

Robust parameter-free algebraic multilevel preconditioning

Y. Notay^{*,†}

*Service de Métrologie Nucléaire, Université Libre de Bruxelles (C.P. 165/84), 50, Av. F.D. Roosevelt,
B-1050 Brussels, Belgium*

SUMMARY

To precondition large sparse linear systems resulting from the discretization of second-order elliptic partial differential equations, many recent works focus on the so-called algebraic multilevel methods. These are based on a block incomplete factorization process applied to the system matrix partitioned in hierarchical form. They have been shown to be both robust and efficient in several circumstances, leading to iterative solution schemes of optimal order of computational complexity. Now, despite the procedure is essentially algebraic, previous works focus generally on a specific context and consider schemes that use classical grid hierarchies with characteristic mesh sizes $h, 2h, 4h$, etc. Therefore, these methods require some extra information besides the matrix of the linear system and lack of robustness in some situations where semi-coarsening would be desirable. In this paper, we develop a general method that can be applied in a black box fashion to a wide class of problems, ranging from 2D model Poisson problems to 3D singularly perturbed convection–diffusion equations. It is based on an automatic coarsening process similar to the one used in the AMG method, and on coarse grid matrices computed according to a simple and cheap aggregation principle. Numerical experiments illustrate the efficiency and the robustness of the proposed approach. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: iterative methods; convergence; preconditioning

1. INTRODUCTION

We consider the solution of large sparse $n \times n$ linear systems

$$A\mathbf{u} = \mathbf{b} \tag{1}$$

arising from the discretization of second-order elliptic partial differential equations (PDEs). In this context, it is generally recognized that iterative methods outperform direct ones if one can use a preconditioning technique that is both robust and efficient.

Multigrid methods are now well known (cf. References [1–3] and the references therein). They exploit a multilevel structure corresponding to a hierarchy of discretizations, on which

* Correspondence to: Y. Notay, Service de Métrologie Nucléaire, Université Libre de Bruxelles (C.P. 165/84), 50, Av. F.D. Roosevelt, B-1050 Brussels, Belgium.

† E-mail: ynotay@ulb.ac.be

local corrections are computed that are combined on the finest grid by means of appropriate prolongation operators. Some such methods have been developed which require only the matrix of the linear system (e.g. References [4–10]). These are called *algebraic* multigrid and try to mimic standard schemes by computing automatically, with appropriate algorithms, the needed hierarchy of grids and discretizations.

Other purely algebraic preconditioners are ILU methods (References [11, 12] and the references therein). These are easier to implement but often less efficient than multigrid for discretized PDEs. This observation motivated ILU-type methods that exploit a multilevel structure through a renumbering of the unknowns [13–19], as well as hybrid versions that also incorporate some multigrid ideas [20, 21].

Somewhere in between these approaches, *algebraic multilevel* preconditioning methods exploit a hierarchy of grids and discretizations (like multigrid), but incorporate it in a block incomplete factorization framework. Despite the procedure is essentially algebraic (see next section for details), previous works (e.g. References [22–39]) all focus on a rather specific context, mostly systems arising from self-adjoint two-dimensional (2D) PDEs, with few results on unsymmetric systems and little attention to three-dimensional (3D) problems. Moreover, all schemes considered so far use classical grid hierarchies with characteristic mesh sizes h , $2h$, $4h$, etc. Therefore, they require some extra information besides the matrix of the linear system.

In our eyes, these limitations come from the focus of these previous studies, which is essentially theoretical and requires therefore a well-defined framework. To some extent, this theoretical stress is welcome, as it gives confidence that the approach is not only efficient, but is also prone to robustness, with many results proving its insensitivity to jumps or anisotropy in the PDE coefficients, and even some analysis showing a potential robustness for highly non-symmetric systems [35].

Nevertheless, it is lacking a general method, that is perhaps less sharpened from a theoretical point of view, but that can be applied in a black box fashion to a wide class of problems, ranging from 2D model Poisson problems to 3D singularly perturbed convection–diffusion equations.

In the present paper, we aim at filling this gap. The resulting method is based on a coarsening process similar to the one used in the AMG method [8, 9], the successive coarse grid matrices being computed according to the simple and cheap aggregation principle from [5]. Besides the use of these ingredients our method has however little in common with algebraic *multigrid* techniques as developed in these works or in References [4, 6, 7, 10]. Indeed, our method does not try to follow the classical multigrid paradigm and does not require the definition of a smoother and a prolongation (or interpolation) operator. Instead, the coarsening process is used to define a block partitioning of the matrix A , which is then preconditioned by means of a block incomplete factorization process in which the above-mentioned coarse grid matrices serve as approximate Schur complement.

On the other hand, our method differs significantly from previous algebraic multilevel block ILU schemes such as proposed in References [13, 14, 16–21]. Indeed, without entering the details of the many proposed variants, these methods have in common that they try to follow more closely the classical ILU paradigm. Hence, the partitioning is first and foremost governed by the need to have block pivots that are easily invertible, e.g. diagonal or close to diagonal. In general, this makes the coarsening significantly slower than in our approach, and generate approximate Schur complements that are much denser.

The presentation is organized as follows. The general framework of algebraic multilevel preconditioning is sketched in Section 2. Our specific choices for the different components are discussed in Sections 3–6 and 7 is devoted to numerical results.

2. GENERAL FRAMEWORK

Algebraic multilevel preconditioning is based on the recursive use of a two-level procedure which starts with the partitioning of the unknowns in fine and coarse grid ones. The matrix is then permuted (in practice implicitly) in the 2×2 block form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad (2)$$

where first block of unknowns correspond to the fine grid nodes and the second block of unknowns to the coarse grid nodes.

Noting $S_A = A_{22} - A_{21}A_{11}^{-1}A_{12}$ the Schur complement of A , the exact block factorization of A

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & S_A \end{pmatrix} \begin{pmatrix} I & A_{11}^{-1}A_{12} \\ & I \end{pmatrix}$$

is approximated with

$$B = \begin{pmatrix} P & \\ A_{21} & S \end{pmatrix} \begin{pmatrix} I & P^{-1}A_{12} \\ & I \end{pmatrix} \quad (3)$$

where P and S are relevant approximations to A_{11} and S_A , respectively.

Previous works were successful by selecting approximate Schur complements S that are (closely related to) discretization matrices on the coarse grid (i.e. the grid induced by the nodes in the second block). Since applying preconditioner (3) requires solving a system with S , multilevel schemes enter the scene with the observation that one level of reduction is generally not enough to make an exact factorization of S affordable. One thus uses the same procedure to define a preconditioner B_S for S , that serves as building block to set up an approximation to S^{-1} . This process is then repeated recursively until the size of S is small enough to allow an exact factorization at a reasonable cost.

The key ingredients of algebraic multilevel preconditioning are therefore:

- a coarsening process, i.e. a strategy according to which the unknowns are partitioned in fine and coarse grid ones;
- a technique to derive relevant approximations P to A_{11} ;
- a technique to derive relevant approximations S to S_A ;
- a method defining the needed approximate inverse of S in function of B_S^{-1} .

Our choice for these four ingredients are discussed (in that order) in the next four sections.

