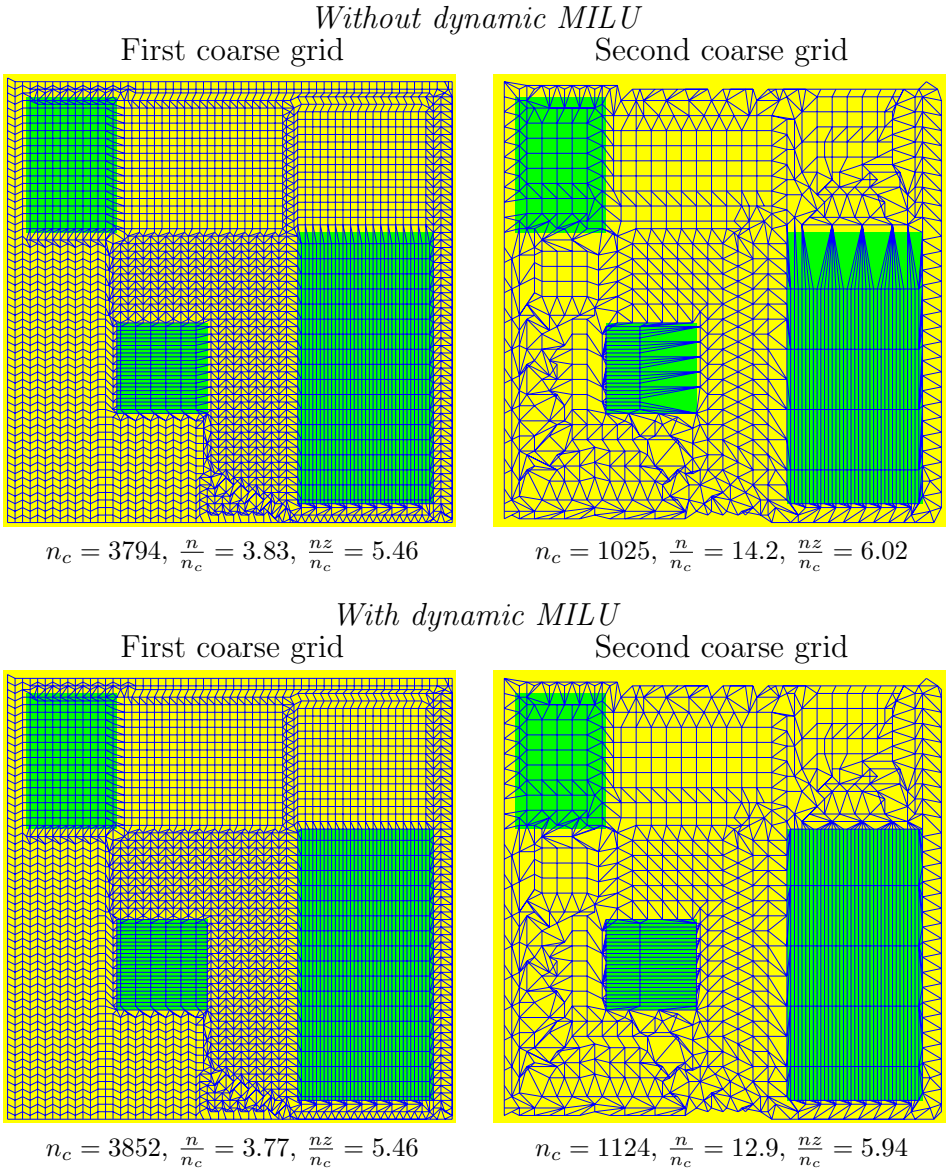

FIG. 2. *Coarsening for Problem* 3 *with* $d = 100$ *and* $h^{-1} = 120$ *(*$n = 14520$*;* $\frac{nz}{n} = 4.96$*, where* nz *stands for the number of nonzero entries).*

$\nu = \infty$ *corresponds to the Laplace equation (no convection). We use either a* uniform mesh *with constant mesh size* h *in all directions or a* stretched mesh*, structurally regular (Cartesian) but refined in the neighborhood of the boundaries, in such a way that the ratio of maximum mesh size to minimum mesh size is equal to* 200*, the ratio of subsequent mesh sizes being kept constant.*

