Miskolc Mathematical Notes
Vol. 14 (2013), No. 1, pp. 143–158

HU e-ISSN 1787-2413

TWO NOVEL AGGREGATION-BASED ALGEBRAIC MULTIGRID METHODS

JIA LIAO, TING-ZHU HUANG, AND BRUNO CARPENTIERI

Received March 30, 2011

Abstract. In the last two decades, substantial effort has been devoted to solve large systems of linear equations with algebraic multigrid (AMG) method. Usually, these systems arise from discretizing partial differential equations (PDE) which we encounter in engineering problems. The main principle of this methodology focuses on the elimination of the so-called algebraic smooth error after the smoother has been applied. Smoothed aggregation style multigrid is a particular class of AMG method whose coarsening process differs from the classic AMG. It is also a very popular and effective iterative solver and preconditioner for many problems. In this paper, we present two kinds of novel methods which both focus on the modification of the aggregation algorithm, and both lead a better performance while apply to several problems, such as Helmholtz equation.

2000 *Mathematics Subject Classification:* 65M55; 65N55; 65F10, 65F08; 65F50

Keywords: algebraic multigrid, smoothed aggregation, preconditioner

1. Introduction

In this study we consider multigrid methods for solving large and sparse $n \times n$ linear systems

$$Au = b$$

arising from the discretization of partial differential equations.

Multigrid methods are widely popular as linear solvers because they can achieve linear algorithmic scalability for many practical problems. A very important ingredient of multigrids is the choice of both the relaxation and the correction scheme. The relaxation scheme utilizes a simple iterative solver such as the Jacobi or (symmetric) Gauss-seidel method as the smoother, and is aimed at eliminating the high-frequency components of the error. Afterwards, the low-frequency components of the error are damped via the coarse-grid correction, which consists of solving approximately the residual equation on a coarser grid.

This research is supported by 973 Program (2013CB329404), Chinese Universities Specialized Research Fund for the Doctoral Program (20110185110020), Sichuan Province Sci. & Tech. Research Project (2012GZX0080).

In order to construct the coarse-grid correction, we need to define several multigrid components, that are the hierarchy of grids, the smoothing and transfer (prolongation/restriction) operators, and a solver for the coarsest grid. Both theory and several reported experiments indicate that the rate of convergence of multigrid methods may be independent on the number of unknowns, and the complexity increases only linearly with the problem size.

In AMG methods, the grids and the operators are selected using only information from the entries of the coefficient matrix, and not using physical meshes like in Geometric multigrids (GMG). It has been shown that AMG often inherits the excellent algorithmic scalability of GMG, without requiring any geometric information. For this reason, the development of efficient and reliable AMG methods has received considerable research attention in the past decade.

Variants of AMG methods based on smoothed aggregation (referred to as SA methods) have been introduced in [17, 18]. The main algorithmic difference between the two approaches consists in the distinction between *coarse* and *fine* variables. In classic AMG techniques, the fine-level variables are splitted into two disjoint subsets, the so-called C/F splitting where C represent the variables on the coarser level. Vice versa, in the SA method the coarsening process is defined by generating aggregates rather than by constructing the C/F splitting. All unknowns in the same aggregate are strongly dependent on each other in order to guarantee the accuracy of the interpolation.

Recent trend of development of SA methods moved in two directions: 1. Use the near-null space information to accelerate the standard algorithm, such as in [3, 6]. 2. Construct more accurate interpolation operators, such as in [6, 12]. In this paper, we follow the second direction and we investigate a different strategy to construct efficient interpolation operators that can also preserve low iteration cost and storage requirements.

The paper is organized as follows. In Section 2, we briefly recall SA methods and we introduce the standard notation that is used throughout the remainder of the paper. In Section 3, we focus on the description of two modified aggregation algorithms. Finally, we demonstrate the performance of our new method in Section 4.

2. Preliminaries

We briefly recall the standard SA method and its principles following [3, 17]. Let us assume that the coefficient matrix A of the linear system is of order $n = n_1$ and it arises from the discretization of a PDE. A hierarchy of coarse-level matrices is defined as

$$A_{l+1} = (S_l P^l_{l+1})^T A_l S_l P^l_{l+1}, A_1 = A,$$

for $l = 1,\dots,L-1$. A simple and proper choice for the prolongation smoother is Richardson's method with a particular step size:

$$S_l = I - \frac{4}{3\lambda_l}A_l,$$

where λ_l is an upper bound on the spectral radius of the matrix on level l, that is $\rho(A_l) \le \lambda_l$. Notice that this smoother is different from the relaxation smoother mentioned in Section 1, like the Jacobi smoother.