3. COARSENING

Let us first briefly discuss the desirable properties of the fine/coarse partitioning. In accordance with previous theoretical and numerical results, the success of the approach requires first and foremost that P and S are excellent approximations to A_{11} and S_A . When using regular ‘geometric’ (say $h-2h$) coarsening on standard problems, one generally starts with the observation that A_{11} is well-conditioned (see References [40–42] for an analysis). Hence, any reasonable preconditioning technique results in a good approximation.

Now, this good conditioning is actually a mere consequence of the coarsening process: if A is weakly diagonally dominant, each row of A_{11} becomes strongly diagonally dominant when the corresponding row of A_{12} has at least one non-zero entry. From there, one discovers that this good conditioning is not ensured in all cases. It depends on the relative value of the entries in A_{12} , i.e. on the relative strength of the coupling between fine and coarse grid unknowns. Hence, A_{11} will, for instance, tend to be badly conditioned when the underlying PDE has strongly anisotropic coefficients. For 2D problems, this may be compensated by defining P with an ILU-type method that is increasingly accurate as anisotropy becomes stronger (see e.g. Reference [34]). Unfortunately, there is little hope to get rid of all situations in 3D examples, except by using problem dependant approaches like semi-coarsening or block ILU approximations to A_{11} with planewise block partitioning.

We therefore conclude that a robust general purpose scheme should be based on an adaptive coarsening strategy for which the good conditioning of A_{11} is *ensured*.

At this stage, it is interesting to look at the coarsening process in the AMG method from References [8, 9]. Indeed, it has been designed with in mind classical multigrid algorithms, but the latter require semi-coarsening in essentially the same situations as algebraic multilevel preconditioning. Hence, this algorithm is a good starting point for us, many examples showing that it successfully mimics geometric coarsening with semi-coarsening effects whenever desirable. Even better, the main step of the algorithm selects one unmarked node i at a time to be the next coarse grid node, and marks as fine grid node all its neighbors j for which the connection a_{ji} is strong enough. Clearly, if A is weakly diagonally dominant (see below for a discussion of the general case), the latter requirement ensures that each row of A_{11} will be strongly diagonally dominant, with a level of diagonal dominance high enough to ensure the good conditioning of A_{11} .

We therefore adopted this algorithm, with only a slight modification of the priority rule for the selection of the next coarse grid node, the one we propose being somewhat simpler to implement and giving on the whole slightly better results than the one considered in Reference [9, p. 475]. A detailed description is given below (Algorithm 3.1).

Let us now briefly discuss the case of matrices that are not weakly diagonally dominant. In the discrete PDE context, these are essentially matrices whose row-sum is zero almost everywhere, but are not M -matrices because part of the offdiagonal entries are positive. Clearly, there is no more guarantee that A_{11} is well-conditioned. We therefore suggest to perform an *a posteriori* check, moving from F to C the nodes for which the corresponding row of A_{11} is not diagonally dominant enough. We further suggest to proceed the nodes sequentially, so that each removal from F increases the diagonal dominance in the neighbouring rows of A_{11} . Then, at the very worst, removal is enforced until A_{11} is diagonal, i.e. until F is an independent set.

Remark

Algorithm 3.1 may be not very satisfactory whenever applied to a very sparse matrix, for instance, a matrix corresponding to a 2D five-point stencil or a 3D seven-point stencil. Indeed, it tends to produce a red–black partitioning and thus does not mimic very well geometric coarsening in such cases. In Reference [9], it is then suggested to consider the so-called *aggressive* coarsening. In our context, we found it more elegant to proceed as follows. If the mean number of non-zero entries per row in A is below some given threshold (e.g. (7)), then Algorithm 3.1 is exchanged for an algorithm that selects F to form a maximal independent set (see Reference [16] for examples and discussion). Since this makes A_{11} diagonal, we set $P=A_{11}$ and compute S equal to the exact Schur complement $S_A=A_{22}-A_{21}A_{11}^{-1}A_{12}$. On the one hand, this makes the factorization (3) exact at the first level. On the other hand, S computed in this way will be somewhat denser than A , allowing to hope a reasonable coarsening from the subsequent application of Algorithm 3.1. Note that when a red–black ordering exists, this amounts to apply the multilevel preconditioner to the reduced system resulting from the elimination of the red nodes.

Algorithm 3.1 (Fine/coarse partitioning)• *Initialization*

$$F = \emptyset, \quad C = \emptyset, \quad U = \{1, \dots, n\}$$

• *Definitions*

$$N_i = \{j \neq i \mid a_{ij} \neq 0\}$$

$$S_i = \left\{ j \neq i \mid a_{ij} < -\frac{1}{4} \max_{k \in N_i} |a_{ik}| \right\}$$

$$S_i^T = \left\{ j \neq i \mid a_{ji} < -\frac{1}{4} \max_{k \in N_j} |a_{jk}| \right\}$$

$$p_i = 4|S_i^T \cap F| + 2|S_i \cap F| + |N_i \cap F|$$

• *Main loop*

Do while $U \neq \emptyset$:

 Select $i \in U$ with maximal p_i

$C = C \cup \{i\}$, $U = U \setminus \{i\}$

 For all $j \in S_i^T \cap U$:

$F = F \cup \{j\}$, $U = U \setminus \{j\}$

 Update p_k for $k \in N_j$

 End For

End Do while

4. APPROXIMATION OF A_{11}

With the partitioning strategy chosen in the preceding section, finding a good preconditioner for A_{11} is easy. However, care is needed because not any good preconditioner for A_{11} will

work well in the context of a block approximate factorization of form (3). According to the analysis in Reference [32], one has, in particular, to pay attention that quotients of the form

$$\frac{\mathbf{v}_2^t A_{21} (I - P^{-1} A_{11})^2 A_{11}^{-1} A_{12} \mathbf{v}_2}{\mathbf{v}_2^t S_A \mathbf{v}_2}$$

are reasonably bounded for any vector \mathbf{v}_2 defined on the coarse grid nodes. Since, in the discrete elliptic PDE context, $\mathbf{v}_2^t S_A \mathbf{v}_2$ is typically small (say $\mathcal{O}(h^2)$) for ‘smooth’ vectors, it does not suffice that the spectral radius of $I - P^{-1} A_{11}$ is nicely bounded (say independently of h). Loosely speaking, P has to be a very accurate approximation to A_{11} for vectors $A_{12} \mathbf{v}_2$ with \mathbf{v}_2 smooth.

Fortunately, the analysis in Reference [32], besides warning against naive choices for P , also shows that problems tend to be prevented when using preconditioners that preserve the row-sum of A_{11} (i.e. that are exact for the constant vector). For instance, if A is a weakly diagonally dominant symmetric M -matrix and P a MILU factorization of A_{11} , then, for B defined by (3), it is proven in Reference [32] that

$$\kappa(B^{-1}A) \leq 4\kappa(P^{-1}A_{11})\kappa(S^{-1}S_A) \quad (4)$$

where $\kappa(C)$ is the spectral condition number of C , i.e. the ratio of extremal eigenvalues. Note that this result is purely algebraic and holds independently of the fine/coarse partitioning and whatever S may be.

Since, on the other hand, MILU(0) factorizations were found effective in practice for both symmetric [34] and (highly) non-symmetric [35] systems, we decided to rest on this choice in the present context. This is consistent with the coarsening procedure since MILU factorizations are well defined for any strongly diagonally dominant matrix [11].

