LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-233772

On Some Versions of the Element Agglomeration AMGe Method

I.V. Lashuk, P.S. Vassilevski

August 17, 2007

Numerical Linear Algebra with Applications

ON SOME VERSIONS OF THE ELEMENT AGGLOMERATION AMGe METHOD

ILYA LASHUK AND PANAYOT S. VASSILEVSKI

Abstract. The present paper deals with element-based AMG methods that target linear systems of equations coming from finite element discretizations of elliptic PDEs. The individual element information (element matrices and element topology) is the main input to construct the AMG hierarchy. We study a number of variants of the spectral agglomerate element based AMG method. The core of the algorithms relies on element agglomeration utilizing the element topology (built recursively from fine to coarse levels). The actual selection of the coarse degrees of freedom (dofs) is based on solving large number of local eigenvalue problems. Additionally, we investigate strategies for adaptive AMG as well as multigrid cycles that are more expensive than the V–cycle utilizing simple interpolation matrices and nested conjugate gradient (CG) based recursive calls between the levels. The presented algorithms are illustrated with an extensive set of experiments based on a matlab implementation of the methods.

1. Introduction

Multigrid (MG) (e.g., [14]) is one of the most efficient and natural methods for solving linear systems of equations coming from partial differential equations (PDEs) discretized on a sequence of grids. In *algebraic* multigrid (AMG) [2, 3, 12, 13] the necessary MG components (coarse grids, coarse-grid operators, interpolation operators) are built by the solver algorithm opposite to geometric multigrid where these components are naturally given by the discretization. An extreme case of an algebraic multigrid approach would lead to a black box solver, i.e., an algorithm which would only use the linear system matrix as input data. In practice, all AMG methods utilize (often assumed) some additional information about the class of problems they are applied to. In the present paper, we deal with class of problems that come from finite element discretization of elliptic PDEs. More specifically, we focus on variants of the element–based AMG (or AMGe) methods developed in [4, 9, 8, 6, 7].

We have implemented several algorithms in matlab in the framework of the element agglomeration spectral AMGe ([7]) and illustrate their performance. Our matlab implementation allows for further extensions and offers potential for illustrating other element based AMG algorithms. More specifically, the paper is structured as follows. Section 2 describes the main building tools of the implemented algorithms. In particular, in Section 2.1 we describe the required input, Section 2.2 reviews some details

Date: July 20, 2007, today is August 9, 2007.

1991 *Mathematics Subject Classification.* 65N30, 65N15.

This work was performed under the auspices of the U. S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

on element agglomeration, and in § 2.3 the so–called "minimal intersection sets" are introduced. These sets are later used, in Section 2.4, to define the coarse degrees of freedom by solving local eigenvalue problems (associated with each minimal intersection set). The recursive nature of the algorithm is summarized in § 2.5. The spectral way of selecting coarse degrees of freedom naturally leads to the construction of tentative prolongators. In § 2.6 the need to improve on the stability of the tentative prolongator is outlined.

Another direction of extending the presented method that we pursue utilizes more sophisticated multigrid cycles that are based on inner (between the levels) preconditioned conjugate gradient (CG) iterations. Such idea goes back to the non–linear AMLI methods in [1]. This respective method, referred to as "nested-CG" (AMLI) cycle is described in Section 2.7.

Section 3 contains description of a specific implementation of the adaptive AMG method (cf., [5]) in the present AMGe setting. We consider also an option which does not utilize the actual element matrices (but does use the element topology relations). This results in a new AMGe algorithm which we view as a main algorithmic contribution of the present paper.

The V–cycle and "nested-CG" cycle are illustrated in Sections 4 and 5, whereas the results utilizing AMG adaptivity are found in the last section 6.

The main result of the present work is that it describes some performance results of a number of AMGe algorithms; some are simple variations of previously implemented ones, in addition to some newly developed ones, such as the "nested-CG" AMLI cycles, as well as the adaptive element agglomerate AMGe.

2. Basic building tools

2.1. **Input data.** We assume finite element setting of the problem in the framework of relation tables as described in detail in [16].

More specifically, our matlab software requires the following input data:

- The "element_dof" relation. It can be defined by treating each element as list of degrees of freedom (dofs). If the dofs and elements are respectively numbered as $1, \ldots, N_D$ and $1, \ldots, N_E$, then the incidence relation "element_dof" can be represented as the boolean sparse matrix M of size $N_E \times N_D$ with entries
$$M_{ij} = \begin{cases} 1, & \text{if element } i \text{ contains dof } j, \\ 0, & \text{otherwise.} \end{cases}$$
- The "element_face" relation. This relation describes the incidence "element i" has a "face j". For triangular elements, element faces are the triangles sides. This relation can be used to define neighboring elements; namely, we say that two elements are neighbors if they share a common face. This defines the relation table "element_element". Again, the relations "element_face" and "element_element" can be implemented as boolean sparse matrices.
- List of boundary dofs. This is a list of dofs which are on the boundary where essential boundary conditions are to be imposed..

- We also need access to the individual element matrices. We assume that all these matrices are symmetric positive semi-definite (or SPSD for short). The actual matrix of the linear system of our main interest is built in 2 steps:
  - First, we assemble a global (in general, singular, SPSD) matrix from the individual element matrices in the usual way, i.e., according to the formula
$$w^T A v = \sum_\tau w_\tau^T A_\tau v_\tau$$
where A_τ denotes the local matrix corresponding to element τ. For a given v and a set of dofs τ, v_τ stands for the restriction of v to the subset of indices (dofs) corresponding to τ.
  - Second, we impose the essential boundary conditions on the global assembled matrix A. That is, for any dof d on "essential boundary" we set to zero all off-diagonal entries in the d-th row and the d-th column of A. For the class of problems we consider, imposing essential boundary conditions in this way produces SPD (symmetric positive definite), hence non-singular matrices.

We refer to the initial set of elements and respective dofs *fine-grid elements* (or fine elements) and *fine-grid dofs* (or fine dofs).

We note that the right–hand side vector (required input in the solution phase) is not needed as input in the construction of the actual AMGe solver.

2.2. **Agglomerating elements.** Agglomeration refers to a partition of the set of all (fine) elements into non-intersecting subsets, called *agglomerates* or *agglomerated elements* (AEs). More specifically, we treat the set of elements as vertices of the undirected graph where two vertices (elements) are linked with an edge if and only if they are neighbors, i.e., share a face. This graph defines (as described before) the relation "element_element". If the relations "element_face" and "face_element" are implemented as (boolean) sparse matrices, then "element_element" equals the product "element_face" $\times$ "face_element".

We use the graph partitioner METIS ([10]) to partition the "element_element" relation into a desired number of components (a user specified parameter referred to as *coarsening factor* or "Crs. f."). We make sure that the produced components are connected. Each (connected) component defines an agglomerate. Figure 1 gives an illustration of this process. We also view each agglomerated element as a list of fine degrees of freedom (the union of the degrees of freedom that come with the fine–grid elements forming the agglomerate).

2.3. **Building minimal intersection sets.** Having created agglomerated elements, we can split the set of all (fine) degrees of freedom into non-overlapping partitions, referred to as *minimal intersection sets* (as in [17]). These partitions represent equivalence classes with respect to the equivalence relation that two dofs belong to a same minimal intersection set if and only if they belong to exactly the same set of agglomerated elements. For example, all interior dofs of an AE constitute a minimal intersection set. A boundary (its interior only) between two AEs constitutes a minimal intersection set, etc. Figure 2 illustrates this process.

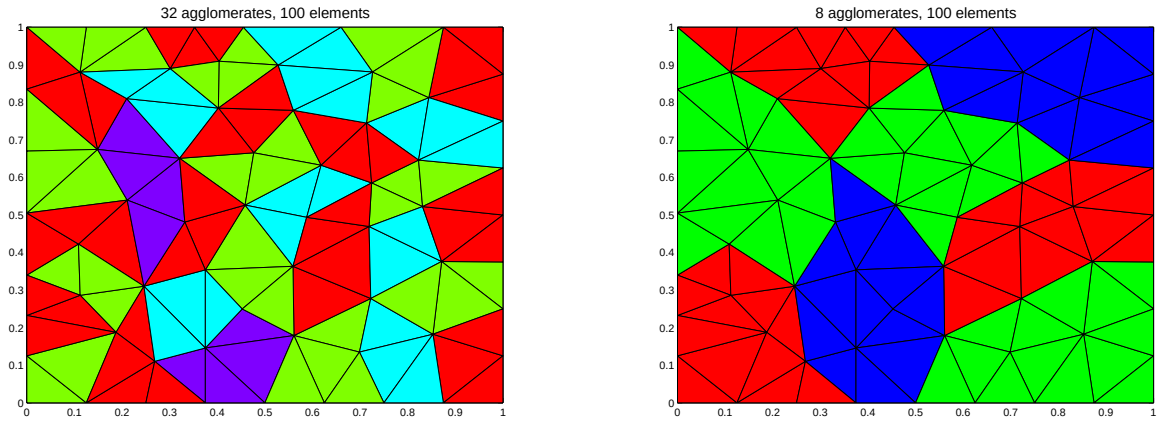

FIGURE 1. Agglomerated elements

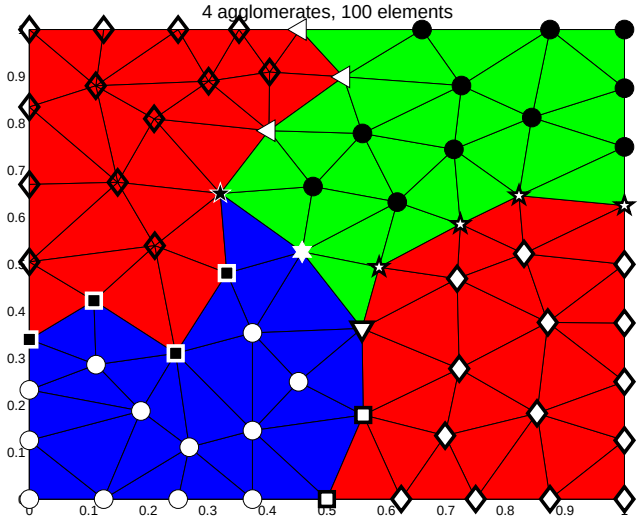

FIGURE 2. Minimal intersection sets

2.4. **Forming Schur complements, computing eigenvalues and building tentative prolongators.** For each minimal intersection set I, we define its neighborhood $N(I)$ to be all (fine) elements that intersect I (i.e., having a common dof with I). We assemble the local matrix $A_{N(I)}$ from the element matrices corresponding to the elements in $N(I)$ and then compute its Schur complement S_I by eliminating all dofs outside I. Note that both $A_{N(I)}$ and S_I are SPSD.