Suppose that we are given a smoothing procedure for the projected system $A_l x_l = b_l$, at each level $l \in \{1,\dots,L\}$, of the form

$$x_l \leftarrow (I - R_l A_l)x_l + R_l b_l.$$

Here, R_l is some approximation of the inverse of A_l for $l = 1,\dots,L-1$, which we consider again to be the Richardson iteration ($R_l = s_l I$, where $s_l \approx \frac{1}{\rho(A_l)}$). Assume for simplicity to use a direct solver at the coarsest level: $R_L = A_L^{-1}$. We assume that

$$\lambda_{\min}(I - R_l A_l) \ge 0 \quad and \quad \lambda_{\min}(R_l) \ge \frac{1}{C_R^2 \rho(A_l)}$$

for constant $C_R > 0$ independent of the level l. This assumption enables us to make use of the existing convergence estimates for Richardson's iteration.

The SA method can be formally viewed as a standard variational multigrid method with prolongator of the form $S_l P_{l+1}^l$. One SA iteration for solving $A_1 x_1 = b_1$ is represented by $x_1 \leftarrow AMG(x_1, b_1)$. We set $AMG = AMG_1$, where $AMG_l(\cdot,\cdot)$, for $l = 1,\dots,L-p = p1$, is defined recursively as follows.

Since the prolongation smoother S_l has been formally introduced before, now we

Algorithm 1 AMG_l

```
(1) Presmoothing: Apply v presmoothings to $A_l x_l = b_l$ of the form $x_l \leftarrow (I - R_l A_l)x_l + R_l b_l$
(2) Coarse-grid correction:
    (a) Set $b_{l+1} = (S_l P_{l+1}^l)^T (b_l - A_l x_l)$.
    (b) If $l + 1 = L$, solve $A_{l+1} x_{l+1} = b_{l+1}$ by a direct method,
        Otherwise set $x_{l+1} = 0$ and perform $\gamma$ iterations of $x_{l+1} \leftarrow AMG_{l+1}(x_{l+1}, b_{l+1})$
    (c) Correct the solution on level $l$ : $x_l \leftarrow x_l + (S_l P_{l+1}^l) x_{l+1}$.
(3) Postsmoothing: apply $v$ postsmoothings to $A_l x_l = b_l$ of the form
    $x_l \leftarrow (I - R_l A_l)x_l + R_l b_l$.
$AMG_L$ returns, simply, $x_L = A_L^{-1} b_L$.
```

focus on the construction of the tentative prolongator operators P_{l+1}^l. For illustration, here we present the simplest prolongator for the 1D Laplace equation discretized on a mesh consisting of $n_1 = 3^{L-1}n_L$ nodes:

$$P^l_{l+1} = \begin{pmatrix} 1 & & & & & \\ 1 & & & & & \\ 1 & & & & & \\ & 1 & & & & \\ & 1 & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & \ddots & & & \\ & & \ddots & & & \\ & & & & 1 & \\ & & & & 1 & \\ & & & & 1 & \end{pmatrix}.$$

For later reference, we outline the standard SA setup in Algorithm 2 below.
The prolongator P^l_{l+1} and the $n_{l+1} \times r$ matrix B_{l+1} are such that

Algorithm 2

```
Given $A_1$, and $L$, do the following for $l = 1, \dots, L-1$:
    (1) Construct $\{C^l_i\}^{N_l}_{i=1}$ based on $A_l$. ($\{C^l_i\}^{N_l}_{i=1}$, a disjoint covering for all nodes.)
    (2) Construct $P^l_{l+1}$ based on $\{C^l_i\}^{N_l}_{i=1}$.
    (3) Construct the prolongation smoother: $S_l$.
    (4) Construct the coarse matrix: $A_{l+1} = (S_l P^l_{l+1})^T A_l S_l P^l_{l+1}$.
```

$$P^l_{l+1} B_{l+1} = B_l,$$

where B_l has been constructed during the setup of P^{l-1}_l or it has been given if $l = 1$. A typical example for B_l is $B_1 = (1, \cdots, 1)^T$ or $B_1 = Kernel(A)$ if A is the discrete representation of a continuous differential operator with an appropriate null space.

The prolongator P^l_{l+1} is constructed from a given system of aggregates $\{C^l_i\}^{N_l}_{i=1}$ which form a disjoint covering of the set $\{1, ..., n_l\}$. Next, we present an algorithm introduced in [19] for generating aggregates based on information on the structure of the matrix A_l.

For a given parameter θ, the strongly coupled neighborhood of the node i is defined as

$$N^l_i(\theta) = \{j : |a_{ij}| \geq \theta\sqrt{a_{ii}a_{jj}}\} \cup \{i\}.$$

This algorithm generates as output a disjoint covering $\{C^l_i\}^{N_l}_{i=1}$ of the set $\{1, ..., n_l\}$.