5. APPROXIMATION OF THE SCHUR COMPLEMENT

Here, we have in some sense more freedom than we had in the preceding section since all available analyses show that the quality of preconditioner (3) depends of course of the quality of the used approximation for S_A , but is otherwise independent of the type of approximation used. This may further be checked by letting

$$\tilde{B} = \begin{pmatrix} P & \\ A_{21} & S_A \end{pmatrix} \begin{pmatrix} I & P^{-1}A_{12} \\ & I \end{pmatrix}$$

be the block factorization (3) with S equal to S_A . One has

$$B^{-1}A = (B^{-1}\tilde{B})(\tilde{B}^{-1}A)$$

with

$$\begin{aligned} B^{-1}\tilde{B} &= \begin{pmatrix} I & -P^{-1}A_{12} \\ & I \end{pmatrix} \begin{pmatrix} I & \\ & S^{-1}S_A \end{pmatrix} \begin{pmatrix} I & P^{-1}A_{12} \\ & I \end{pmatrix} \\ &= \begin{pmatrix} I & * \\ & S^{-1}S_A \end{pmatrix} \end{aligned}$$

Hence, in the symmetric positive definite (SPD) case,

$$\kappa(B^{-1}A) \leq \kappa(S^{-1}S_A)\kappa(\tilde{B}^{-1}A)$$

which shows that if the method works well with the exact Schur complement, it will also work well with any good approximation of it.

Now, two approaches have been followed so far with algebraic multilevel preconditioning. The one in References [22–25, 34–36] computes an approximate Schur complement algebraically by means of relatively simple incomplete elimination principles. It is efficient for M -matrices but, as for any ILU-type scheme, its robustness outside that specific field is questionable. For instance, there is no simple mean to ensure the positive definiteness of S whenever A and therefore S_A are positive definite. Note for completeness that the method from References [34, 35] is otherwise simple to implement and compatible with Algorithm 3.1, since S is just set equal to the Schur complement of the matrix in which A_{11} has been exchanged for a diagonal approximation; in general, we observed that this association worked well for M -matrices with, in rare occasions, an excessive increase of the number of non-zero entries in the successive approximate Schur complements.

We therefore searched for a method that could be more general while using still sparser approximations.

Unfortunately, the other approach commonly used with algebraic multilevel preconditioning tends to use somewhat denser matrices. It is based on hierarchical finite elements and use the so-called *Galerkin* approximations

$$\begin{aligned} S &= \begin{pmatrix} J_{12}^T & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} J_{12} \\ I \end{pmatrix} \\ &= A_{22} + J_{12}^T A_{12} + A_{21} J_{12} + J_{12}^T A_{11} J_{12} \end{aligned} \quad (5)$$

where J_{12} is a matrix that allows to interpolate on the fine grid a vector defined on the coarse grid nodes. This choice is essentially motivated by the possibility to estimate then (in the SPD case) the condition number by means of an analysis of the so-called ‘strengthened CBS constant’ [43].

Algebraic multigrid methods [5–10] use the same type of approximations as coarse grid matrices, and encounter a similar pitfall: one has to use the best possible interpolation, but better J_{12} means denser J_{12} and therefore increasing complexity in S . Note that the tradeoff is particularly tricky whenever using an automatic coarsening process whose behaviour is by nature not completely predictable.

Now, the interpolation matrix plays an important role in multigrid methods first and foremost because it enters the definition of the prolongation and restriction operators. These are essential components of the algorithm, and most of the demand for an accurate interpolation seems to come from the need to have relevant prolongation and restriction, and much less from a demand originating from the coarse grid matrix itself. Hence, simpler choices should be possible in the context of algebraic multilevel preconditioning, since the interpolation is used here only to define S , and plays otherwise no role in the algorithm.

This led us to consider the simplest possible scheme, corresponding to the so-called *aggregation* [5, 10]. Formally, J_{12} is then a Boolean matrix with exactly one non-zero entry

per row, connecting the concerned fine grid node to the coarse grid node with which it is ‘aggregated’. Clearly, this is a very crude ‘interpolation’ that should not be used as is in multigrid algorithms Reference [9, p. 524]. But regarding the quality of the coarse grid matrix only, we found that the impact was much more limited. For instance, on model constant coefficients problems with regular $h-2h$ coarsening, this results in a matrix very similar to a coarse discretization, up to a scaling factor.

From a theoretical point of view, note that the possible positive definiteness of A is automatically transferred to approximate Schur complements of form (5) whatever J_{12} may be. On the other hand, with the simple aggregation strategy, one necessarily loses the benefit of theoretical analyses based on the strengthened CBS inequality, but their generalization to an automatic coarsening context seems anyway out of reach.

We conclude this section with two technical remarks.

Firstly, if we use formally the same scheme as in Reference [5] to generate the coarse grid matrices, we need a different rule to form the aggregates. Indeed, in Reference [5], these are directly defined during the coarsening process, without specifying which nodes are fine and which nodes are coarse. In the present context, such a specification is required for the proper definition of the block incomplete factorization (3), and we already chose Algorithm 3.1 to this purpose. We thus form the aggregates by defining the matrix J_{12} , that is by selecting for each fine grid node j a unique coarse grid node with which it is aggregated. How do we proceed? In the context defined by Algorithm 3.1, we find it logical to select the node $k \in C$ for which a_{jk} is minimal (i.e. is the most strongly negative). Numerical experiments however show that it is nevertheless worthwhile to give some priority to the node i that initially motivated the addition of j to F (because j was in $S_i^T \cap U$ while i was processed as ‘new coarse grid node’). Thus, we keep the latter node if $a_{ji} \leq 0.99 \min_{k \in C} a_{jk}$ and otherwise select any node $k \in C$ for which a_{jk} reaches its minimal value.

Secondly, we already mentioned above that in ‘model’ situations the aggregation produces ‘right’ coarse grid matrices, but up to a scaling factors. Actually, the summation process tends to overweight somewhat the coefficients in S , compared with more standard approaches. We therefore always multiply the resulting matrix by n_c/n , where n is the number of unknowns at the considered level, and n_c the number of them that have been marked as coarse.

6. FROM TWO- TO MULTILEVEL

Up to now we have essentially discussed the two-level scheme (3). As stated in Section 2, *multilevel* schemes are based on the recursive use of the same technique to define preconditioners B_S for the successive coarse grid matrices S . The standard approach in algebraic multilevel preconditioning set then up an approximate inverse of S with

$$M = p_m(B_S^{-1}S)S^{-1}$$

where p_m is a shifted Chebyshev polynomial of degree m such that $p_m(0)=0$. This allows to exchange formulation (3) for

$$B = \begin{pmatrix} P & \\ & M^{-1} \end{pmatrix} \begin{pmatrix} I & P^{-1}A_{12} \\ & I \end{pmatrix} \quad (6)$$

whose application does no more require multiplication by the inverse of S but only multiplication by M , which is implemented with m applications of B_S and $m - 1$ multiplications by S .

In the SPD case, condition number estimates independent of the number of levels are proved if

$$\kappa(B_S^{-1}S) < m^2$$

holds at every level (see References [22, 27]). The approach has thus strong theoretical foundations. Unfortunately, the polynomials need to be defined in function of upper and lower bounds on the spectrum of $B_S^{-1}S$. Hence, not only the analysis of the method, but also its proper use require accurate theoretical results. This makes its extension to the non-symmetric case very difficult and is clearly incompatible with the goals pursued in this paper.