Next, we calculate all the eigenpairs $(\lambda_{I,k},\ q_{I,k})$, $k = 1,\ \ldots,\ |I|$, of S_I. Here, $|I|$ stands for the cardinality of the set I. Let Q_I be the matrix whose columns are the (orthogonal and normalized) eigenvectors $q_{I,k}$ of S_I.

Due to our assumptions, we have that all eigenvalues $\lambda_{I,k}$ of S_I are non-negative.

Based on a user specified tolerance $\tau \in [0,\ 1]$ (referred to as the *spectral tolerance*), we partition Q_I into two submatrices. The first one, $Q_{I,c}$, has columns that are the eigenvectors $q_{I,k}$ corresponding to the lower part of the spectrum of S_I, i.e., all $q_{I,k}$ for

k such that $\lambda_k \leq \tau \, \|A_{N(I)}\|$, where $\|.\|$ stands for the ℓ_∞-norm of the neighborhood matrix $A_{N(I)}$ (or the diagonal of $A_{N(I)}$). The second block $Q_{I,f}$ contains all remaining columns of Q_I (i.e., eigenvectors corresponding to the upper part of the spectrum, $\lambda_{I,\,k} > \tau \, \|A_{N(I)}\|$).

The *tentative prolongator* Q_c is then formed from the blocks $Q_{I,c}$ (extended by zero outside the set I), i.e., $Q_c = [\, \ldots, \, Q_{I,c}, \, \ldots \,]$. Since all matrices $Q_{I,c}$ are orthogonal and the sets I do not overlap, Q_c is also an orthogonal matrix.

2.5. **Proceeding by recursion.** So far, we have constructed the components necessary for a 2-level method. More specifically, we assume a standard smoother M such as (block) Gauss-Seidel. The coarse-grid matrix A^c is obtained by standard (Galerkin) procedure; namely, with $P = Q_c$ (or some improved version of Q_c), we set $A^c = P^T A P$. A symmetric two–grid iteration for solving $Ax = b$, for a given current iterate x_0, takes the form:

Algorithm 2.1 (Symmetric two–grid algorithm)**.**

- *pre–smooth, i.e., compute*
$$y = x_0 + M^{-1}(b - Ax_0).$$
- *compute coarse–grid correction*
$$x_c = (A^c)^{-1} \, P^T (b - Ay).$$
- *interpolate and update the result*
$$z = y + P x_c.$$
- *post–smooth, i.e., compute*
$$x = z + M^{-T}(b - Az).$$

The mapping $b \mapsto x = B_{TG}^{-1} b$ resulting from Algorithm 2.1 (with $x_0 = 0$) defines a two–grid preconditioner B_{TG}. It is well-known (and readily checked) that its inverse admits the following explicit form

$$B_{TG}^{-1} = (I - M^{-T}A) P \, (A^c)^{-1} \, P^T (I - AM^{-1}) + \overline{M}^{-1}, \tag{1}$$

where $\overline{M} = M(M + M^T - A)^{-1} M^T$ is the so-called symmetrized smoother. For example, if $M = D + L$ (D-diagonal, L-strictly lower triangular) represents the forward Gauss–Seidel method coming from $A = D + L + L^T$, then $\overline{M} = (D + L)D^{-1}(D + L^T)$ gives rise exactly to the symmetric Gauss–Seidel method.

To define a MG algorithm the exact solve with A^c is replaced with a corresponding B_c^{-1} defined by recursion. At the coarsest level B_c typically equals the matrix at that level.

In our setting, to exploit recursion, we need to construct coarse elements (the needed topology relations), coarse dofs and coarse element matrices.

The topology of the agglomerated elements (which serve as coarse elements) is constructed based on the algorithms described in [16]. This part of the setup is independent of the selection of the coarse dofs. The required input here is only the fine–grid "element_face" topological relation.

The coarse dofs at a given level can be identified with the columns of the tentative prolongator Q_c that we construct based on the lower part of the spectrum of the Schur complements S_I associated with each minimal intersection set I.

To build coarse elements (as lists of coarse degrees of freedom), we use the agglomerated elements. Each agglomerated element (as a set of fine dofs) can be split into several minimal intersection sets and each minimal intersection set has one or several coarse dofs (eigenmodes) associated with its respective Schur complement. For each agglomerated element T, the list of coarse degrees of freedom associated with all minimal intersection sets that form T defines the relation "AE_coarse dof". This relation is the coarse counterpart of the (fine-grid) relation "element_dof".

To construct coarse element matrices, we proceed as follows. For each agglomerated element T, we first assemble the local matrix A_T from the (fine–grid) element matrices A_τ, $\tau \subset T$. Also, based on T, we form the submatrix $Q_{T,c}$ of the tentative prolongator Q_c that corresponds to the coarse dofs in T. In what follows, we refer to $Q_{T,c}$ as the "local tentative prolongators". Finally, we construct the coarse element matrix A_T^c based on the local Galerkin relation $A_T^c = Q_{T,c}^T A_T Q_{T,c}$.

2.6. **On the theoretical justification of the spectral AMGe approach.** Consider the orthogonal (in the euclidean inner product) splitting $R^n = V_c \oplus V_f$ where $V_c = Range(Q_c)$ and $V_f = Range^{\perp}(Q_c)$. Let P_{V_f} denote the orthogonal projection onto V_f. It can be proved (see [7]) that the restriction $A_{ff} = P_{V_f} A|_{V_f}$ of A to the subspace V_f is well-conditioned if we choose sufficiently large portion of the eigenmodes (in the lower part of the spectrum) of each Schur complement S_I to form the columns of the tentative prolongator Q_c. That is, larger the spectral tolerance $\tau \in (0, 1)$ better the condition number of the resulting A_{ff}.

We note that the fact that A_{ff} is well-conditioned by itself is not sufficient to conclude that a two–grid method has good convergence factor; we also need some stability property of the interpolation matrix. Since our tentative prolongator is "block–diagonal" its stability properties in A–norm are not very good. That is, the tentative prolongator needs to be improved in general. For example, we can "smooth" it out as in the smoothed aggregation AMG ([15]). This is a feasible option that can lead to better two–grid convergence rates. In our setting though, some of the minimal intersection sets are "thin" aggregates, which led to higher operator complexities. That is why, we did not pursue this option in the present paper. Operator complexity (C) reflects the cost (in terms of arithmetic operations) of one V–cycle. It is defined based on the number of non-zero entries N_l of the l-th level matrix A_l ($l = 0$ is finest level, $l = L$ is the coarsest level) by the formula

$$C = \frac{\sum_{l=0}^{L} N_l}{N_0}. \tag{2}$$

One option that we chose to stabilize the tentative prolongator is based on harmonic extension defined as follows. For each agglomerated element T, we construct Q_c only for minimal intersection sets I that are on the boundary of T (i.e., shared also by other agglomerated elements). Then Q_c defined only on the boundary of T is extended into the interior of T by inverting the block of the matrix A corresponding to the interior of T. More specifically, if A restricted to T has rows $[A_{T,i},\ A_{T,b}]$ corresponding

to the interior of T, then the actual interpolation mapping P has the block form $P = \begin{bmatrix} -A_{T,i}^{-1} A_{T,b}\, Q_c \\ Q_c \end{bmatrix}$.

Another possible approach in the multilevel case is to keep the "unstable" tentative prolongators but compensate for that with more expensive cycles (like W-cycles, for example). In the present paper, we chose to use recursive calls to coarse levels based on preconditioned CG (conjugate gradient) iterations with a preconditioner (defined by recursion) that is a (mildly) nonlinear mapping. This is described in more detail in the next subsection.

2.7. **The nonlinear "nested-CG" AMLI-type multigrid cycles.** The alternative to compensate for a unstable tentative prolongator that we chose is to use more expensive multigrid cycles based on CG (conjugate gradient) inner (between levels) iterations goes back to the nonlinear AMLI–cycle hierarchical basis methods proposed in [1]. Recent analysis of this type of cycles used in MG setting is found in [11].

We implemented an algorithm to be referred to as a "nested-CG" (AMLI) cycle. It consists of CG-acceleration on each level of the currently defined (by recursion from coarse to fine–levels) (nonlinear) preconditioner. More specifically, we run CG iterations for the linear system $Ax = b$ (where A is the finest level matrix) with a preconditioner that is a non-linear mapping. To compute $y = B^{-1}(b)$, we use the following recursive algorithm:

- For a given smoother M, apply a pre-smoothing step using $y = 0$ as initial guess, i.e., compute $y := M^{-1}b$
- Restrict the residual: $r_c := P^T(b - Ay)$
- At the coarsest level solve directly, i.e., compute $y_c{:=}(A^c)^{-1} r_c$; otherwise apply several CG iterations to the linear system $A^c y_c = r_c$, using $y_c = 0$ as initial guess and a preconditioner whose inverse actions B_c^{-1} are defined recursively (by the present algorithm); that is, compute y_c as an approximate solution to $(A^c)^{-1} r_c$.
- Correct y based on y_c as $y := y + Py_c$.
- Based on M^T perform a post-smoothing step, i.e., compute $y := y{+}M^{-T}(b - Ay)$.