Algorithm 3

Let A_l be a matrix of order n_l and $\theta \in [0,1)$. Generate a disjoint covering $\{C_i^l\}_{i=1}^{N_l}$ of the set $\{1,\dots,n_l\}$ as follows.

```
Aggregate ($U$).
  Initialization: set $U = \{1,\dots,n_l\}$, $i = 1$ and $j = 0$.
Step 1: disjoint strongly-coupled neighborhoods are selected as the initial approximation of the covering:
        {
          for($i \in U$)
          {
            if($N_i^l(\theta) \subset U$)
            {$j = j+1; C_j^l = N_i^l(\theta); U = U - C_j^l;$}
          }
Step 2: each remaining $i \in U$ is added to one of the selected sets to which it is strongly connected, if possible:
          for($k <= j$) $\overline{C_k^l} = C_k^l;$
          for($i \in U$)
          {
            for($k <= j$)
            {
             if($N_i^l(\theta) \cap \overline{C_k^l} \neq \varnothing$)
               {$C_k^l = C_k^l \cup \{i\}; U = U - \{i\};$}
            }
          }
Step 3: combine the remaining $i \in U$ into aggregates that consist of subsets of strongly coupled neighborhoods:
          for($i \in U$) until $U = \varnothing$
          {
              $j = j+1; C_j^l = N_i^l(\theta) \cap U; U = U - C_j^l;$
          }
        }
```

The columns of P_{l+1}^l associated with the aggregate C_i^l are formed by restriction of the rows of B_l onto the aggregate C_i^l. Each aggregate gives rise to r degrees of freedom on the coarse-grid.

The detailed algorithm follows here. For ease of presentation, we assume that the fine level unknowns are numbered lexicographically within each aggregate. A different order can be used by renumbering them. The algorithm and figure described

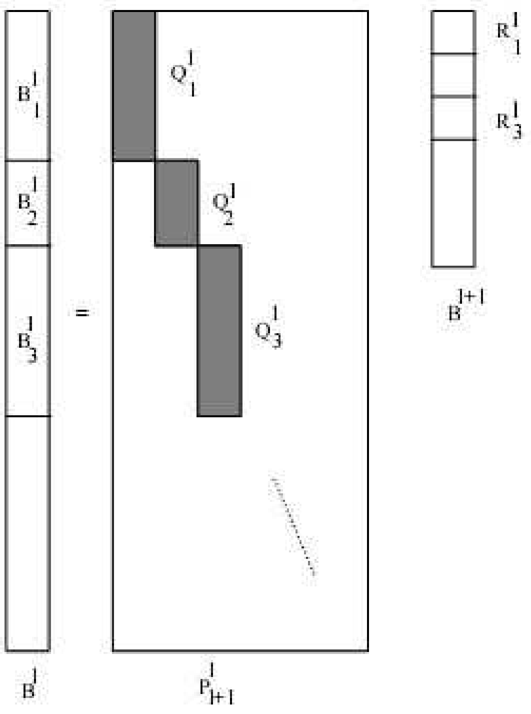

FIGURE 1. The structure of the tentative prolongator P_{l+1}^{l}.

above are based on the assumption that the fine level unknowns are numbered consecutively within each aggregate. The typical structure of a tentative prolongator is described below, where we denote by $\times$ one nonzero element in the matrix.

$$\begin{pmatrix} \times & & & & \\ \times & & & & \\ & \times & & & \\ & \times & & & \\ \times & & & & \\ & & \times & & \\ & \times & & & \\ & & \times & & \\ & & \times & & \\ & & & \times & \\ & & \times & & \\ & & & & \times \end{pmatrix}.$$

Algorithm 4

For the given system of aggregates $\{C_i^l\}_{i=1}^{N_l}$ and the $n_l \times r$ matrix satisfying $P_l^1 B_l = B_1$, we create a prolongator, a matrix B_{l+1} and the unknowns on level $l+1$ as follows:

(1) Let d_i denote the number of unknowns associated with aggregate C_i^l. Split the $n_l \times r$ matrix B_l into blocks B_i^l of size $d_i \times r$, $i = 1, \dots, N_l$, each corresponding to the unknowns on an aggregate C_i^l.

(2) A reduced QR decomposition is applied to $B_i^l = Q_i^l R_i^l$, where Q_i^l is an $d_i \times r$ orthogonal matrix, and R_i^l is an $r \times r$ upper triangular matrix.

(3) Create the tentative prolongator $P_{l+1}^l = diag(Q_1^l, \dots, Q_{N_l}^l)$, and set

$$B_{l+1} = \begin{pmatrix} R_1^l \\ R_2^L \\ \cdots \\ R_{N_l}^l \end{pmatrix}.$$

.