We therefore followed another approach, already considered in Reference [29]. It is based on the observation that the above technique does not really avoid solving systems of the form

$$S\mathbf{y}_2 = \mathbf{z}_2 \tag{7}$$

that arise when applying preconditioner (3). Simply, the exact solution is exchanged for the approximate solution obtained with m steps of the preconditioned Chebyshev iterative method. A very natural idea is then to exchange this for a few steps of a parameter-free iterative method. This is expected to have a positive impact on robustness since in usual circumstances optimal parameter-free iterative methods like GMRES or the conjugate gradient (CG) method converge at least as fast as the best polynomial iterative method [11, 12].

The present context is however somewhat particular because the use of such a method at some level makes the preconditioner seen at every higher level slightly variable from step to step. Fortunately, both GMRES and CG possess *flexible* variants that are designed to accommodate variable or ‘inexact’ preconditioning [44–48]. Since the theoretical analyses of these methods is not yet completely satisfactory,[‡] it is difficult to assess rigorously the impact of their use and state precisely the conditions under which the convergence remains independent of the number of levels. We expect some loss of efficiency on problems for which well tuned Chebyshev polynomials could be used. Indeed parameter free methods, for the same number m of iterations, require extra vector operations and one more multiplication by S .[§] The present study is however more concerned with robustness and, from that point of view, we expect that the situation is actually improved with the use of self-adaptive methods for the inner iterations.

Now, before assessing this claim through numerical experiments, it is important to check an assumption that is implicitly used in the above discussion. Indeed, with the inner iterative process, one controls the error introduced in the solution of approximate Schur complement systems (7), but what is important at the higher level is the global error made in the solution to

$$B\mathbf{v} = \mathbf{g} \tag{8}$$

[‡]There is no bound that reflects well their optimal convergence properties as the variations in the preconditioner tend to zero.

[§]The latter overhead is usually negligible but relatively more important here because m is typically very small.

where B is the two-level preconditioner (3). Thus, we have to check that a small error in (7) cannot be amplified and produce a large error in (8).

In this view, let us write

$$B = LDU$$

with

$$L = \begin{pmatrix} I & \\ A_{21}P^{-1} & I \end{pmatrix}, \quad D = \begin{pmatrix} P & \\ & S \end{pmatrix}, \quad U = \begin{pmatrix} I & P^{-1}A_{12} \\ & I \end{pmatrix}$$

Equation (8) is solved in three steps:

- (1) $\mathbf{z} = L^{-1}\mathbf{g}$;
- (2) $\mathbf{y} = D^{-1}\mathbf{z}$;
- (3) $\mathbf{v} = U^{-1}\mathbf{y}$.

Consider now that S is not inverted exactly. It means that instead of \mathbf{y} the quantity computed at step 2 is

$$\tilde{\mathbf{y}} = D^{-1} \left(\mathbf{z} - \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix} \right)$$

where \mathbf{r}_2 is the residual error in the solution to (7). Since

$$L \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix} = L^{-1} \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix}$$

one has

$$\tilde{\mathbf{y}} = D^{-1}L^{-1} \left(\mathbf{g} - \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix} \right)$$

and the computed solution to (8) is

$$\tilde{\mathbf{v}} = B^{-1} \left(\mathbf{g} - \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix} \right)$$

This means that the residual in (8) is

$$\mathbf{r} = \begin{pmatrix} 0 \\ \mathbf{r}_2 \end{pmatrix}$$

i.e. it is obtained by padding with zeros the initial residual error.

This is perfect from a practical point of view since the control of the error is precisely based on the residual norm (see below). From a theoretical point of view, in the SPD case, analyses of flexible variants of CG (FCG) are rather based on the error measured in energy norm [44–46]. Here, we have (with $L = U^T$)

$$(\mathbf{r}, B^{-1}\mathbf{r}) = (L^{-1}\mathbf{r}, D^{-1}L^{-1}\mathbf{r}) = (\mathbf{r}_2, S^{-1}\mathbf{r}_2)$$

showing that the energy norm of the error in (8) is just equal to the energy norm of the error in (7).

We conclude this section with the discussion of some practical issues.

Firstly, one needs some stopping criterion for the inner iterations. As usual, one most easily controls the relative residual error and, after some numerical testing, we fixed the threshold to 0.35. Besides, it is also necessary to fix a limit on the number of inner iterations. On the one hand, in the non-symmetric case, this allows to use full GMRES (with no restarting) while limiting the size of the needed workspace. On the other hand, one has to have a safeguard against situations where, because the method does not work as expected, the cost required for the application of the preconditioner becomes excessive without real possibility of control by the end user.

To fix the notation, consider that the actual fine grid corresponds to level 1, the next one to level 2, etc. and let S_k be the system matrix for inner iterations at that level k (S_1 is thus the approximate Schur complement generated from A , S_2 the one generated from S_1 , etc.). To implement the needed safeguard, we set the maximal number of inner iterations equal to the integer part of $nz(A)/nz(S_1)$ at level 1 and $nz(S_k)/nz(S_{k-1})$ at levels $k > 1$. Indeed, let then m_k be the mean number of inner iterations at level k , and $c_k nz(S_k)$ be the mean cost of one inner iteration *excluding* the cost of needed inner iterations at subsequent levels (i.e. the factor c_k takes into account multiplications by S_k , vector operations and operations with the matrices P , A_{12} and A_{21} computed from S_k , but not those associated with S_{k+1}). The overall cost associated with the multilevel preconditioner is given by

$$\text{Cost} = c_0 nz(A) + m_1(c_1 nz(S_1) + m_2(c_2 nz(S_2) + m_3(c_3 nz(S_3) + \dots)))$$

and, since our secondary stopping test implies

$$m_1 nz(S_1) < nz(A), \quad m_1 m_2 nz(S_2) < nz(A), \dots$$

one has

$$\text{Cost} < \left(\sum_{k=0}^{\ell} c_k \right) nz(A)$$

where ℓ is the index of the last level on which inner iterations are performed ($S_{\ell+1}$ being factorized exactly). At the very worst, the cost increases thus linearly with the number of levels. Note that one should normally remain below this value. If the number of inner iterations systematically reaches its upper limit without satisfying the convergence criterion, it probably means that the preconditioning method does not behave satisfactory, and the convergence of the outer iterations is likely to be slower than expected.

Note that the upper bound on the number of inner iterations may occasionally be equal to one. This happens when Algorithm 3.1 produces severe semi-coarsening effects, so that the number of non-zero entries in S decreases only slowly despite our aggregation strategy. Then, we skip inner iterations and use just one application of the preconditioner instead. Observe that this does not spoil the method if this relatively large number of non-zero entries in S goes with an increasing quality of the basic two-level preconditioner, relaxing the need for inner iterations.

Finally, we give a procedure (Algorithm 6.1 below) that implements the action of the preconditioner when the method used for the inner iterations is the FCG(1) method from Reference [46], that is a flexible variant of CG which improves its robustness with respect to variations in the preconditioner at the price of one more inner product computation per iteration.[¶] Note that the procedure is recursive: it is initially called from the main (outer) iterative scheme with level index $k=1$, but at each level $k \leq \ell$ the procedure calls itself with level index $k+1$ to (approximately) solve system(s) with the preconditioner of S_k .