Note that the above algorithm defines an inverse of a preconditioner, B^{-1}, that is a *nonlinear* mapping.

In the implementation, we limit the number of inner CG iterations per level so that the complexity of the resulting cycle is kept under control. More specifically, the number of CG iterations at level l is chosen based on the coarsening factor defined as the ratio N_l/N_{l+1}.

3. Adaptivity in the element agglomeration AMGe framework

In the present section we first describe the main steps of the adaptive AMG method from [5]. The method based on its performance improves itself by augmenting its current hierarchy (if any) so that the refined coarse spaces contain more "algebraically smooth" vectors. These vectors represent approximations to the minimal eigenmode of the generalized eigenvalue problem $Ax = \lambda Bx$, where B^{-1} stands for the matrix representation of the V–cycle corresponding to the most current AMG hierarchy. If

no hierarchy is available then B stands for the current level symmetrized smoother $\overline{M}$. The choice of the smoothers is fixed, for example Gauss–Seidel.

We may assume that an initial AMG hierarchy has been built as described in the preceding section. That is, based on an AMGe algorithm, we have created sequence of interpolation matrices, respective (Galerkin) coarse-grid matrices and the associated with them smoothing matrices M (that choice we assume fixed). In the adaptive AMG we begin with performing several "dry runs" based on the current method to measure its performance. If inefficiency is detected then we try to improve the method by *adapting the AMG hierarchy*. Some preliminary results of an adaptive AMG method (in AMGe framework) were reported in [17]. We describe, in what follows, the particular version of the approach that we have implemented.

For simplicity, we start with a two-level method. In our setting, it comes with a tentative prolongator Q_c, an "improved" interpolation mapping P, the coarse-grid matrix $A^c = P^T A P$ and a smoother M associated with A. These components define a two-level preconditioner B (as in Algorithm 2.1 or formula (1)). Note that the case $Q_c = 0$ and $B = \overline{M}$ is treated similarly (with obvious modification). This case is refereed to as adaptive AMG starting "from scratch".

Then we perform several stationary iterations to solve $Ax = 0$ with a random initial guess x, i.e.,

$$x := \left(I - B^{-1}A\right) x,$$

where I is the identity matrix. At every iteration step we monitor the convergence rate, i.e., we compute

$$\rho^2 = \frac{x^T A x}{x_{old}^T A x_{old}}.$$

If after few (for example, five) iterations ρ is greater than a desired convergence factor, we stop the iteration. Our current (two–level) solver B cannot efficiently handle this vector, i.e., x is rich in eigenmode components in the lower part of the spectrum of $B^{-1}A$. We refer to x as an "algebraically smooth" vector. Note that x is "algebraically smooth" with respect to the current (two-grid) preconditioner B (which initially may be simply the smoother $\overline{M}$). The iteration process above can be interpreted as calculating approximations to the highest eigenmode of the A-symmetric two–level iteration matrix $E = I - B^{-1}A$. Since we consider symmetric MG cycles and A-convergent smoothers (like Gauss–Seidel), this implies that AE is symmetric positive semi–definite.

The next step in the adaptive AMG, is to incorporate the "algebraically smooth" vector x in the two-level hierarchy by changing the tentative prolongator Q_c and afterwards, the respective "improved" interpolation matrix P. After a new P has been computed we re-compute the coarse-grid matrix $A^c = P^T A P$.

The following algorithm implements the above steps:

Algorithm 3.1 (Augmenting coarse spaces based on additional vectors)**.**

- *We compute the interpolation error $e = x - Q_c Q_c^T x$. Since Q_c has mutually orthogonal columns, e is the projection of x onto the orthogonal complement of the span of the columns of Q_c.*

- *For each minimal intersection set I, consider e_I (the restriction of e to I). Then, if $\|e_I\|/\|x\|$ is greater than some given threshold, we add e_I (extended by zeros outside I) as an extra column to the tentative prolongator Q_c. If $\|e_I\|/\|x\|$ is less than the threshold, we discard e_I.*
- *At the end, compute P, the improved interpolation matrix from Q_c using harmonic extension as described in section 2.6. Using this updated P, we recompute $A^c = P^T A P$.*

In the multilevel case, we have to perform few additional steps at any given level. Note that below the current coarse level, we may already have a hierarchy of coarse spaces. Since we have augmented Q_c with more columns and built a new P, we have generally changed the dimension of the coarse space at the given level. Hence, to relate the next level coarse space with the current level coarse space based on the previously available interpolation matrix P_{next} is not possible because the dimensions do not match. To fix this problem, we need to add some additional rows to P_{next} to match the dimensions. The new rows are simply set to zero. Thus the second coarse matrix does not have to be recomputed. This is seen as follows. Let Q_c^{old} be the current level tentative prolongator. We add few new columns to Q_c^{old} based on the vector e computed in Algorithm 3.1. That is, we have $Q_c = [Q_c^{new},\ Q_c^{old}]$. Similarly, the old interpolation matrix P^{old} gets updated with the same number of columns; i.e., we have $P = [P^{new},\ P^{old}]$. The new coarse matrix is $A^c = P^T A P$. The second coarse level interpolation matrix P_{next}^{old} gets modified with some extra zero rows. It becomes $P_{next} = \begin{bmatrix} 0 \\ P_{next}^{old} \end{bmatrix}$. Therefore, the next coarse level matrix equals

$$\begin{aligned} A_{next}^c = P_{next}^T A^c P_{next} &= \begin{bmatrix} 0 \\ P_{next}^{old} \end{bmatrix}^T [P^{new},\ P^{old}]^T A [P^{new},\ P^{old}] \begin{bmatrix} 0 \\ P_{next}^{old} \end{bmatrix} \\ &= (P_{next}^{old})^T (P^{old})^T A P^{old} P_{next}^{old}, \end{aligned}$$

which is the expression defining the next level old coarse matrix.

A second problem that needs to be fixed is to augment the minimal intersection sets at the given coarse level since we have added some new coarse dofs there. To accomplish this task, during the setup phase of the algorithm (more specifically, when we agglomerate elements, build the minimal intersection sets, etc.) we keep a hierarchy structure of the minimal intersection sets. We say that a level l minimal intersection set I_l is associated with a coarser level $l+1$ minimal intersection set I_{l+1} if there exists a coarse dof d_{l+1} from I_{l+1} associated with I_l. Recall that (initially) every coarse dof corresponds to an eigenvector of a local matrix associated with a minimal intersection set. It is easily seen that for each I_l there exists a unique coarse minimal intersection set I_{l+1} that is associated with I_l. Based on the hierarchy of the minimal intersection sets, we distribute the newly created coarse dofs among the coarser minimal intersection sets. Then we proceed by recursion. The previously current coarse level becomes fine and we apply the method starting from that level. The initial vector x is now not random; it equals $Q^T x$ where x was the vector computed at the previous (fine) level and Q is the augmented tentative prolongator at that level. The same scheme applies to all subsequent levels till we end up with two levels only. Then, we apply the two-level scheme described above.

The following "monotonicity" result demonstrates the fact that augmenting a current coarse space (in the sense described above) leads to a better two-grid method.

Proposition 3.1. *Let A, M, P^{old} and $A^c_{old} = \left(P^{old}\right)^T A \left(P^{old}\right)$ define a current two–grid preconditioner B_{old} (as in (1)). Augment the coarse space so that $P = \left[P^{new},\ P^{old}\right]$ is the new interpolation matrix and $A^c = P^T A P$ is the new coarse matrix. Then A, M, P and A^c define the new two–grid preconditioner B (also as in (1)). Then the following inequalities hold:*

$$v^T A^{-1} v \geq v^T B^{-1} v \geq v^T B_{old}^{-1} v.$$

That is, B^{-1} provides a more accurate approximate inverse to A than the old two–grid preconditioner B_{old}^{-1}.

Proof. Let $(u,v)_A = v^T A u$ denote the A–inner product (note that A is s.p.d.). It is equivalent to show that for all u

$$0 \leq \left(\left(I - B^{-1}A\right)u, u\right)_A \leq \left(\left(I - B_{old}^{-1}A\right)u, u\right)_A. \tag{3}$$

The symmetric two-level cycle leads to an error propagation matrix that admits the following product form

$$I - B^{-1}A = \left(I - M^{-T}A\right)\left(I - P\left(A^c\right)^{-1}P^T A\right)\left(I - M^{-1}A\right)$$

and similarly

$$I - B_{old}^{-1}A = \left(I - M^{-T}A\right)\left(I - P^{old}\left(A^c_{old}\right)^{-1}\left(P^{old}\right)^T A\right)\left(I - M^{-1}A\right).$$

We also note that $(I - M^{-T}A)$ is A-adjoint to $(I - M^{-1}A)$. Thus (3) with $v = (I - M^{-1}A)u$ reduces to

$$\begin{aligned} 0 &\leq \left(\left(I - P\left(A^c\right)^{-1}P^T A\right)v, v\right)_A \\ &\leq \left(\left(I - P^{old}\left(A^c_{old}\right)^{-1}\left(P^{old}\right)^T A\right)v, v\right)_A. \end{aligned}$$

These inequalities are readily seen from the fact that both $\left(I - P\left(A^c\right)^{-1}P^T A\right)$ and $\left(I - P^{old}\left(A^c_{old}\right)^{-1}\left(P^{old}\right)^T A\right)$ are A-projectors providing the best approximation in the A–norm from two nested spaces Range $\left[P^{new},\ P^{old}\right]$ and Range (P^{old}). □

4. Numerical results for spectral agglomerate AMGe

In this and subsequent sections we illustrate the performance of the methods discussed in the paper. The problems we consider come from finite element (f.e.) discretization of 2D anisotropic diffusion as well as 2D (indefinite) Helmholtz equation rewritten as a first order (mixed) system casted in a least-squares form (the so-called FOSLS formulation). The latter problem gives rise to three degrees of freedom per node.