We first illustrate the behavior of our aggregation scheme. In this respect, the most interesting symmetric problem is perhaps Problem 3. For $d = 100$ and $h^{-1} = 120$, the rules given in section 4.3 entail that there are only three levels, the second coarse grid being already sufficiently small to be factorized exactly. Figure 2 displays these first two coarse grids (displaying the fine grid would be uninteresting

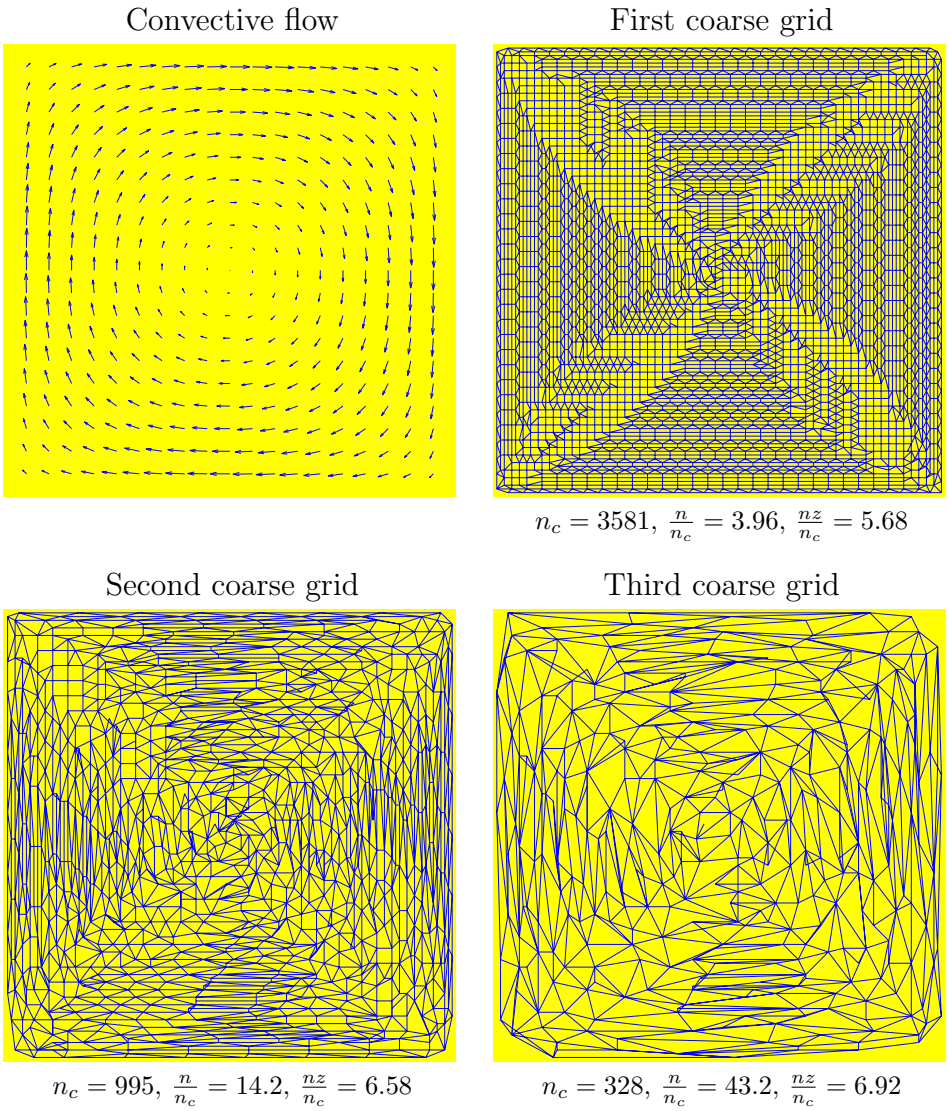

FIG. 3. *Coarsening with dynamic MILU for Problem* 6 *with* $\nu = 10^{-4}$ *and a uniform* 121×121 *grid* ($n = 14161$; $\frac{nz}{n} = 4.96$, *where nz stands for the number of nonzero entries).*

because it is uniform). On the top we illustrate our coarsening without dynamic MILU (Algorithm 2.2 alone), whereas the bottom figures correspond to the combined use of Algorithms 2.2 and 4.1. For clarity, the zones in which either $a_x \neq 1$ or $a_y \neq 1$ or both have been slightly shadowed (in green with color display). One sees the semi-coarsening $h \to 4\,h$ in the anisotropic zones, and one may check that the refinement of the F/C partitioning by dynamic MILU has only a marginal impact on the number of nodes in the coarse grids. Nevertheless, it not only guarantees the quality of the MILU factorization of A_{FF}, but seems also to improve the regularity of the coarsening. This suggests that it could be useful in other contexts as well, offering, for instance, a cheap alternative to the iterative improvement of the coarsening by so-called compatible relaxation [9, 13].