(4) For each aggregate C_i^l, the coarsening gives rise to r unknowns on the coarse level (the i-th block column of P_{l+1}^l).

In Algorithm 4, the QR decomposition has been used to generate the prolongator P_{l+1}^l. A basic prolongator P_{l+1}^l may be computed to be piecewise constant, which leads to the simplest definition:

$$(P_{l+1}^l)_{ij} = \begin{cases} 1, i \in C_j^l, \\ 0, \text{otherwise}. \end{cases}$$

However, this choice is not efficient in general and the computation is still time consuming. A different way to generate the prolongator P_{l+1}^l has been considered in [6]. It uses the singular value decomposition for minimizing the interpolation error. In Algorithm 4, we need to change the step 2 as:

The reduced singular value decomposition is applied to $B_i^l = Q_i^l T_i^l M_i^l$, $R_i^l = T_i^l M_i^l$, where Q_i^l is an $d_i \times r$ matrix, and R_i^l is an $r \times r$ matrix.

For ease of discussion, in this paper we simply denote this algorithm as the SVSA method. See [6] for more details.

3. TWO NOVEL AGGREGATION SCHEMES

We briefly describe a general measure of strength of connection between two unknowns. In [2], the authors propose a general definition of strength of connection and an algorithm, which is not restricted to M-matrices.

We define the column vector $G^{(i)}$ first, with entries $(G^{(i)})_j = (A^{-1})_{ij}$. Next, we define the strength measure as follows:

$$S_{ij} = \frac{\left\| G^{(i)} - (G^{(i)})_j I^{(j)} \right\|_A}{\left\| G^{(i)} \right\|_A},$$

where $I^{(j)}$ is the jth canonical unit vector.

It is evident that this measure is not a practical option to implement, as $G^{(i)}$ is the column vector of A^{-1}. A natural choice is to apply the relaxation scheme of the AMG solver on the Gauss-Jordan system $AG = I$ columnwise, that is $AG^{(i)} = I^{(i)}$ for each node i.

Now, we can give the full algorithmic computation of the matrix S_{ij} as follows: for each node i, $1 \le i \le n$:
(1) relax μ times on $AG^{(i)} = I^{(i)}$, starting from a zero initial guess,
(2) for each j such that the approximation to $(G^{(i)})_j$ is non-zero, compute

$$S_{ij} = \frac{\left\| \widehat{G}^{(i)} - (\widehat{G}^{(i)})_j I^{(j)} \right\|_A}{\left\| \widehat{G}^{(i)} \right\|_A}. \tag{3.1}$$

Once the measure of strength of connection has been computed, the set of nodes that are strongly dependent on node i is defined as

$$S_i = \{ j : S_{ij} - 1 \ge \theta \max_{k \ne i} \{ S_{ik} - 1 \} \} \tag{3.2}$$

In [2], the authors define the strong connections based on the entries of A^{-1} instead of those of A; then they present the theoretical analysis, and finally the practical algorithmic implementation. The key point in [2] is to replace the traditional definition of strong dependence in classic AMG with the Formula 3.1, which produces efficient AMG V-cycles for many problems.

Inspired by this analysis, we aim at applying the Formula 3.2 to our SA method. As we mentioned before, clearly the connection between two nodes in the same aggregate should be as strong as possible to make the interpolation more accurate. The aggregation Algorithm 3 is based on the definition of strongly coupled neighborhood:

$$N_i^l(\theta) = \{ j : |a_{ij}| \ge \theta \sqrt{a_{ii} a_{jj}} \} \cup \{ i \}.$$

Here, we utilize the Formula 3.2 to the Algorithm 3 and we have a modified aggregation scheme, which results into Algorithm 5:

Generally, as output of this algorithm, a disjoint covering is constructed by Algorithm 5. Intuitively, the nodes in the same aggregate inherit the strong connectivity from Formula 3.2, which does favor better performance of the AMG method.

It is not difficult to establish an AMG scheme with the components defined above, once an appropriate smoother and cycling strategy are selected. Then we have a novel

Algorithm 5

Input: $U = \{1,\dots,n_l\}$, strong coupling threshold θ

Output: a disjoint covering $\{C_i^l\}_{i=1}^{N_l}$ of the set $\{1,\dots,n_l\}$

Definition: for all $i \in U$, $N_i^l(\theta) = \{j : S_{ij} - 1 \geq \theta \max_{k \neq i}\{S_{ik} - 1\}\}$, where S_{ij} is defined by Formula 3.1

Algorithm: the loop process is the same as Algorithm 3, the difference is the definition of S_{ij}.

SA method whose prolongator differs from the one of the traditional SA method because of the different aggregation scheme. The numerical results in the next section will show better performance of our novel methods.