The specific PDEs, posed in $\Omega = (0,1)^2$, read:

- "anisotropic" diffusion:

$$-\text{div}\,(\epsilon I + \underline{bb}^T)\nabla p = f, \quad p = 0 \text{ on } \partial\Omega,$$

where ϵ is either 1 (which corresponds to no anisotropy, i.e. the operator is Laplacian if $\underline{b} = 0$) or 0.001 and $\underline{b}$ is a given constant vector in $\mathbb{R}^2$. Here we use f.e. discretization based on standard piecewise linear f.e. space S_h over a given triangular mesh $\mathcal{T}_h$ of Ω.

- Helmholtz equation $-\Delta p - k^2 p = f$, $p = 0$ on $\partial\Omega$ for a given k^2. The FOSLS formulation reads: Compute

$$\|\text{curl } \underline{u}_h\|^2 + \|\underline{u}_h - \nabla p\|^2 + \|f + \text{div } \underline{u}_h + k^2 p_h\|^2 \mapsto \min,$$

over $\underline{u}_h \in (S_h)^2$ and $p_h \in S_h$, where S_h is a finite element space of piecewise linear functions over a given triangular mesh $\mathcal{T}_h$ on Ω. Here $\|.\|$ stands for the $L_2(\Omega)$ norm. p_h and the tangential component of $\underline{u}_h$ are set to zero on $\partial\Omega$. The additional vector unknown $\underline{u}_h$ approximates ∇p.

In our tests, we varied the spectral tolerance $\tau \in [0, 1)$ (see Section 2.4) as well as the smoothers used: Gauss–Seidel ("GS") and its block–version. We used (overlapping) blocks referring to the agglomerated elements (viewed as sets of fine–grid dofs). This block smoother is denoted in the tables by "AE-BlockGS". We used two types of multigrid cycles: the standard symmetric $V(1,1)$–cycle (forward (block) Gauss–Seidel in fine-to-coarse direction and backward (block) Gauss–Seidel in coarse-to-fine direction) and the "nested CG" cycle described in Section 2.7. For each method and problem, we compare its two–level and multilevel versions. To reduce the complexity of the methods we use different coarsening factors ("Crs. f."), one at the initial level (equal to sixteen or eight) and another one (equal to four) at all remaining coarse levels. The number of levels used is denoted by $\mathrm{N_{lev}}$. We also list ϱ that is an estimate of the convergence factor. In most of the tables the resulting multigrid cycle is used as a preconditioner in the CG (conjugate gradient) method. In addition, the convergence history of the methods is illustrated graphically. Finally, some typical graphs of the coarsest level basis functions illustrating the kind of interpolation matrices (their columns viewed on the finest grid) that result from a particular algorithm are shown. The tables also contain the operator complexity C defined in (2) as well as a related quantity C_{A+P} defined as:

$$C_{A+P} = C + \frac{\sum_{l=0}^{L-1} \text{nnz}\,(P_l)}{\text{nnz}(A_0)}.$$

That is, to the commonly used operator complexity C we add the total number of non-zero entries of all interpolation matrices P_l divided by the number of non-zeros of the finest level matrix.

In our numerical experiments, we use 2 types of meshes: a structured "square" mesh and unstructured triangular meshes. A given uniform "square" mesh produces the actual structured mesh by subdividing each square cell using its diagonal into two triangles. In the case of structured grid, we choose the anisotropy vector $\underline{b}$ aligned with the grid. All unstructured meshes we consider are obtained by refinement of the mesh shown in Figs. 1-2.

In this particular section, we use V-cycle iterations based on the so–called (fully) harmonic interpolation matrices. These interpolation matrices are obtained by first selecting coarse dofs based on a portion of the lower part of the spectrum of the Schur

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ
8 4	0	1.4	1.7	2	GS	0.22
8 4	0	1.5	2.0	5	GS	0.23

TABLE 1. 2d Laplacian on square grid. 2048 elements, 1089 dofs. "Harmonic" prolongator is used inside V-cycle. No CG acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ
8 4	0	1.3	1.7	2	GS	0.24
8 4	0	1.4	1.9	7	GS	0.24

TABLE 2. 2d Laplacian on square grid. 32768 elements, 16641 dofs. "Harmonic" prolongator is used inside V-cycle. No CG acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
16 4	0	1.5	1.9	2	GS	0.15	4/1
16 4	0	1.6	2.2	6	GS	0.18	4/2

TABLE 3. 2d Laplacian on unstructured grid, 6400 elements, 3321 dofs. "Harmonic" prolongator is used inside V-cycle. Conjugate gradient acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
16 4	0	1.6	2.2	7	GS	0.21	4/3
16 4	0	1.5	1.9	2	GS	0.18	4/4

TABLE 4. 2d Laplacian on unstructured grid, 25600 elements, 13041 dofs. "Harmonic" prolongator is used inside V-cycle. Conjugate gradient acceleration is used.

complements S_I associated with each minimal intersection set I (as described previously) leading to an orthogonal matrix Q_c, Based on the remaining part of the spectrum we can also construct the complementary (also orthogonal) matrix Q_f. Then, we first perform a change of variables that leads to a matrix $[Q_f,\ Q_c]^T A[Q_f,\ Q_c] = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}$. Note that similar structure is obtained for all neighborhood matrices $A_N(I)$ (discussed in Section 2.4). Having the 2×2 block form of the transformed matrix with the coarse dofs identified (denoted by "c" index) the actual interpolation matrix is constructed as described in [9]. The resulting interpolation matrix P can be viewed as a locally "harmonic" extension of the tentative prolongator Q_c. This was the construction used also in [7].

Numerical results for this section are found in tables 1, 2, 3, 4, 5, 6, 7, 8. Figures 4, 7, 8 contain convergence plots for runs in the tables listed above. Figures 3, 5, 6, 9 contain graphs of some coarsest basis functions.

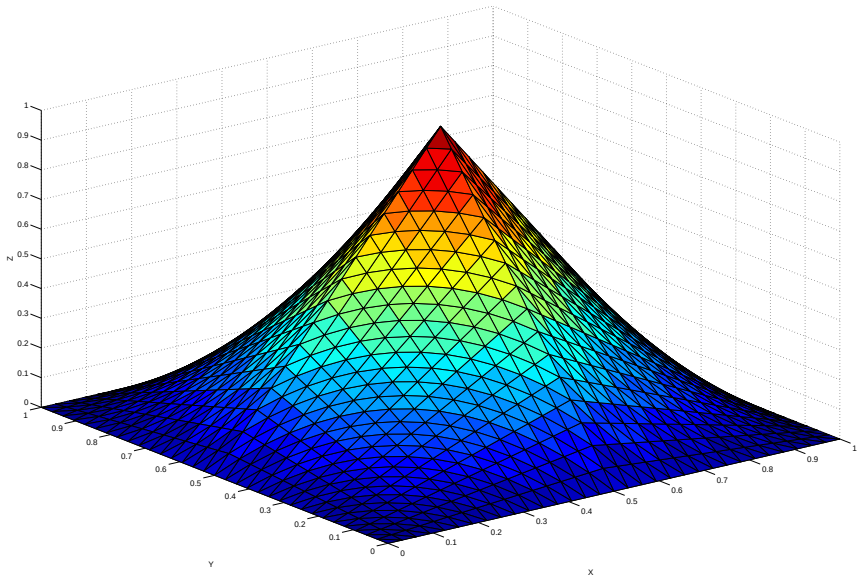

FIGURE 3. One of coarsest-grid basis functions. Laplacian on square grid, 2048 elements, 1089 dofs. "Fully-harmonic" prolongator is used.

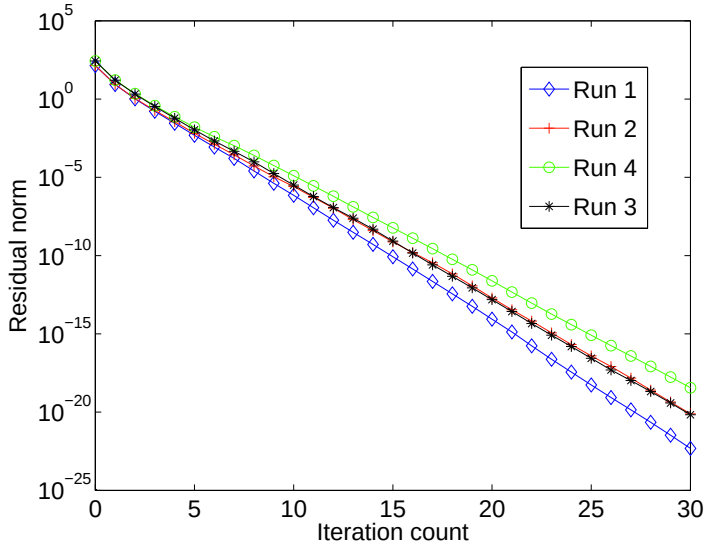

FIGURE 4. Residual decrease history for some runs in tables 3, 4.