TABLE 1
Results for Problem 1.

		$h^{-1} = 600$				$h^{-1} = 1200$			
a_x	a_y	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
				Geometric multigrid					
1	1	3.99	1.00	9	28.3	4.00	1.00	9	28.5
				Aggregation-based algebraic multilevel					
1	1	3.99	1.94	18	63.3	4.00	2.00	19	70.3
1	2	3.98	1.95	19	63.0	3.99	1.95	20	68.4
1	4	3.97	2.10	20	70.6	3.98	2.10	21	76.2
1	10	3.95	2.05	22	72.1	3.98	2.10	21	70.7
1	10^2	3.95	2.00	22	69.0	3.98	2.06	18	58.9
1	10^4	3.95	1.94	18	54.9	3.98	1.89	18	55.4

TABLE 2
Results for Problem 2.

		$h^{-1} = 600$				$h^{-1} = 1200$			
a_x	a_y	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
1	1	3.99	1.94	17	55.2	3.99	1.94	17	56.4
1	2	3.99	2.20	15	54.0	4.00	2.20	15	55.6
1	4	3.99	2.11	19	68.9	4.00	2.05	20	72.2
1	10	3.99	2.00	20	68.5	4.00	2.05	20	74.2
1	10^2	3.99	1.95	20	69.1	4.00	1.75	24	80.0
1	10^4	3.99	1.75	20	65.5	4.00	1.83	18	60.3

We also illustrate the coarsening on the unsymmetric Problem 6 with $\nu = 10^{-4}$ and a uniform 121×121 grid. Here, four levels and thus three coarse grids are necessary. (As already written in section 4.3, we have to use a more strict rule in the nonsymmetric case because of the need for pivoting.) We show these coarse grids in Figure 3, displaying only those obtained with dynamic MILU because there is nothing new to learn from those obtained without this enhancement. We also graphically represent the magnitude and the direction of the convective flow $\overline{v}$. This allows us to see that semicoarsening effects appear in the regions where the flow is largest in magnitude and aligned with the grid lines.

We now assess the performances of our multilevel preconditioner. All problems were solved by the flexible CG method [15, 21] (Problems 1–5) or FGMRES [27, 29] with restart parameter set equal to 10 (Problems 6–7). The initial approximation was the zero vector, and the stopping criterion was a decrease of the relative residual error by 10^{-6}. In all cases, we used the algebraic multilevel preconditioner, as described in section 4, based on the aggregation procedure given in section 2. All parameters were set to their default value.

The results are reported in Tables 1–7. In these tables, n/n_C is the coarsening ratio at the first level, *inner* the mean number of inner iterations to (approximately) solve the coarse grid system on this first level, *iter.* the number of (outer) iterations to solve the system, and *sol.* the solution cost (number of floating point operations), relative to the cost of 1 unpreconditioned CG iteration. To have a more precise idea of this relative efficiency, we also solved the model Problem 1 with $a_x = a_y = 1$ using a standard geometric multigrid scheme accelerated by CG (bilinear interpolation, one pre- and one postdamped Jacobi smoothing step with $\omega = 0.5$, V-cycle).

TABLE 3
Results for Problem 3.

	$h^{-1} = 600$				$h^{-1} = 1200$			
d	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
1	3.99	1.76	21	67.1	4.00	1.76	21	68.8
2	3.97	2.00	20	69.8	3.99	2.00	23	82.9
4	3.96	2.04	24	84.1	3.98	2.00	25	90.3
10	3.95	2.04	24	84.2	3.98	2.04	26	94.0
10^2	3.95	2.04	24	82.5	3.98	2.00	26	90.6
10^4	3.95	1.96	26	88.0	3.98	2.04	27	95.6
10^6	3.95	2.15	26	92.2	3.98	2.00	31	107.9

TABLE 4
Results for Problem 4.