Similarly to what we did in Algorithm 5, here we also propose another novel aggregation scheme. Let us go back to the standard aggregation Algorithm 3 and highlight the step 1. The strongly coupled neighborhood of each node has been generated and disjoint strongly coupled neighborhoods are selected as initial approximation of the covering. We noticed that the set U is changing over the recursion. If we calculate the strongly coupled neighborhood only in the remaining set U, to be more specific, we define:

$$M_i^l(\theta) = \{j : |a_{ij}| \geq \theta\sqrt{a_{ii}a_{jj}}\} \cup \{i\}$$
$$N_i^l(\theta) = M_i^l(\theta) \cap U.$$

Another variant of standard aggregation scheme results. In step 1, the assumption $N_i^l(\theta) \subset U$ is not necessary anymore for the definition of $N_i^l(\theta)$. Algorithm 6 follows below:

Algorithm 6

Input: $U = \{1,\dots,n_l\}$, strong coupling threshold θ

Output: a disjoint covering $\{C_i^l\}_{i=1}^{N_l}$ of the set $\{1,\dots,n_l\}$

Definition: for all $i \in U$, $M_i^l(\theta) = \{j : |a_{ij}| \geq \theta\sqrt{a_{ii}a_{jj}}\} \cup \{i\}$, $N_i^l(\theta) = M_i^l(\theta) \cap U$

```
Step 1: {
          $j = 0; i = 1;$
          for($i \in U$)
          {
            $j = j + 1, C_j^l = N_i^l(\theta), U = U - C_j^l$
          }
        }
```

After step 1, the strongly coupled neighborhood of each node is either $\varnothing$ or it is the covering set. The latter case means that the algorithm stops. From Algorithm 6, we clearly see that all the nodes in the same aggregate are mutually coupled, contrast to Algorithm 3 and 5. Additionally, we achieve some gaining in time, leading potentially to overall better performances.

4. Numerical Tests

We illustrate the performance of our two methods on different matrix problems, also in comparison against the standard SA method in the first subsection and the classic AMG method in the second subsection.

In our experiments, we used an adaptive scheme which consists of applying a few iterations of relaxation to the homogeneous problem, $Ae = 0$, to expose the slowly converging components of the error. The residuals produced by the iterations are then used to update the approximate solution. We combined the SA method with this adaptive scheme to yield an enhanced SA method. Notice that this approach is different from the pure αSA method described in [3], which has an expensive setup cost and high efficiency, that cannot be achieved in our problems. For the sake of simplicity and cost effectiveness, we applied several steps of a weighted Jacobi smoother to generate some near-null space vectors that were used as initial approximations.

Next, we introduce some parameters for the construction of the SA scheme. We used V(1,1) with weighted Jacobi smoother, and we ran five post-smoothing steps with a weighed Jacobi smoother. Here we have to point out that, according to the basic theory of MG methods, it is a fact that no value of ω effectively eliminates the smooth, or low-frequency, error components. However, it is possible to reduce them significantly. We denote by R_ω the error propagation matrix of the weighted Jacobi method. The eigenvalues of the iteration matrix are given by

$$\lambda_k(R_\omega) = 1 - 2\omega \sin^2(\frac{k\pi}{2n}), \quad 1 \leq k \leq n-1$$

so that

$$\lambda_{n/2}(R_\omega) = -\lambda_n(R_\omega).$$

The solution of this equation gives the value $\omega = \frac{2}{3}$, which is our choice here. Another parameter is the strength threshold θ which is set to 0.25 in our experiments. In Algorithm 4, we used the singular value decomposition instead of the QR decomposition to achieve better performance. The traditional SA method with singular value decomposition was denoted as SVSA; the variant of SVSA based on Algorithm 5 is denoted in the forthcoming experiments as SVSA1, and that based on Algorithm 6 as SVSA2. All our numerical tests are performed using the MATLAB environment version 7.8.0 (R2009a) run on the Windows 7 operating system.

For each experiment shown in the following tables, we report the following performance measures:

(1) the time cost for the construction of the AMG preconditioner (column ”P-T”) and for solving the linear system (column ”I-T”);

(2) the memory burden (column "M-cost"), calculated as the sum of the number of nozero entries in the projected matrix A at each level divided by the number of nonzeros in the coefficient matrix A;
(3) the number of iterations (column "Its") required by the Krylov subspace method to reduce the initial residual by six orders of magnitude. The symbol "-1" indicates a failure in the solving phase due to two possible reasons, either the preconditioner is ill-conditioned and the iterative solution was not started, or the number of iterations was larger than 2000.
(4) The reduction factor (column "r-ratio"), which gives the ratio of the sum of the number of unknowns at all levels of the initial matrices to the number of unknowns in the original system.