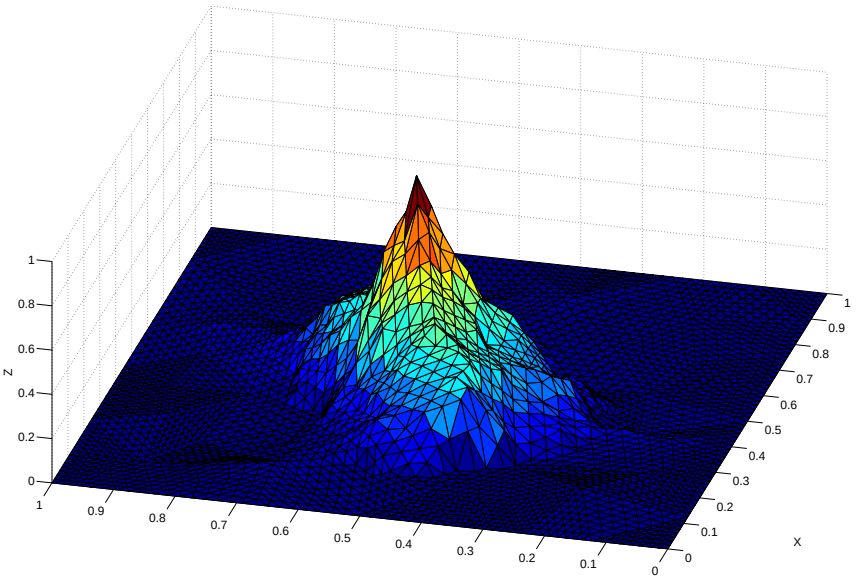

FIGURE 5. One of coarsest-grid basis functions. Laplacian on unstructured grid, 6400 elements, 3321 dofs. "Fully-harmonic" prolongator is used. Coarsest grid contains 8 elements.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ
8 4	0.05	2.0	2.4	2	GS	0.27
8 4	0.05	3.9	6.0	5	GS	0.55
8 4	0.05	2.0	2.4	2	AE-BlockGS	0.11
8 4	0.05	3.9	6.0	5	AE-BlockGS	0.11

TABLE 5. 2d anisotropic diffusion on square grid. Anisotropy is grid-aligned. 2048 elements, 1089 dofs. "Harmonic" prolongator is used inside V-cycle. No CG acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ
8 4	0.05	1.9	2.3	2	GS	0.27
8 4	0.05	4.3	6.8	7	GS	0.74
8 4	0.05	1.9	2.3	2	AE-BlockGS	0.11
8 4	0.05	4.3	6.8	7	AE-BlockGS	0.11

TABLE 6. 2d anisotropic diffusion on square grid. Anisotropy is grid-aligned. 32768 elements, 16641 dofs. "Harmonic" prolongator is used inside V-cycle. No CG acceleration is used.

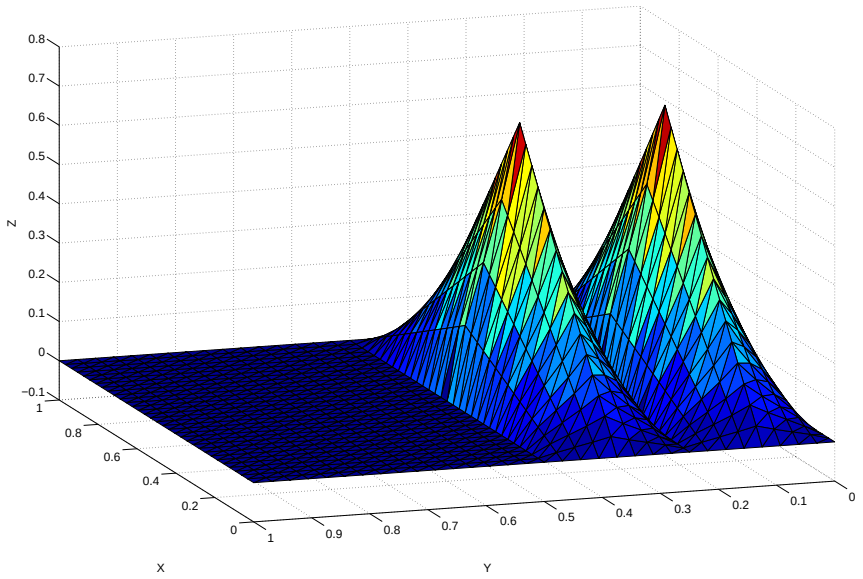

FIGURE 6. One of coarsest-grid basis functions. Anisotropic diffusion on square grid, 2048 elements, 1089 dofs. "Fully-harmonic" prolongator is used. Note that this is one function, not sum of two functions.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
16 4	0.15	2.0	2.5	2	AE-BlockGS	0.13	7/1
16 4	0.15	4.0	5.9	7	AE-BlockGS	0.16	7/2
16 4	0	1.5	1.9	2	AE-BlockGS	0.61	8/1
16 4	0	1.6	2.2	7	AE-BlockGS	0.62	8/2

TABLE 7. 2d anisotropic diffusion on unstructured grid, 25600 elements, 13041 dofs. "Harmonic" prolongator is used inside V-cycle. Conjugate gradient acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
16 4	0.15	2.0	2.5	2	AE-BlockGS	0.12	7/3
16 4	0.15	3.5	5.2	6	AE-BlockGS	0.14	7/4
16 4	0	1.5	1.9	2	AE-BlockGS	0.44	8/3
16 4	0	1.6	2.2	6	AE-BlockGS	0.44	8/4

TABLE 8. 2d anisotropic diffusion on unstructured grid, 6400 elements, 3321 dofs. "Harmonic" prolongator is used inside V-cycle. Conjugate gradient acceleration is used.

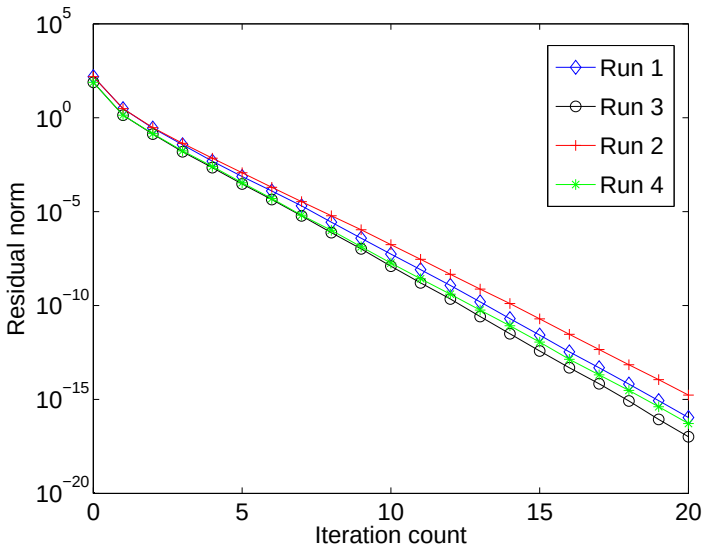

FIGURE 7. Residual decrease history for some runs in tables 8, 7. In these runs, additional spectral coarse dofs were used ($\tau = 0.15$).

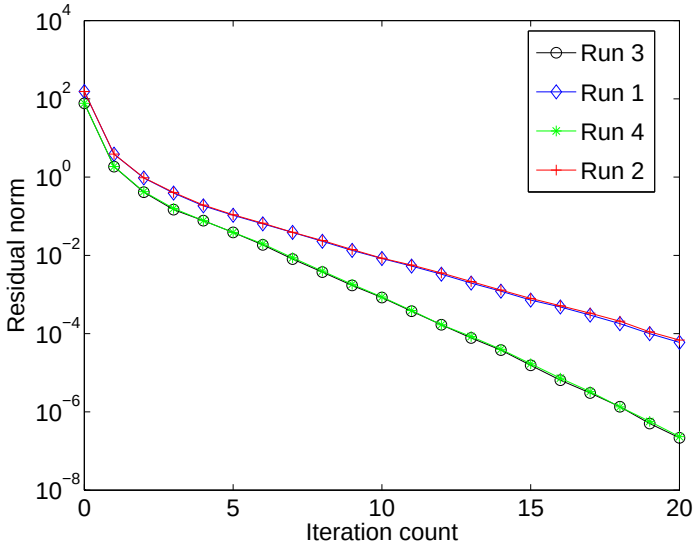

FIGURE 8. Residual decrease history for some runs in tables 8, 7. In these runs, additional spectral coarse dofs were NOT used ($\tau = 0$).

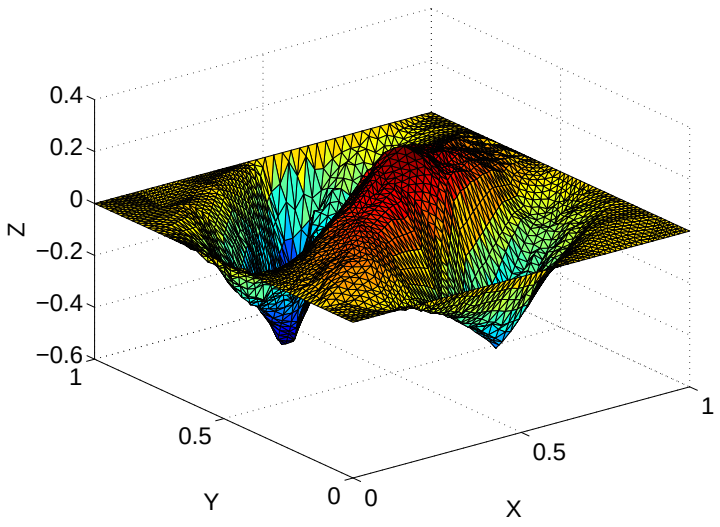

FIGURE 9. One of coarsest-grid basis functions. Anisotropic diffusion on unstructured grid, 6400 elements, 3321 dofs. "Fully-harmonic" prolongator is used. Coarsest grid contains 2 elements. This function corresponds to additional "spectral" coarse dofs.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
32 4	0	1.2	1.4	2	GS	0.21	10/1
32 4	0	1.3	1.5	4	GS	0.21	10/2