			$h^{-1} = 100$				$h^{-1} = 200$			
a_x	a_y	a_z	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
1	1	1	4.00	2.00	19	91.0	4.00	2.00	19	93.8
1	4	1	4.00	2.00	19	91.0	4.00	2.00	19	93.8
1	4	2	3.90	2.16	19	88.8	3.95	2.37	19	95.1
1	4	4	3.93	2.06	17	73.0	3.96	2.06	18	79.0
1	16	1	3.74	2.05	20	80.6	3.87	2.05	20	91.3
1	16	4	3.81	2.16	19	85.5	3.90	2.26	19	93.7
1	16	16	3.92	2.13	16	72.6	3.96	2.18	17	78.7
1	10^2	1	3.74	2.05	19	74.2	3.87	1.78	27	110.2
1	10^2	10	3.74	2.11	19	87.3	3.87	2.10	21	88.2
1	10^2	10^2	3.92	2.12	17	74.0	3.96	2.19	16	72.8
1	10^4	1	3.74	2.00	15	59.0	3.87	2.29	14	62.2
1	10^4	10^2	3.74	2.06	17	69.2	3.87	1.95	21	89.6
1	10^4	10^4	3.92	2.06	17	72.1	3.96	2.06	17	74.2

TABLE 5
Results for Problem 5.

	$101 \times 101 \times 101$ grid				$201 \times 201 \times 201$ grid			
d	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
				Uniform mesh				
1	4.00	2.05	19	85.0	4.00	2.10	20	94.4
4	4.00	2.05	21	96.8	4.00	2.00	22	105.8
10	4.00	2.05	21	97.1	4.00	2.05	21	104.5
10^2	4.00	2.05	22	104.8	4.00	2.05	22	108.3
10^4	4.00	2.09	22	104.9	4.00	2.05	22	109.3
10^6	4.00	2.09	22	106.5	4.00	2.04	23	113.7
				Nonuniform mesh				
1	3.93	2.05	22	100.3	3.97	2.04	24	105.2
2	3.89	2.04	23	107.4	3.95	1.96	26	115.0
4	3.90	2.05	22	103.1	3.95	2.04	24	111.4
10	3.89	2.05	22	103.8	3.95	2.04	24	111.7
10^2	3.89	2.04	23	106.0	3.95	2.04	24	110.4
10^4	3.89	2.04	23	108.1	3.95	2.08	25	118.4
10^6	3.89	2.04	23	108.3	3.95	2.13	24	112.9

TABLE 6
Results for Problem 6.

	601×601 grid				1201×1201 grid			
ν	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
				Uniform mesh				
∞	3.99	1.85	13	61.7	4.00	1.85	13	61.9
1	3.96	2.00	15	71.6	3.85	2.00	17	87.0
10^{-1}	3.99	2.00	15	69.8	3.99	1.87	15	67.1
10^{-2}	3.98	2.00	17	87.4	3.96	2.00	18	89.0
10^{-4}	3.94	2.81	21	149.7	3.93	2.90	21	176.3
10^{-6}	3.99	2.83	23	158.3	4.00	2.87	23	178.0
				Stretched mesh				
∞	3.98	1.89	18	78.6	3.98	1.89	18	79.9
1	3.98	1.88	16	68.4	3.98	1.89	18	79.9
10^{-1}	3.98	1.94	17	74.0	3.98	1.89	18	79.8
10^{-2}	3.97	2.00	17	76.2	3.99	1.89	18	80.4
10^{-4}	3.79	2.40	25	130.0	3.83	2.71	24	138.4
10^{-6}	3.41	1.97	29	157.8	3.61	2.85	27	180.8

TABLE 7
Results for Problem 7.

	$101 \times 101 \times 101$ grid				$201 \times 201 \times 201$ grid			
ν	$\frac{n}{n_c}$	inner	iter.	sol.	$\frac{n}{n_c}$	inner	iter.	sol.
				Uniform mesh				
∞	4.00	2.00	15	79.6	4.00	2.00	15	82.0
1	3.76	2.00	17	109.7	3.81	2.00	17	108.3
10^{-1}	3.80	2.00	16	99.4	3.81	2.00	17	111.2
10^{-2}	3.75	2.00	18	119.7	3.84	1.94	18	111.7
10^{-4}	3.93	2.00	21	136.9	3.93	2.00	21	136.8
10^{-6}	3.93	2.00	26	167.2	3.96	2.00	30	203.0
				Stretched mesh				
∞	3.91	1.94	16	79.3	3.94	1.88	17	85.8
1	3.91	1.94	16	80.0	3.94	1.88	17	85.8
10^{-1}	3.91	2.00	16	79.2	3.94	1.88	17	85.7
10^{-2}	3.92	1.65	20	95.5	3.95	1.94	17	85.3
10^{-4}	3.46	1.81	21	117.1	3.64	1.87	23	125.8
10^{-6}	3.64	2.00	27	171.0	3.28	2.00	27	186.3