4.1. *Helmholtz equation*

In this section we report on experiments for solving sparse linear systems resulting from the five point finite difference approximation of the 2D Helmholtz equation

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}-k^2u=f \quad in \quad \Omega=(0,1)\times(0,1)$$

with the right function $f=\sin(\pi x)\sin(\pi y)\sin(\sqrt{2}\pi x)\sin(\sqrt{3}\pi y)$ and $u(x,y)=0$ everywhere on $\partial\Omega$.

For Helmholtz equation, empirically, we keep $\frac{k}{n}=0.625$ constant for the iterative solution. While $k=20,30$, the grid number is $n=32,48$. Here we use a multigrid-preconditioned Krylov subspace method. The Bi-CGSTAB method is the Krylov subspace method of choice in our experiments for the Helmholtz equation, especially for the problem without damping.

TABLE 1. Experiments with SA-preconditioned Bi-CGSTAB method for Helmholtz equation.

	Method	Its	M-cost	r-ratio	P-T	I-T
k=20	SVSA	209.5	1.2845	1.1719	127.1	11.2
	SVSA1	141.5	1.0929	1.0605	292.1	7.8
	SVSA2	88.5	1.6859	1.5000	127.1	6.9
k=30	SVSA	620.5	1.2931	1.1719	699.0	47.5
	SVSA1	383.5	1.0791	1.0512	2032.8	25.0
	SVSA2	293	1.6907	1.5000	715.5	67.5

Here, for comparison we also try to solve the system using another Krylov subspace method, which is GMRES with restart equal to 20.

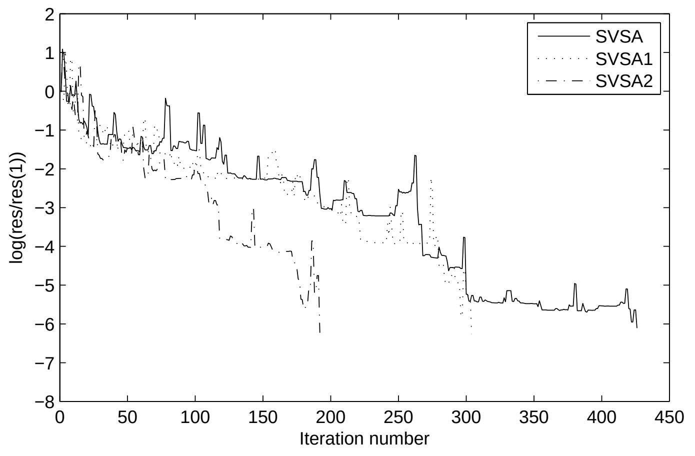

FIGURE 2. Table 1, k=20.

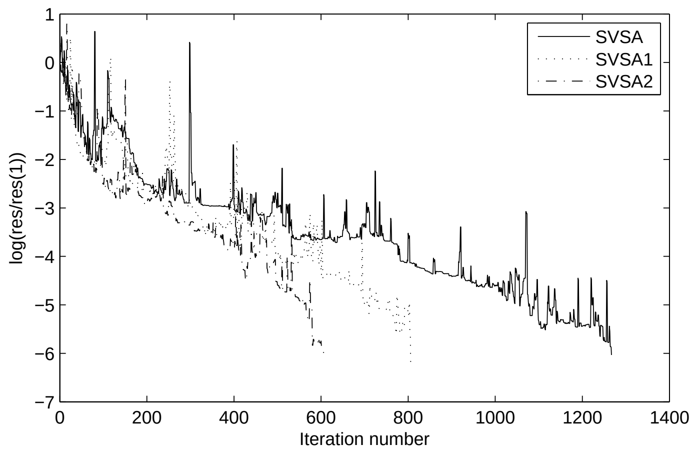

FIGURE 3. Table 1, k=30.

4.2. *Experiments with Harwell-Boeing matrices.*

Here, in Table 3, we present the comparison of classic AMG and our new method on five matrices. These matrices arise from different applications: SHERMAN1 and SAYLR3 come from computational fluid dynamics problems, PLBUCKLE and BCSSTK08 from structural problems and CAGE8 from directed weighted graphs.

The column ”cond” shows the condition number of the matrix, which is a measure of the sensitivity of linear solvers to numerical errors. We can see from the Table 3

TABLE 2. SA-preconditioned GMRES method for Helmholtz equation.