TABLE 9. 2d Laplacian on square grid, 2048 elements, 1089 dofs. "Tentative" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
32 4	0	1.2	1.4	2	GS	0.22	10/3
32 4	0	1.3	1.5	6	GS	0.22	10/4

TABLE 10. 2d Laplacian on square grid, 32768 elements, 16641 dofs. "Tentative" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

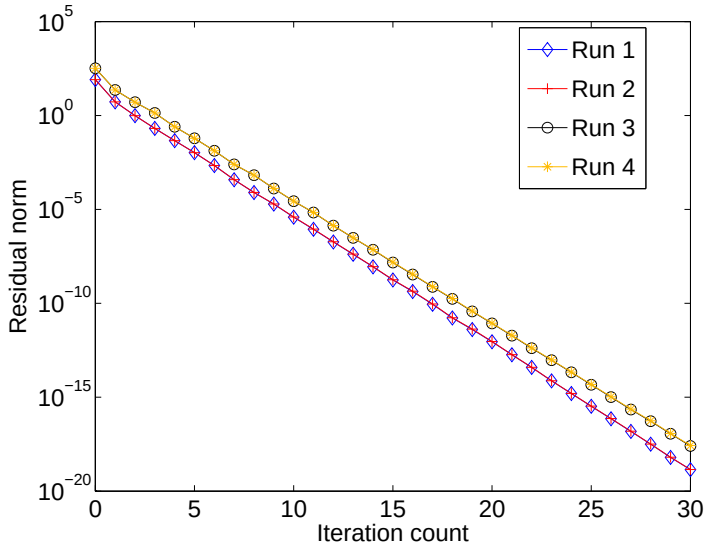

FIGURE 10. Residual decrease history for some runs in tables 9, 10.

5. TESTS WITH TENTATIVE PROLONGATORS WITHIN NESTED CG CYCLES

In this section we illustrate the performance of the method when tentative (i.e. block-diagonal and orthogonal) prolongators (Q_c) are used. In order to compensate for prolongator instability, we use "nested CG" cycles described in Section 2.7. We report in this section test results for Laplacian and anisotropic diffusion. We use (pointwise) Gauss-Seidel as a smoother here, since our cycle is now more robust (and expensive). Numerical results for this section are found in tables 9, 10, 11, 12, 13, 14. Figures 10, 12, 14 contain convergence plots for runs in the tables listed above. Figures 11, 13, 15 show graphs of some of the resulting coarsest basis functions.

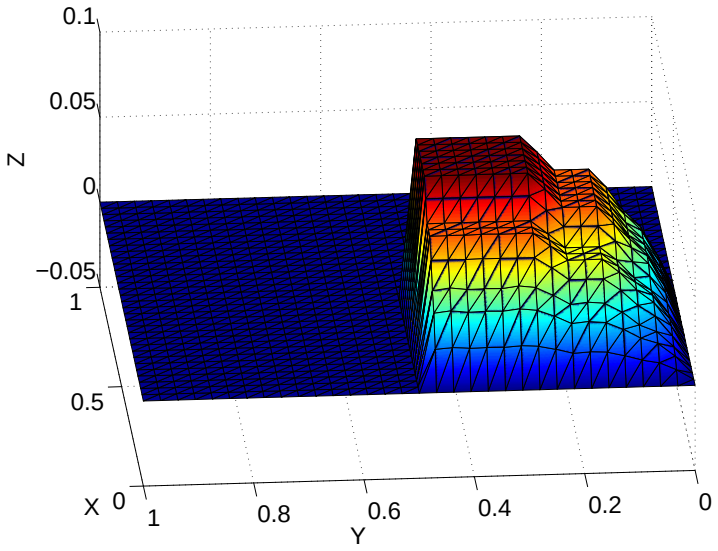

FIGURE 11. One of coarsest-grid basis functions. 2d Laplacian on square grid, 2048 elements, 1089 dofs. "Tentative" prolongator is used. Coarsest grid contains 4 elements.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
32 4	0	1.4	1.5	2	GS	0.31	12/1
32 4	0	1.5	1.7	5	GS	0.32	12/2

TABLE 11. 2d Laplacian on unstructured grid, 6400 elements, 3321 dofs. "Tentative" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
32 4	0	1.4	1.5	2	GS	0.37	12/3
32 4	0	1.5	1.7	6	GS	0.42	12/4

TABLE 12. 2d Laplacian on unstructured grid, 25600 elements, 13041 dofs. "Tentative" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

6. Adaptive AMG tests

Here we present numerical results for adaptive AMG discussed in Section 3. We apply this method to the Helmholtz problem described in Section 4, with k^2 varying from 0 (which is just Laplace operator) to 200. In our tests, we use the partially harmonic extension of the tentative prolongators (as explained in Section 2.6) to define the actual interpolation matrices P_l used within the adaptive V-cycle AMG. We also employ nested–CG cycles (described in Section 2.7).

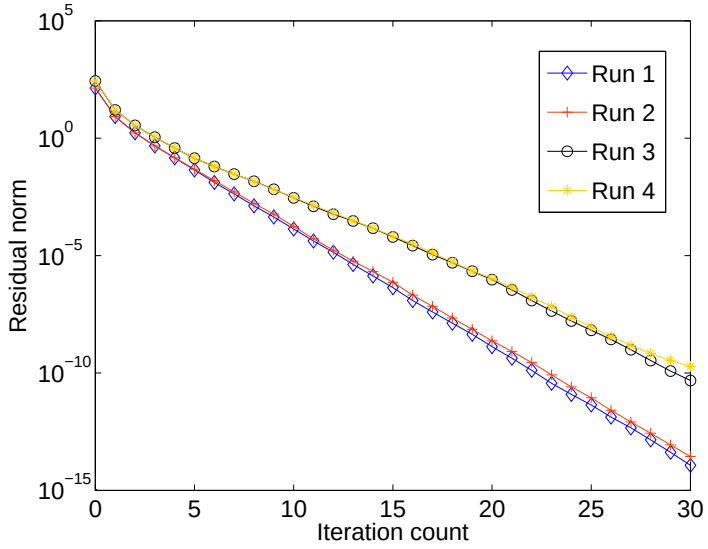

FIGURE 12. Residual decrease history for some runs in tables 11, 12.

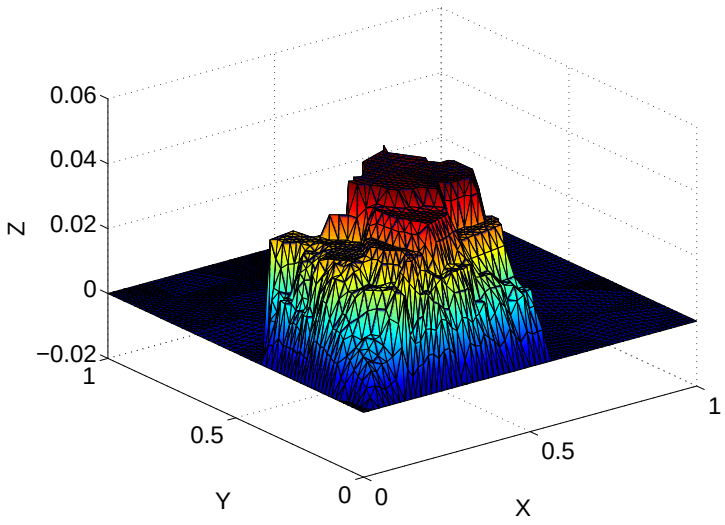

FIGURE 13. One of coarsest-grid basis functions. 2d Laplacian on unstructured grid, 6400 elements, 3321 dofs. "Tentative" prolongator is used. Coarsest grid contains 4 elements.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
32 4	0.05	2.0	2.2	2	GS	0.40	14/1
32 4	0.05	2.7	3.5	5	GS	0.44	14/2

TABLE 13. 2d anisotropic diffusion on unstructured grid, 6400 elements, 3321 dofs. "Tentative" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	τ	C_A	C_{A+P}	N_{lev}	Smoother	ρ	Fig/Run
32 4	0.05	2.1	2.4	2	GS	0.43	14/3
32 4	0.05	3.3	4.4	6	GS	0.46	14/4

TABLE 14. 2d anisotropic diffusion on unstructured grid, 25600 elements, 13041 dofs. "Tentative" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

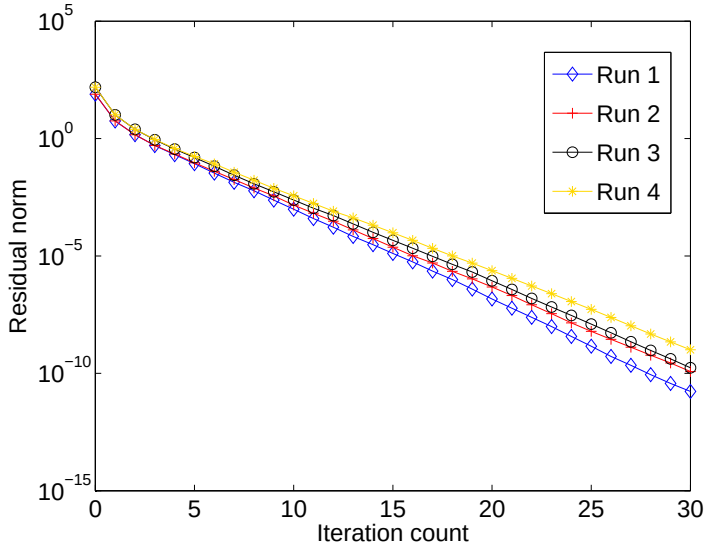

FIGURE 14. Residual decrease history for some runs in tables 13, 14.