Algorithm 6.1 (Recursive procedure to solve $B_k \mathbf{v}^{(k)} = \mathbf{g}^{(k)}$)

$$\mathbf{v}^{(k)} = \begin{pmatrix} \mathbf{v}_1^{(k)} \\ \mathbf{v}_2^{(k)} \end{pmatrix}, \quad \mathbf{g}^{(k)} = \begin{pmatrix} \mathbf{g}_1^{(k)} \\ \mathbf{g}_2^{(k)} \end{pmatrix}$$

$$B_k = \begin{pmatrix} P^{(k)} & \\ A_{21}^{(k)} & S_k \end{pmatrix} \begin{pmatrix} I & P^{(k)-1} A_{12}^{(k)} \\ & I \end{pmatrix} \text{ approximates } A_k = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{pmatrix}$$

where $A_1 = A$ and $A_k = S_{k-1}$ for $k > 1$.

1. $\mathbf{w}_1^{(k)} = P^{(k)-1} \mathbf{g}_1^{(k)}$
2. $\mathbf{w}_2^{(k)} = \mathbf{g}_2^{(k)} - A_{21}^{(k)} \mathbf{w}_1^{(k)}$
3. if $k = \ell + 1$:
 - $\mathbf{v}_2^{(k)} = S_k^{-1} \mathbf{w}_2^{(k)}$
 - else if $\text{maxit}^{(k)} = 1$:
 - apply procedure to solve $B_{k+1} \mathbf{v}_2^{(k)} = \mathbf{w}_2^{(k)}$
 - else (FCG(1) iteration to solve $S_k \mathbf{v}_2^{(k)} = \mathbf{w}_2^{(k)}$):
 - $\mathbf{u}_0 = \mathbf{0}; \mathbf{r}_0 = \mathbf{w}_2^{(k)}; i = 0$
 - while $i < \text{maxit}^{(k)}$ and $\|\mathbf{r}_i\| > \varepsilon \|\mathbf{w}_2^{(k)}\|$:
 - apply procedure to solve $B_{k+1} \mathbf{z}_i = \mathbf{r}_i$
 - $\mathbf{d}_i = \mathbf{z}_i - \frac{(\mathbf{z}_i, S_k \mathbf{d}_{i-1})}{(\mathbf{d}_{i-1}, S_k \mathbf{d}_{i-1})} \mathbf{d}_{i-1}$ ($i=0 : \mathbf{d}_i = \mathbf{z}_i$)
 - $\mathbf{u}_{i+1} = \mathbf{u}_i + \frac{(\mathbf{d}_i, \mathbf{r}_i)}{(\mathbf{d}_i, S_k \mathbf{d}_i)} \mathbf{d}_i$
 - $\mathbf{r}_{i+1} = \mathbf{r}_i - \frac{(\mathbf{d}_i, \mathbf{r}_i)}{(\mathbf{d}_i, S_k \mathbf{d}_i)} S_k \mathbf{d}_i$
 - $i = i + 1$
 - end while
 - $\mathbf{v}_2^{(k)} = \mathbf{u}_i$
 - end if
 - 4. $\mathbf{v}_1^{(k)} = \mathbf{w}_1^{(k)} - P^{(k)-1} A_{12}^{(k)} \mathbf{v}_2^{(k)}$

[¶]the same method is referred to as IPCG in Reference [45].

7. NUMERICAL RESULTS

We first consider the model 2D (3D) anisotropic equation

$$-a \frac{\partial^2 u}{\partial x^2} - b \frac{\partial^2 u}{\partial y^2} \left(-c \frac{\partial^2 u}{\partial z^2} \right) = 1$$

in the unit square (cube) with homogeneous Neumann boundary conditions everywhere, except on the right boundary $x=1, 0 \leq y \leq 1$ ($x=1, 0 \leq y, z \leq 1$) where we set homogeneous Dirichlet boundary conditions. We use five (seven)-point finite difference approximation on a uniform mesh with constant mesh size h in all directions, and test the multilevel preconditioner presented in Sections 3–6 for the arising linear system. Since we are in the SPD case, we use the FCG(1) method for both inner and outer iterations.

Besides this preconditioner, referred to as *multilevel with automatic coarsening, aggregation and FCG inner iterations*, we also consider as ‘reference’ scheme the multilevel method from Reference [34] which uses as basic ingredients standard geometric coarsening, the same MILU(0) factorization to approximate A_{11} and an algebraic approximation of the Schur complement that, for the considered problems, corresponds to the same five- or seven-point stencil applied on the coarse grid (with some scaling factor). For the sake of completeness we include two versions: the one based on polynomial approximations to S^{-1} as described at the beginning of Section 6 (*multilevel with geometric coarsening and polynomial approximation*) and the one based on FCG(1) inner iterations. (*Multilevel with geometric coarsening and FCG inner iterations.*) For the former, in 2D examples, we use second order Chebyshev polynomials as indicated in Reference [34]. In 3D examples, no theoretical analysis is available that allows to set up appropriate polynomials; for the case $a=b=c=1$ we nevertheless define the best possible ones by handmade optimization. Note that the preconditioner in then no more variable from step to step, so we use a standard implementation of CG with this preconditioner.

We report in Tables I and II the number of iterations and the number of floating point operations (*flops*), setup excluded, needed to reduce the relative residual error by 10^{-6} whenever using the zero vector as initial approximation. To assess memory requirements, we also give the relative ‘complexities’

$$C_n = 1 + \frac{\sum_{k=1}^{\ell+1} n_k}{n}, \quad C_{nz} = 1 + \frac{\sum_{k=1}^{\ell+1} nz(S_k)}{nz(A)}$$

where n_k is the size of S_k .

As expected the new multilevel preconditioner is less efficient on model problems than previous well tuned specific schemes. However, one does not loses too much, especially if one takes into account that the comparison with schemes based on polynomial approximations is somewhat artificial because of their very restricted applicability. The comparison with the scheme using geometric coarsening and FCG inner iterations is more instructive and tells us that the use of our automatic coarsening and aggregation strategies has only a limited impact on performances, especially in 2D problems. In 3D problems, the situation is somewhat less satisfactory for ‘easy’ configurations, but this is clearly compensated by an improved robustness for ‘difficult’ configurations.

Table I. Results for the 2D model anisotropic problem; *flops* are given per unknown.

	$h^{-1}=256$				$h^{-1}=512$				$h^{-1}=1024$			
	C_n	C_{nz}	No. of. it.	<i>Flops</i>	C_n	C_{nz}	No. of. it.	<i>Flops</i>	C_n	C_{nz}	No. of. it.	<i>Flops</i>
<i>Multilevel with geometric coarsening and polynomial approximation</i>												
$a=1, b=1$	1.3	1.3	11	598	1.3	1.3	11	600	1.3	1.3	12	657
$a=10^{-2}, b=1$	1.3	1.3	14	756	1.3	1.3	14	764	1.3	1.3	14	767
$a=10^{-4}, b=1$	1.3	1.3	15	811	1.3	1.3	16	873	1.3	1.3	16	876
<i>Multilevel with geometric coarsening and FCG inner iterations</i>												
$a=1, b=1$	1.3	1.3	13	784	1.3	1.3	12	763	1.3	1.3	13	848
$a=10^{-2}, b=1$	1.3	1.3	13	866	1.3	1.3	14	962	1.3	1.3	14	990
$a=10^{-4}, b=1$	1.3	1.3	15	997	1.3	1.3	15	1008	1.3	1.3	15	1036
<i>Multilevel with automatic coarsening, aggregation and FCG inner iterations</i>												
$a=1, b=1$	1.7	2.1	19	909	1.7	2.2	19	1016	1.7	2.2	20	1076
$a=10^{-2}, b=1$	2.0	2.8	15	854	2.0	2.9	22	1400	2.0	2.9	22	1471
$a=10^{-4}, b=1$	2.0	2.8	14	777	2.0	2.8	14	853	2.0	2.8	15	969

Table II. Results for the 3D model anisotropic problem; *flops* are given per unknown.