	Method	Its	M-cost	r-ratio	P-T	I-T
k=20	SVSA	2800	1.2845	1.1719	134.0	82.3
	SVSA1	1179	1.0929	1.0605	317.9	33.3
	SVSA2	856	1.6859	1.5000	135.7	33.9
k=30	SVSA	15738	1.2931	1.1719	698.0	646.2
	SVSA1	3519	1.0791	1.0512	1959.3	117.8
	SVSA2	2062	1.6907	1.5000	669.8	240.4

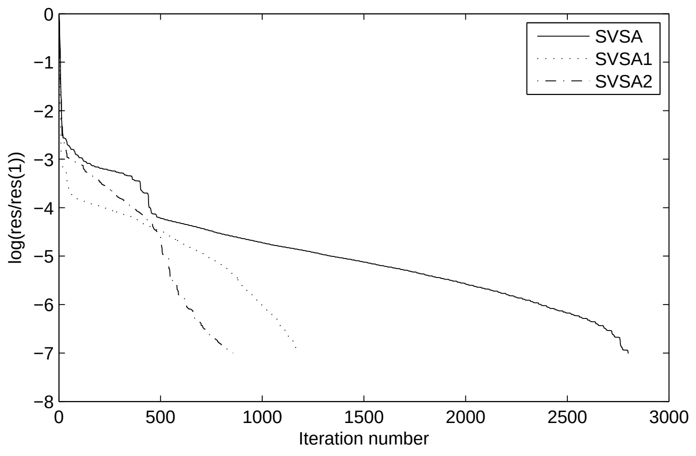

FIGURE 4. Table 2, k=20.

that some linear systems are ill-conditioned and thus difficult to solve. In the third column, called ”Method”, ”C-AMG” denotes the classic AMG.

The results highlight the robustness of the SVSA2 preconditioner, which delivered faster and more stable convergence. There are three cases where C-AMG diverge and SVSA2 converges within a few iteration steps and requires less time cost. Even on the two problems where C-AMG converges, our method is more efficient with respect to memory cost, number of iterations and solution time.

5. CONCLUSION

All the numerical experiments reported in this paper illustrate an interesting behavior of the three proposed aggregation schemes. In most of our runs, the SVSA2 method decreases the number of iterations of the standard SA method by a factor of two, or sometimes larger. Also the solution time is moderately reduced for some

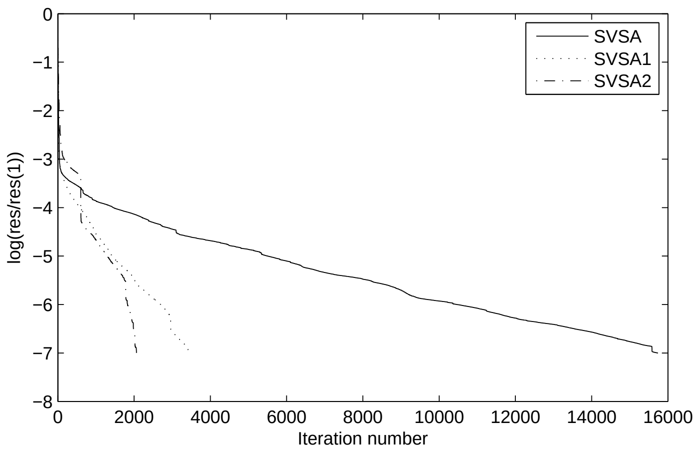

FIGURE 5. Table 2, k=30.

Matrix	cond	Method	P-T	I-T	Its	r-ratio	M-cost
SHERMAN1	2.2575×10^4	C-AMG	68.809	0.55672	-1	-1	-1
		SVSA2	40.885	0.86451	6.5	1.684	1.6672
PLBUCKLE	4.128×10^6	C-AMG	73.739	9.5594	60.5	1.3378	1.4889
		SVSA2	56.905	3.3188	51	1.3261	1.2119
BCSSTK08	4.7262×10^7	C-AMG	62.33	0.75017	-1	-1	-1
		SVSA2	54.329	0.63049	4	1.7486	1.8298
CAGE8	5.8238×10	C-AMG	59.569	0.12071	1	1.2512	1.9009
		SVSA2	43.33	0.048572	1	1.2148	1.3515
SAYLR3	INF	C-AMG	73.839	0.55805	-1	-1	-1
		SVSA2	42.71	0.65484	6.5	1.684	1.6672

TABLE 3. Performance comparison of the two methods C-AMG and SVSA2.

problems. The only flaw are the larger r-ratio and M-cost, which means extra storage cost. However, it is relatively small in comparison to the order of matrix A.

A significant disadvantage of SVSA1 is its computational complexity in computing the aggregates, being inherited from [2]. In Algorithm 5, the calculation of strong neighborhood requires the number of relaxation to be small. Now, searching for an optimal number of relaxations steps in Formula 3.1 is still a problem under investigation.

The results suggest that the two novel methods proposed in this paper can improve the performance of the SA method to some extent for various problems. The whole

SA scheme includes many ingredients, such as the adaptive scheme and the choice of the smoother. Different choices lead to different performance for each problem, but the key point is the interplay between interpolation and adaptive method, which is a subject of ongoing research.