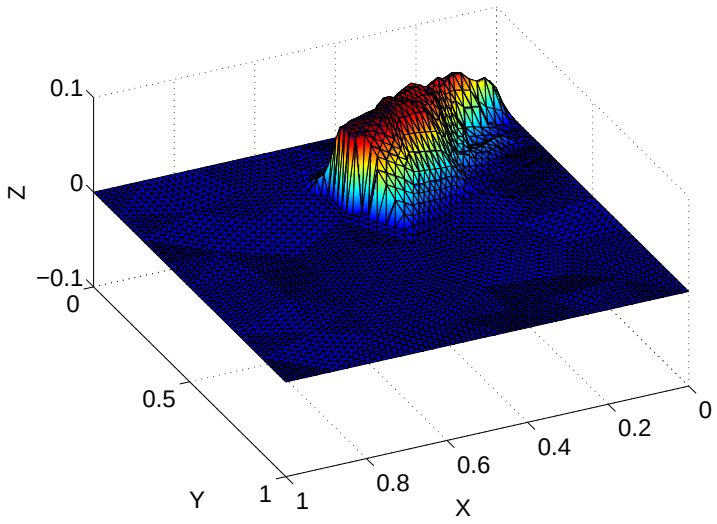

FIGURE 15. One of coarsest-grid basis functions. 2d anisotropic diffusion on unstructured grid, 25600 elements, 13041 dofs. "Tentative" prolongator is used. Coarsest grid contains 4 elements.

Crs. f.	C_A	C_{A+P}	N_{lev}	Smoother	N_{vec}	ρ	Fig/Run
32 4	2.2	3.1	2	SymGS	6	0.10	16/1
32 4	2.7	4.1	5	SymGS	6	0.11	16/2

TABLE 15. Adaptivity: FOSLS Helmholtz, $k^2 = 0$, curl penalty=1, 6400 elements, 9963 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	C_A	C_{A+P}	N_{lev}	Smoother	N_{vec}	ρ	Fig/Run
32 4	2.4	3.3	2	SymGS	7	0.11	16/3
32 4	3.2	4.8	6	SymGS	7	0.10	16/4

TABLE 16. Adaptivity: FOSLS Helmholtz, $k^2 = 0$, curl penalty=1, 25600 elements, 39123 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	C_A	C_{A+P}	N_{lev}	Smoother	N_{vec}	ρ	Fig/Run
32 4	2.6	3.7	2	SymGS	10	0.11	17/1
32 4	3.6	5.5	5	SymGS	10	0.11	17/2

TABLE 17. Adaptivity: FOSLS Helmholtz, $k^2 = 100$, curl penalty=1, 6400 elements, 9963 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	C_A	C_{A+P}	N_{lev}	Smoother	N_{vec}	ρ	Fig/Run
32 4	2.6	3.7	2	SymGS	10	0.10	17/3
32 4	4.0	6.1	6	SymGS	11	0.13	17/4

TABLE 18. Adaptivity: FOSLS Helmholtz, $k^2 = 100$, curl penalty=1, 25600 elements, 39123 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Everywhere adaptivity is done from scratch (i.e., no initial hierarchy is assumed, hence adaptivity is based initially on the symmetrized Gauss-Seidel smoother). While obtaining "smooth vectors" for adaptivity, we do not use CG at the finest level, i.e., we run the "nested-CG" cycle as a stationary iteration. However, we do use CG in the final test runs. The number of "smooth vectors" used to build the final AMG hierarchy is shown in the column "N_{vec}" of each table.

Numerical results for this section are found in tables 15, 16, 17, 18, 19, 20 . Figures 16, 17, 18 contain convergence plots for runs shown in the above tables. Figures 19, 20, 21 show graphs of some of the resulting coarsest basis functions. Finally, in Figure 22, we show how the complexity and convergence rate vary with the number of incorporated "algebraically smooth" vectors.

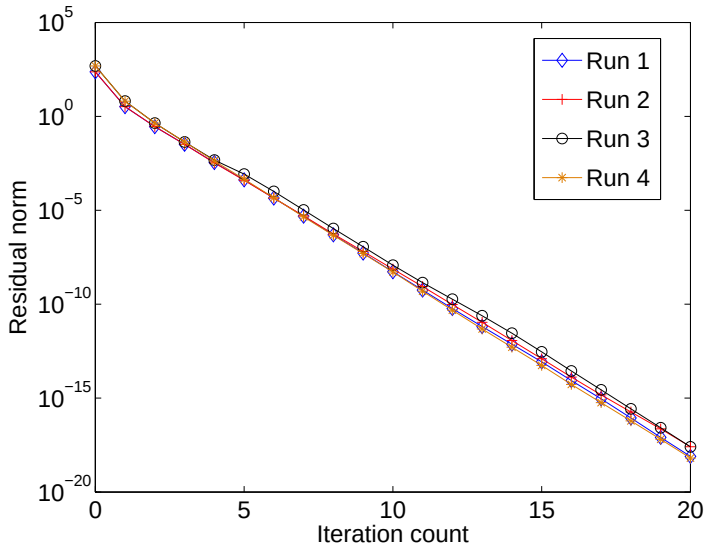

FIGURE 16. Residual decrease history for some runs in tables 15, 16.

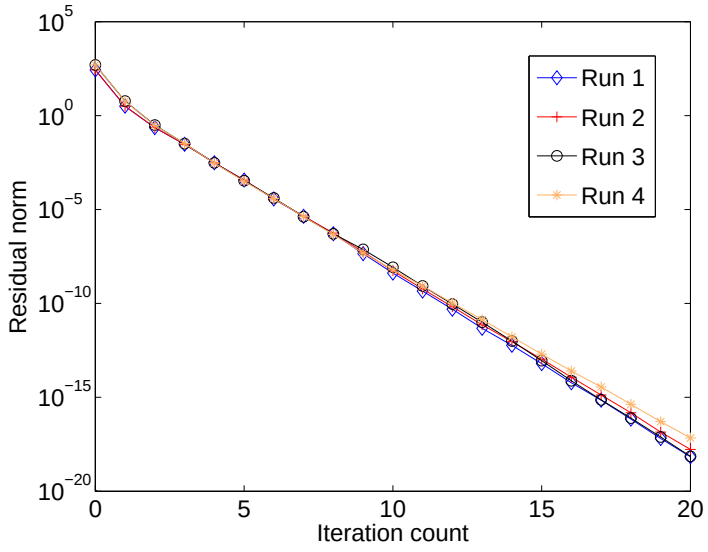

FIGURE 17. Residual decrease history for some runs in tables 17, 18.

Crs. f.	C_A	C_{A+P}	N_{lev}	Smoother	N_{vec}	ρ	Fig/Run
32 4	2.5	3.6	2	SymGS	9	0.13	18/1
32 4	3.6	5.5	5	SymGS	10	0.18	18/2

TABLE 19. Adaptivity: FOSLS Helmholtz, $k^2 = 200$, curl penalty=1, 6400 elements, 9963 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

Crs. f.	C_A	C_{A+P}	N_{lev}	Smoother	N_{vec}	ρ	Fig/Run
32 4	2.7	3.7	2	SymGS	10	0.10	18/3
32 4	4.1	6.2	6	SymGS	11	0.25	18/4

TABLE 20. Adaptivity: FOSLS Helmholtz, $k^2 = 200$, curl penalty=1, 25600 elements, 39123 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Conjugate gradient acceleration is used.

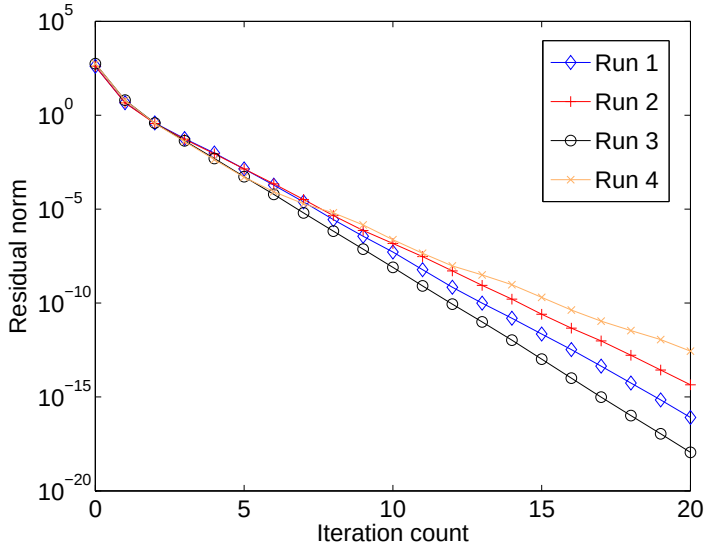

FIGURE 18. Residual decrease history for some runs in tables 19, 20.

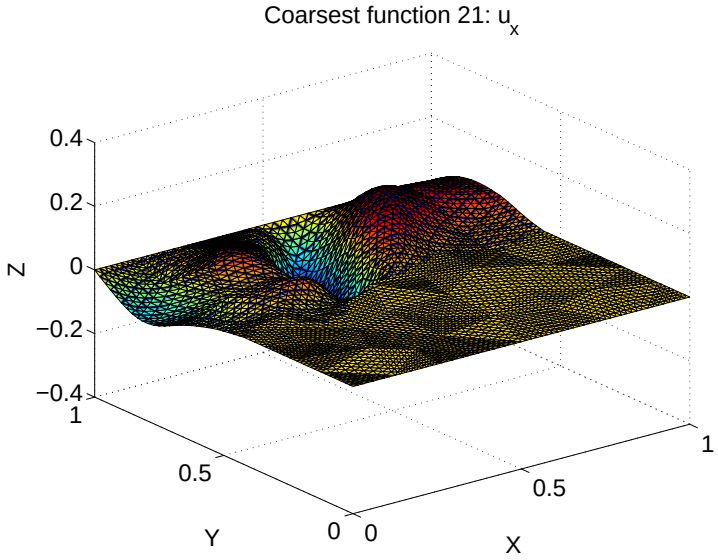

FIGURE 19. One of coarsest-grid basis functions, u_x component. Adaptive AMG for FOSLS Helmholtz, $k^2 = 200$, 25600 elements, 39123 dofs. "Partially-harmonic" prolongator is used.