	$h^{-1}=50$				$h^{-1}=100$			
	C_n	C_{nz}	No. of. it.	<i>Flops</i>	C_n	C_{nz}	No. of. it.	<i>Flops</i>
<i>Multilevel with geometric coarsening and polynomial approximation</i>								
$a=1, b=1, c=1$	1.1	1.1	15	1024	1.1	1.1	15	1025
<i>Multilevel with geometric coarsening and FCG inner iterations</i>								
$a=1, b=1, c=1$	1.1	1.1	18	1193	1.1	1.1	18	1221
$a=10^{-2}, b=1, c=1$	1.1	1.1	26	1993	1.1	1.1	27	2197
$a=10^{-4}, b=1, c=1$	1.1	1.1	70	7446	1.1	1.1	91	12678
$a=10^{-2}, b=10^{-2}, c=1$	1.1	1.1	35	2780	1.1	1.1	37	3192
$a=10^{-4}, b=10^{-2}, c=1$	1.1	1.1	81	8758	1.1	1.1	98	13855
$a=10^{-4}, b=10^{-4}, c=1$	1.1	1.1	52	4478	1.1	1.1	55	5741
<i>Multilevel with automatic coarsening, aggregation and FCG inner iterations</i>								
$a=1, b=1, c=1$	1.6	2.6	29	2399	1.6	2.6	29	3003
$a=10^{-2}, b=1, c=1$	1.7	2.9	17	1948	1.7	2.9	17	2188
$a=10^{-4}, b=1, c=1$	1.7	2.9	17	1953	1.7	2.9	17	2194
$a=10^{-2}, b=10^{-2}, c=1$	2.0	3.9	17	2308	2.0	4.0	17	2758
$a=10^{-4}, b=10^{-2}, c=1$	2.0	3.9	16	2093	2.0	4.0	16	2554
$a=10^{-4}, b=10^{-4}, c=1$	2.0	3.9	14	1746	2.0	4.0	16	2278

Concerning complexities, note that the larger values for the new method essentially originate from the first stage which, as discussed at the end of Section 3, is a standard red–black reduction with exact elimination of the red nodes. For the matrices considered here, the resulting S (i.e. the matrix for the black nodes) corresponds then to a nine-point stencil in 2D and a 19-point stencil in 3D. Therefore, C_{nz} can never be less than 2 in 2D problems and 2.5 in 3D problems.

We next consider more difficult 2D (3D) convection–diffusions

$$-v\Delta u + \bar{v} \nabla u = 0$$

in the unit square (cube), with homogeneous Dirichlet boundary conditions everywhere, except on the boundary $y=1$, $0 \leq x \leq 1$ ($z=1$, $0 \leq x, y \leq 1$) where we set $u=1$. Here, v is a constant parameter (viscosity) and \bar{v} a convective flow. We tested several configurations for this flow, according the following list.

- Problem *Poisson*: $\bar{v}=0$.
- 2D problem *Constant flow*:

$$\bar{v} = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}$$

- 2D problem *Rotating flow*:

$$\bar{v} = \begin{pmatrix} \sin(\pi x) \cos(\pi y) \\ -\cos(\pi x) \sin(\pi y) \end{pmatrix}$$

- 2D problem *Highly varying flow* [49]:

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

- 3D problem *Constant flow*:

$$\bar{v} = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \\ 0 \end{pmatrix}$$

- 3D problem *Rotating flow 1*:

$$\bar{v} = \begin{pmatrix} z(1-z) \sin(\pi x) \cos(\pi y) \\ -z(1-z) \cos(\pi x) \sin(\pi y) \\ 0 \end{pmatrix}$$

- 3D problem *Rotating flow 2*:

$$\bar{v} = \begin{pmatrix} 2z(1-2z) \sin(\pi x) \cos(\pi y) \\ -2z(1-2z) \cos(\pi x) \sin(\pi y) z(1-z) \\ 0 \end{pmatrix} \quad \text{for } z \leq 0.5 \text{ and } \bar{v}=0 \text{ elsewhere}$$

- 3D problem *Highly varying flow*:

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

Table III. Results on 2D convection–diffusion equations for the multilevel method with automatic coarsening, aggregation and FGMRES inner iterations; *flops* are given per unknown.

	$h^{-1}=256$				$h^{-1}=512$				$h^{-1}=1024$			
	C_n	C_{nz}	No. of. it.	<i>Flops</i>	C_n	C_{nz}	No. of. it.	<i>Flops</i>	C_n	C_{nz}	No. of. it.	<i>Flops</i>
$\nu=10^{-3}$												
Poisson	1.7	2.2	15	1037	1.7	2.2	15	1169	1.7	2.2	15	1271
Constant fl.	1.7	2.2	15	1216	1.7	2.2	17	1450	1.7	2.2	18	1898
Rotating fl.	1.8	2.5	18	1857	1.7	2.2	18	1767	1.7	2.2	21	2071
High. var. fl.	1.7	2.2	19	1385	1.7	2.2	18	1770	1.7	2.2	17	2250
<i>Highly varying flow</i>												
$\nu=1$	1.7	2.2	15	1075	1.7	2.2	15	1145	1.7	2.2	15	1305
$\nu=10^{-2}$	1.7	2.2	16	1153	1.7	2.2	15	1263	1.7	2.2	15	1346
$\nu=10^{-4}$	1.9	2.6	30	1977	1.8	2.5	19	2015	1.7	2.2	18	2213
$\nu=10^{-6}$	1.9	2.8	29	2356	1.9	2.8	31	2672	1.9	2.8	32	3242
<i>Highly varying flow with stretched grid</i>												
$\nu=1$	1.8	2.6	16	1479	1.9	2.6	15	1606	1.9	2.6	15	1998
$\nu=10^{-2}$	1.8	2.6	14	1357	1.9	2.6	15	1573	1.9	2.6	15	1922
$\nu=10^{-4}$	1.8	2.5	17	1612	1.9	2.6	19	2044	1.9	2.6	19	2508
$\nu=10^{-6}$	1.8	2.5	21	2015	1.8	2.5	22	2216	1.8	2.5	22	2625

Table IV. Results on 3D convection–diffusion equations for the multilevel method with automatic coarsening, aggregation and FGMRES inner iterations; *flops* are given per unknown.