ACKNOWLEDGEMENT

We are thankful to the anonymous reviewers for their valuable comments that greatly helped us improve the quality of the original manuscript. This work was partly developed using the computing facilities and thanks to the fertile working environment of the Institute of Mathematics and Computing Science, University of Groningen.

REFERENCES

[1] T. Airaksinen, E. Heikkola, A. Pennanen, and J. Toivanen, "An algebraic multigrid based shifted-laplacian preconditioner for the Helmholtz equation," *Journal of Computational Physics.*, vol. 226, no. 1, pp. 1196–1210, 2007.

[2] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, "An energe-based AMG coarsening strategy," *Numer. Linear Algebra Appl.*, vol. 13, pp. 133–148, 2006.

[3] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, "Adaptive smoothed aggregation(αSA) multigrid," *SIAM Rev.*, vol. 47, no. 2, pp. 317–346, 2005.

[4] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. McCormick, and J. Ruge, "Adaptive algebraic multigrid," *SIAM. J Sci. Comput.*, vol. 27, no. 4, pp. 1261–1286, 2006.

[5] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. Philadelphis: SIAM, 2000.

[6] E. Chow, "Aggregation-based multilevel method using smooth error vectors," *SIAM. J. Sci. Comput.*, vol. 27, no. 5, pp. 1727–1741, 2006.

[7] H. C. Elman, O. G. Ernst, and D. P. O'Leary, "A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations," *SIAM. J. Sci. Comput.*, vol. 23, no. 4, pp. 1291–1315, 2001.

[8] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik, "A novel multigrid based preconditioner for heterogeneous Helmholtz problems," *SIAM. J Sci. Comput.*, vol. 27, no. 4, pp. 1471–1492, 2006.

[9] Y. A. Erlangga, C. Vuik, and C. W. Oosterlee, "Comparison of multigrid and incomplete LU shifted-Laplace preconditioners for the in homogeneous Helmholtz equation," *Applied Numerical Mathematics.*, vol. 56, no. 5, pp. 648–666, 2006.

[10] A. C. Muresan and Y. Notay, "Analysis of aggregation-based multigrid," *SIAM. J. Sci. Comput.*, vol. 30, no. 2, pp. 1082–1103, 2008.

[11] A. Nägel, R. D. Falgout, and G. Wittum, "Filtering algebraic multigrid and adaptive strategies," *Comput Visual Sci.*, vol. 11, pp. 159–167, 2008.

[12] Y. Notay, "Aggregation-based algebraic multilevel preconditioning," *SIAM J. Matrix Anal. Appl.*, vol. 27, no. 4, pp. 998–1018, 2006.

[13] Y. Notay and P. S. Vassilevski, "Recursive Krylov-based multigrid cycles," *Numer. Linear. Algebra. Appl.*, vol. 15, pp. 473–487, 2008.

[14] L. N. Olson and J. B. Schroder, "Smoothed aggregation for Helmholtz problems," *Numer. Linear Algebra Appl.*, vol. 17, pp. 361–386, 2010.

[15] K. Stüben, "A review of algebraic multigrid," *Journal of Computational and Applied Mathematics.*, vol. 128, no. 1-2, pp. 281–309, 2001.

[16] O. Tatebe, "MGCG method: a robust and highly parallel iterative method," Ph.D. dissertation, Tsukuba University, the graduate school of the partial fulfillment of the requirements for the degree of doctor of science in information science, 1996.

[17] P. Vaněk, M. Brezina, and J. Mandel, "Convergence of algebaic multigrid based on smoothed aggregation," *Numer. Math.*, vol. 88, pp. 559–579, 2001.

[18] P. Vaněk, J. Mandel, and M. Brezina, "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems," *Computing.*, vol. 56, no. 3, pp. 179–196, 2002.

[19] C. Wagner, "Introduction to Algebraic Multigrid," 1998, Course Notes of an Algebraic Multigrid Course at University of Heidelberg in the Wintersemester.

Authors' addresses

Jia Liao
University of Electronic Science and Technology of China, School of Mathematical Sciences, 611731 Chengdu, P. R. China
Current address: University of Groningen, Institute of Mathematics and Computing Science, 9747 AG Groningen, The Netherlands
E-mail address: 343434hg@163.com, j.liao@rug.nl

Ting-Zhu Huang
University of Electronic Science and Technology of China, School of Mathematical Sciences, 611731 Chengdu, P. R. China
E-mail address: tingzhuhuang@126.com, tzhuang@uestc.edu.cn

Bruno Carpentieri
University of Groningen, Institute of Mathematics and Computing Science, 9747 AG Groningen, The Netherlands
E-mail address: b.carpentieri@rug.nl, bcarpentieri@gmail.com