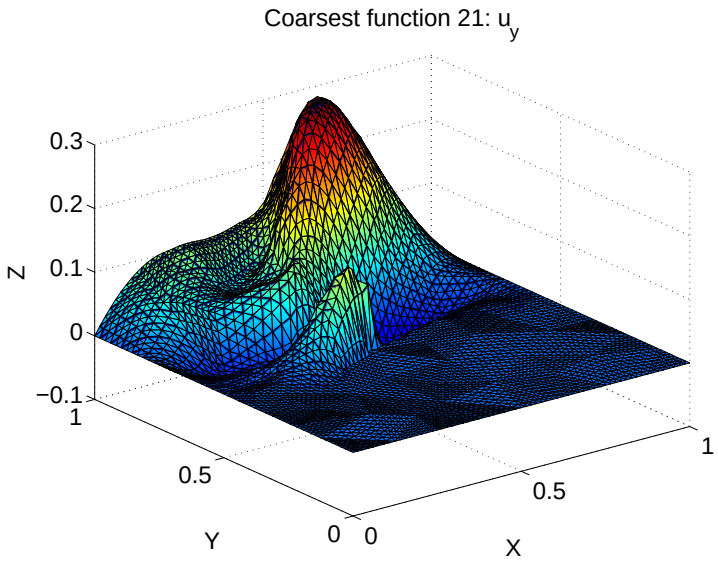

FIGURE 20. One of coarsest-grid basis functions, u_y component. Adaptive AMG for FOSLS Helmholtz, $k^2 = 200$, 25600 elements, 39123 dofs. "Partially-harmonic" prolongator is used.

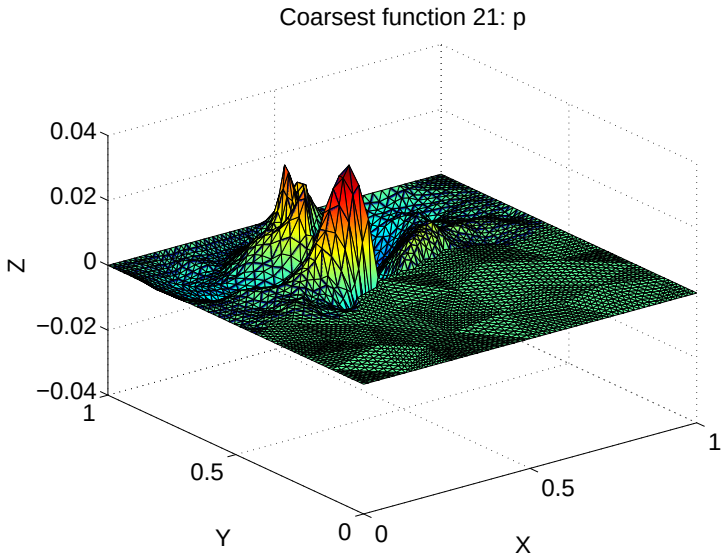

FIGURE 21. One of coarsest-grid basis functions, p component. Adaptive AMG for FOSLS Helmholtz, $k^2 = 200$, 25600 elements, 39123 dofs. "Partially-harmonic" prolongator is used.

CONCLUSIONS

We have implemented a version of the spectral agglomerate AMGe method that leads in a natural way to (orthogonal) tentative prolongation matrices. These tentative prolongators can then be used to construct more stable interpolation matrices

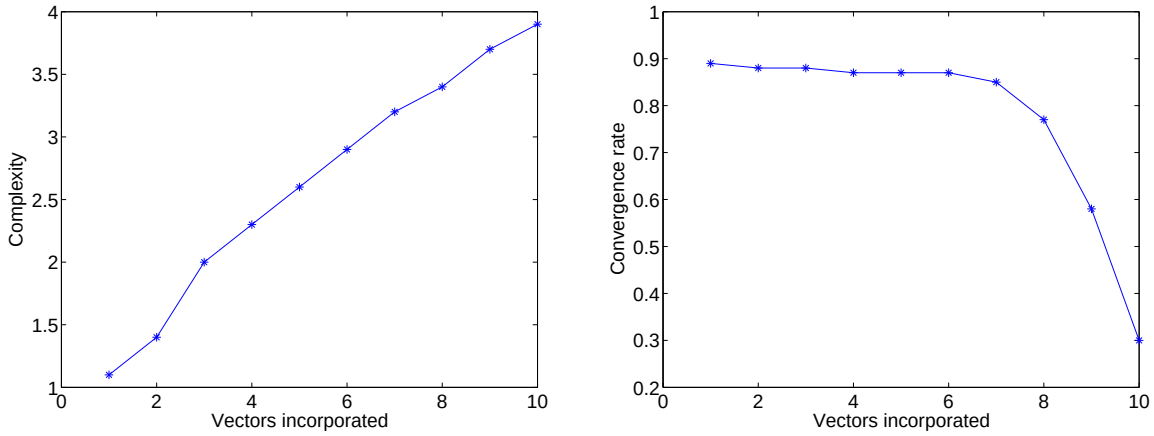

FIGURE 22. Adaptivity: FOSLS Helmholtz, $k^2 = 200$, 6400 elements, 9963 dofs. "Partially-harmonic" prolongator is used inside "nested CG cycle". Complexity and convergence rate depending on the number of incorporated vectors. The convergence rate is for the *stationary* method.

by harmonic extension (in the interior of the agglomerated elements) that results in better convergence properties of standard V-cycle MG. The spectral choice of coarse dofs by itself defines a scale of AMG methods that can get more powerful (with the expense of increasingly higher operator complexity). Another alternative that we explored was to keep the tentative prolongators in the computation and compensate for their poor stability by using more expensive MG cycles; we considered the "nested CG cycle" that gives rise to (mildly) non-linear MG mappings. Finally, we note that a low tolerance spectral agglomerate AMGe can be used to initiate an adaptive AMG cycle. Once the initial AMG hierarchy has been constructed the individual element matrices are no longer needed for the adaptive AMG we constructed. We chose, in the numerical tests, to construct the initial AMG hierarchy "from scratch" which version completely eliminated the need for the fine–grid element matrices. This particular method though did utilize the fine-grid element topology relations needed to construct the multilevel agglomerates and respective minimal intersection sets. All approaches offer potential to generate better (in convergence properties) AMG methods. The approach based on tentative prolongators (and more expensive MG cycles) has the fastest setup among all, however one cycle is more expensive (in setup and/or in cost per cycle). In general, all approaches can get fairly expensive if we want to get too small convergence factors. We have tried to demonstrate their performance on a variety of test problems including scalar anisotropic diffusion as well as Helmholtz problem in a SPD (FOSLS) formulation.

References

[1] Owe Axelsson and Panayot S. Vassilevski. Variable-step multilevel preconditioning methods. *Numer. Lin. Alg. Appl.*, 1:75–101, 1994.

[2] A. Brandt, S. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations. Technical report, Inst. for Computational Studies, Fort Collins, Colorado, October 1982.

[3] A. Brandt, S. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse equations. In D. J. Evans, editor, *Sparsity and its applications (Loughborough 1983)*, pages 257–284. Cambridge Univ. Press, Cambridge, 1985.

[4] 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(5):1570–1592 (electronic), 2000.

[5] M. Brezina, R. D. Falgout, S. MacLachlan, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. Adaptive smoothed aggregation (αSA). *SIAM J. Sci. Comput.*, 25:1896–1920 (electronic), 2004.

[6] 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(1):1–26 (electronic), 2003.

[7] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, P. S. Vassilevski, T. Manteuffel, S. McCormick, and J. Ruge. Spectral element agglomerate AMGe. In *Proceedings of the 16th International Conference on Domain Decomposition Methods*, pages 513–521, 2007.

[8] Van Emden Henson and Panayot S. Vassilevski. Element-free AMGe: general algorithms for computing interpolation weights in AMG. *SIAM J. Sci. Comput.*, 23(2):629–650 (electronic), 2001. Copper Mountain Conference (2000).

[9] Jim E. Jones and Panayot S. Vassilevski. AMGe based on element agglomeration. *SIAM J. Sci. Comput.*, 23(1):109–133 (electronic), 2001.

[10] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392 (electronic), 1998.

[11] Yvan Notay and Panayot S. Vassilevski. Recursive krylov-based multigrid cycles. *Numer. Lin. Alg. Appl.*, 14:(in press), 2007.

[12] J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, volume 3 of *Frontiers Appl. Math.*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[13] K. Stüben. Algebraic multigrid (AMG): An Introduction with Applications. GMD Report 53(1999), GMD - Forschungszentrum Informationstechnik GmbH, Schloss Birlinghoven, Sankt Augustin, Germany.

[14] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.

[15] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996. International GAMM-Workshop on Multi-level Methods (Meisdorf, 1994).

[16] Panayot S. Vassilevski. Sparse matrix element topology with application to amg and preconditioning. *Numer. Lin. Alg. Appl.*, 9:429–444, 2002.

[17] Panayot S. Vassilevski and Ludmil T. Zikatanov. Multiple vector preserving interpolation mappings in algebraic multigrid. *SIAM J. Matrix Anal. Appl.*, 27(4):1040–1055 (electronic), 2006.

Department of Mathematical Sciences, University of Colorado at Denver, Denver, CO 80217

E-mail address: ilashuk@math.cudenver.edu

Center for Applied Scientific Computing, UC Lawrence Livermore National Laboratory, 7000 East avenue, Mail–Stop L-560, Livermore, CA 94551

E-mail address: panayot@llnl.gov