	$h^{-1}=50$				$h^{-1}=100$			
	C_n	C_{nz}	No. of. it.	<i>Flops</i>	C_n	C_{nz}	No. of. it.	<i>Flops</i>
$\nu=10^{-3}$								
Poisson	1.6	2.6	20	1984	1.6	2.6	19	2101
Constant fl.	1.9	3.3	15	2819	1.9	3.3	15	2823
Rot. fl. 1	1.7	2.8	25	2927	1.6	2.7	22	3111
Rot. fl. 2	1.6	2.7	25	2629	1.6	2.6	24	2949
High. var. fl.	1.7	2.8	25	3070	1.6	2.7	22	3255
<i>Highly varying flow</i>								
$\nu=1$	1.6	2.6	22	2185	1.6	2.6	17	1826
$\nu=10^{-2}$	1.6	2.6	18	1907	1.6	2.6	17	2065
$\nu=10^{-4}$	1.9	3.6	31	3918	1.9	3.5	40	4863
$\nu=10^{-6}$	1.9	3.7	32	4421	1.9	3.7	38	5813
<i>Highly varying flow with stretched grid</i>								
$\nu=1$	1.8	3.1	23	3409	1.8	3.2	23	3761
$\nu=10^{-2}$	1.8	3.1	21	3096	1.8	3.2	25	3986
$\nu=10^{-4}$	1.7	3.0	23	3258	1.8	3.1	24	3729
$\nu=10^{-6}$	1.9	3.5	44	5233	1.9	3.5	43	5391

Table V. Results for the 2D 'Highly varying flow' problem; the multilevel method is the one with automatic coarsening, aggregation and FGMRES inner iterations; CPU times are given in seconds.

	$h^{-1} = 256$						$h^{-1} = 512$						$h^{-1} = 1024$								
	CPU time			CPU time			CPU time			CPU time			CPU time			CPU time					
	No. of. it.	Setup	Sol.	Total	No. of. it.	Setup	Sol.	Total	No. of. it.	Setup	Sol.	Total	No. of. it.	Setup	Sol.	Total	No. of. it.	Setup	Sol.	Total	
$v = 1$																					
Mult.	15	2.60	2.55	5.15	15	9.95	12.1	22.0	15	150.	66.8	216.	15	150.	66.8	216.	15	150.	66.8	216.	15
ILU	38	3.14	9.63	12.8	98	12.5	103.	115.	268	50.7	1400.	1451.	268	50.7	1400.	1451.	268	50.7	1400.	1451.	268
$v = 10^{-2}$																					
Mult.	16	1.96	2.81	4.77	15	30.0	12.3	42.3	15	67.9	66.3	134.	15	67.9	66.3	134.	15	67.9	66.3	134.	15
ILU	69	3.30	26.6	29.9	143	12.7	162.	175.	317	50.3	1640.	1690.	317	50.3	1640.	1690.	317	50.3	1640.	1690.	317
$v = 10^{-4}$																					
Mult.	30	2.17	4.96	7.13	19	13.3	19.9	33.2	18	108.	106.	214.	18	108.	106.	214.	18	108.	106.	214.	18
ILU	155	3.30	56.3	59.6	839	12.6	902.	915.	>999	—	—	—	>999	—	—	—	>999	—	—	—	>999
$v = 10^{-6}$																					
Mult.	29	2.67	5.47	7.94	31	11.1	27.8	38.9	32	204.	165.	369.	32	204.	165.	369.	32	204.	165.	369.	32
ILU	158	3.07	39.9	43.0	720	12.7	833.	846.	>999	—	—	—	>999	—	—	—	>999	—	—	—	>999

Table VI. Results for the 3D ‘Highly varying flow’ problem the multilevel method is the one with automatic coarsening, aggregation and FGMRES inner iterations; CPU times are given in seconds.

	$h^{-1}=50$				$h^{-1}=100$			
	CPU time				CPU time			
	No. of. it.	Setup	Sol.	Total	No. of. it.	Setup	Sol.	Total
$v=1$								
Mult.	22	10.6	12.4	23.0	17	109.	78.3	187.
ILU	23	11.8	12.9	24.7	54	105.	310.	415.
$v=10^{-2}$								
Mult.	18	10.6	9.57	20.2	17	111.	90.9	202.
ILU	35	11.8	19.6	31.4	81	106.	469.	575.
$v=10^{-4}$								
Mult.	31	14.9	15.6	30.5	40	163.	251.	414.
ILU	73	11.8	40.9	52.7	349	105.	2000.	2105.
$v=10^{-6}$								
Mult.	32	13.4	17.2	30.6	38	172.	267.	439.
ILU	68	11.7	38.1	49.8	339	105.	1960.	2065.

Note that all these flows are divergence free, and most of them have closed characteristic curves, making difficult their solution with classical multigrid methods as $v \rightarrow 0$.

Here, we use five (seven)-point finite difference approximation, with standard upwinding for the first-order derivatives. We always use a uniform mesh with constant mesh size h in all directions, except when referring to stretched grids. In these cases, we use a grid which is refined in the neighbourhood of the boundaries, in such a way that the ratio of maximum mesh size to minimum mesh size is equal to 200 in 2D and 50 in 3D, the ratio of subsequent mesh sizes being kept constant (Then, the parameter h referenced in the tables indicates that the number of grid point in each direction is equal to $h^{-1} + 1$.)

Since the matrices are non-symmetric, we use here FGMRES [12, 47] for both inner and outer iterations, with restart of the outer scheme each 10 iterations (as discussed in Section 6, restarting is irrelevant for inner iterations because at most a few are allowed). As above, the stopping criterion is a decrease of the relative residual error by 10^{-6} and the initial approximation is the zero vector.

The results are reported in Tables III and IV. One may summarize them with the observation that the ratio between the actual flops count and the reference flops count for the Poisson problem presents, for each columns of Tables III and IV, a mean value ranging from 1.47 (2D, $h^{-1}=512$) to 1.67 (3D, $h^{-1}=100$). Moreover, the worst such ratio never exceeds $\frac{5}{3}$ of the corresponding mean value, showing a low discrepancy in the performances. On the other hand, efficiency is not really sacrificed: the above ‘reference flops count’ is at worst 2.05 larger than the corresponding flops count from Table I or II for the model SPD problem ($a=b(=c)=1$) solved with the ‘model’ optimal order multilevel method (based on geometric coarsening and polynomial approximation).

Finally, we compare in CPU time our multilevel method with standard ILU preconditioning [12]. As preliminary tests showed that ILU with low fill-in is not robust at all (see also Reference [35, p. 265]), we increased the level of fill (defined as in Reference [12, p. 280]) until the setup time reaches values comparable to that needed for the multilevel method, leading to consider ILU(7) in 2D and ILU(3) in 3D.

The results are reported in Tables V and VI. Despite the high level of fill-in (with memory requirements actually larger than for the multilevel method), ILU preconditioning appears not only less efficient for small mesh sizes (as expected), but also much less robust with respect to the numerical difficulties associated with small values of ν .

ACKNOWLEDGEMENTS

This work was supported by Fonds National de la Recherche Scientifique.