One sees that our method is about 2.5 times slower than a well-tuned geometric multigrid method on a model 2D problem. This is, however, compensated by a great robustness. The different test problems feature several difficulties: strong anisotropies, sometimes with varying direction; large jumps in the PDE coefficients; anisotropy induced by nonuniform grids; large positive off-diagonal entries induced by the bilinear finite element approximation of an anisotropic problem (and the associated difficulty in detecting the right direction of strong coupling); highly nonsymmetric matrices without any regular pattern because they originate from a convective flow that is varying both in magnitude and direction. Despite this, the tables look rather similar, with about the same coarsening ratios and numbers of inner or outer iterations, whereas, except for the convection-diffusion problems with extremely low viscosity,

the relative solution cost is never larger than 1.7 times that for the model 2D problem.

Acknowledgments. I thank the referees for their careful reading and helpful suggestions.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.
[2] O. Axelsson and M. Neytcheva, *Algebraic multilevel iterations for Stieltjes matrices*, Numer. Linear Algebra Appl., 1 (1994), pp. 213–236.
[3] O. Axelsson and P. Vassilevski, *Algebraic multilevel preconditioning methods,* II, SIAM J. Numer. Anal., 27 (1990), pp. 1569–1590.
[4] O. Axelsson and P. S. Vassilevski, *Variable-step multilevel preconditioning methods.* I. *Self-adjoint and positive definite elliptic problems*, Numer. Linear Algebra Appl., 1 (1994), pp. 75–101.
[5] R. E. Bank and R. K. Smith, *An algebraic multilevel multigraph algorithm*, SIAM J. Sci. Comput., 23 (2002), pp. 1572–1592.
[6] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.
[7] E. F. F. Botta and F. W. Wubs, *Matrix renumbering ILU: An effective algebraic multilevel ILU preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1007–1026.
[8] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393.
[9] A. Brandt, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.
[10] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.
[11] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., 25 (2003), pp. 1–26.
[12] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1886–1908.
[13] R. D. Falgout and P. S. Vassilevski, *On generalizing the algebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693.
[14] V. Frayssé, L. Giraud, and S. Gratton, *Algorithm* 842: *A set of GMRES routines for real and complex arithmetics on high performance computers*, ACM Trans. Math. Software, 31 (2005), pp. 228–238.
[15] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iterations*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.
[16] W. Hackbusch, *Multi-grid Methods and Applications*, Springer-Verlag, Berlin, 1985.
[17] V. E. Henson and P. S. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, SIAM J. Sci. Comput., 23 (2001), pp. 629–650.
[18] *HSL archive*, available via http://hsl.rl.ac.uk/archive/hslarchive.
[19] J. E. Jones and P. S. Vassilevski, *AMGE based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.
[20] Y. Notay, *Using approximate inverses in algebraic multilevel methods*, Numer. Math., 80 (1998), pp. 397–417.
[21] Y. Notay, *Flexible conjugate gradients*, SIAM J. Sci. Comput., 22 (2000), pp. 1444–1460.
[22] Y. Notay, *Optimal order preconditioning of finite difference matrices*, SIAM J. Sci. Comput., 21 (2000), pp. 1991–2007.
[23] Y. Notay, *A robust algebraic multilevel preconditioner for non-symmetric M-matrices*, Numer. Linear Algebra Appl., 7 (2000), pp. 243–267.
[24] Y. Notay, *Robust parameter free algebraic multilevel preconditioning*, Numer. Linear Algebra Appl., 9 (2002), pp. 409–428.
[25] Y. Notay, *Algebraic multigrid and algebraic multilevel methods: A theoretical comparison*, Numer. Linear Algebra Appl., 12 (2005), pp. 419–451.
[26] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers in Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[27] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[28] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[30] 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.
[31] Y. Saad and J. Zhang, *BILUM: Block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 20 (1999), pp. 2103–2121.
[32] K. Stüben, *An introduction to algebraic multigrid*, in Multigrid, U. Trottenberg, C. W. Oosterlee, and A. Schüller, Academic Press, London, 2001, Appendix A, pp. 413–532.
[33] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[34] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.