REFERENCES

1. Hackbusch W. *Multi-grid Methods and Applications*. Springer: Berlin, 1985.
2. Trottenberg U, Oosterlee CW, Schüller A. *Multigrid*. Academic Press: London, 2001.
3. Wesseling P. *An Introduction to Multigrid Methods*. Wiley: Chichester, 1992.
4. Bank RE, Smith RK. An algebraic multilevel multigraph algorithm. *SIAM Journal on Scientific Computing* 2002; **23**:1572–1592.
5. Braess D. Towards algebraic multigrid for elliptic problems of second order. *Computing* 1995; **55**:379–393.
6. Brezina M, Cleary AJ, Falgout RD, Henson VE, Jones JE, Manteuffel TA, McCormick SF, Ruge JW. Algebraic multigrid based on element interpolation (AMGe). *SIAM Journal on Scientific Computing* 2000; **22**: 1570–1592.
7. Jones JE, Vassilevski PS. AMGe based on element agglomeration. *SIAM Journal on Scientific Computing* 2001; **23**:109–133.
8. Ruge JW, Stüben K. Algebraic multigrid (AMG). In *Multigrid Methods*, McCormick SF (ed.), *Frontiers in Applied Mathematics*, vol. 3, SIAM: Philadelphia, 1987; 73–130.
9. Stüben K. *An Introduction to Algebraic Multigrid*. In Trottenberg V, Oosterlee CW, Schuller (eds). Academic Press: London, 2001; 413–532 (Appendix A).
10. Vaněk P, Mandel J, Brezina M. Algebraic multigrid based on smoothed aggregation for second and fourth order problems. *Computing* 1996; **56**:179–196.
11. Axelsson P. *Iterative Solution Methods*. Cambridge University Press: Cambridge, 1994.
12. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing: New York, 1996.
13. Bank RE, Wagner C. Multilevel ILU decomposition. *Numerische Mathematik* 1999; **82**:543–576.
14. Botta EFF, Wubs FW. Matrix renumbering ILU: an effective algebraic multilevel ILU preconditioner for sparse matrices. *SIAM Journal on Matrix Analysis and Applications* 1999; **20**:1007–1026.
15. Notay Y, Ould Amar Z. A nearly optimal preconditioning based on recursive red–black orderings. *Numerical Linear Algebra with Applications* 1997; **4**:369–391.
16. Saad Y. ILUM: a parallel multi-elimination ILU preconditioner for general sparse matrices. *SIAM Journal on Scientific Computing* 1996; **17**:830–847.
17. Saad Y, Suchoemel B. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. *Technical Report unsi-99-107-REVIS*, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, 2001.
18. Saad Y, Zhang J. BILUM: block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* 1999; **20**:2103–2121.
19. van der Ploeg A, Botta EFF, Wubs FW. Nested grids ILU-decomposition (NGILU). *Journal of Computational and Applied Mathematics* 1996; **66**:515–526.
20. Bank RE, Smith RK. The incomplete factorization multigraph algorithm. *SIAM Journal on Scientific Computing* 1999; **20**:1349–1364.
21. Reusken A. On the approximate cyclic reduction preconditioner. *SIAM Journal on Scientific Computing* 1999; **21**:565–590.
22. Axelsson O. The method of diagonal compensation of reduced matrix entries and multilevel iteration. *Journal of Computational and Applied Mathematics* 1991; **38**:31–43.

23. Axelsson O, Eijkhout V. Analysis of recursive 5-point/9-point factorization method. In *Preconditioned Conjugate Gradient Methods*, Axelsson O, Kolotilina LYu (eds), Lectures Notes in Mathematics, vol. 1457. Springer: Berlin, Heidelberg, New York, 1990; 154–173.
24. Axelsson O, Eijkhout V. The nested recursive two level factorization for nine-point difference matrices. *SIAM Journal on Scientific Computing* 1991; **12**:1373–1400.
25. Axelsson O, Neytcheva M. Algebraic multilevel iterations for Stieltjes matrices. *Numerical Linear Algebra with Applications* 1994; **1**:213–236.
26. Axelsson O, Padiy A. On the additive version of the algebraic multilevel iteration method for anisotropic elliptic problems. *SIAM Journal on Scientific Computing* 1999; **20**:1807–1830.
27. Axelsson O, Vassilevski PS. Algebraic multilevel preconditioning methods, I. *Numerische Mathematik* 1989; **56**:157–177.
28. Axelsson O, Vassilevski PS. Algebraic multilevel preconditioning methods, II. *SIAM Journal on Numerical Analysis* 1990; **27**:1569–1590.
29. Axelsson O, Vassilevski PS. Variable-step multilevel preconditioning methods. I. self adjoint and positive definite elliptic problems. *Numerical Linear Algebra with Applications* 1994; **1**:75–101.
30. Margenov S. Semi-coarsening AMLI algorithms for elasticity problems. *Numerical Linear Algebra with Applications* 1999; **5**:347–362.
31. Margenov S, Vassilevski P. Algebraic multilevel preconditioning of anisotropic elliptic problems. *SIAM Journal on Scientific Computing* 1994; **15**:1026–1037.
32. Notay Y. Using approximate inverses in algebraic multilevel methods. *Numerische Mathematik* 1998; **80**:397–417.
33. Notay Y. Optimal V cycle algebraic multilevel preconditioning. *Numerical Linear Algebra with Applications* 1998; **5**:441–459.
34. Notay Y. Optimal order preconditioning of finite difference matrices. *SIAM Journal on Scientific Computing* 2000; **21**:1991–2007.
35. Notay Y. A robust algebraic multilevel preconditioner for non symmetric M -matrices. *Numerical Linear Algebra with Applications* 2000; **7**:243–267.
36. Reusken A. A multigrid method based on incomplete Gaussian elimination. *Numerical Linear Algebra with Applications* 1996; **3**:369–390.
37. Vassilevski P. Nearly optimal iterative methods for solving finite element elliptic equations based on the multilevel splitting of the matrix. *Technical Report # 1989-09*, Institute for Scientific Computation, University of Wyoming, Laramie, U.S.A., 1989.
38. Vassilevski P. Hybrid V-cycle algebraic multilevel preconditioners. *Mathematics of Computation* 1992; **58**:489–512.
39. Vassilevski P. On two ways of stabilizing the hierarchical basis multilevel methods. *SIAM Review* 1997; **39**:18–53.
40. Axelsson O. On multigrid methods of the two-level type. In *Multigrid Methods*, Hackbusch W, Trottenberg U (eds), Lectures Notes in Mathematics, vol. 960. Springer: Berlin, Heidelberg, New York, 1981; 352–367.
41. Axelsson O, Gustafsson I. Preconditioning and two-level multigrid methods of arbitrary degree of approximation. *Mathematics of Computation* 1983; **40**:214–242.
42. Bank RE, Dupont TF. Analysis of a two-level scheme for solving finite element equations. *Technical Report CNA-159*, Center for Numerical Analysis, The University of Texas at Austin, TX, U.S.A., 1980.
43. Eijkhout V, Vassilevski P. The role of the strengthened c.b.s. inequality in multilevel methods. *SIAM Review* 1991; **33**:405–419.
44. Axelsson O, Vassilevski PS. A black-box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. *SIAM Journal on Matrix Analysis and Applications* 1991; **12**:625–644.
45. Golub GH, Ye Q. Inexact preconditioned conjugate gradient method with inner-outer iterations. *SIAM Journal on Scientific Computing* 1999; **21**:1305–1320.
46. Notay Y. Flexible conjugate gradients. *SIAM Journal on Scientific Computing* 2000; **22**:1444–1460.
47. Saad Y. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing* 1993; **14**:461–469.
48. van der Vorst H, Vuik C. GMRESR: a family of nested GMRES methods. *Numerical Linear Algebra with Applications* 1994; **1**:369–386.
49. Elman H, Silvester D. Fast non-symmetric iterations and preconditioning for Navier–Stokes equations. *SIAM Journal on Scientific Computing* 1996; **17**:33–46.