# SPARSIFIED COARSENING MULTIGRID FOR CONVECTION DIFFUSION

ERAN TREISTER

Abstract. Traditional algebraic multigrid (AMG) methods use (Petrov-)Galerkin coarsening where the sparsity pattern and operator complexity of the multigrid hierarchy is dictated by the multigrid transfer operators. Therefore, AMG algorithms usually compromise between the quality of these operators and the aggressiveness of the coarsening, which affects their rate of convergence and operator complexity. In many scenarios, the multigrid coarse operators tend to be much denser than the fine operator as the coarsening progresses. Such behavior is especially problematic in parallel AMG computations where it imposes expensive communication overhead. In this work we present a new algebraic technique for controlling the sparsity pattern of the operators in the multigrid hierarchy, independently of the choice of transfer operators. Our algorithm sparsifies the (Petrov-)Galerking operators while preserving their right and left near null-spaces. Numerical experiments for Convection Diffusion problems demonstrate the efficiency and potential of this multigrid approach.

## 1. Introduction

Multigrid methods are well known for their efficiency in solving linear systems arising from the discretization of elliptic partial differential equations (PDEs) [3, 18, 6, 21]. The discretization yields a sparse, typically large system of equations, $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{n\times n}$, and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. To solve it, AMG methods use two complementary components: *relaxations*, and coarse-grid correction (CGC). The *relaxations* are local iterative methods, such as Jacobi or Gauss-Seidel, and are usually inefficient in handling certain error modes, called "algebraically smooth". CGC aims at handling these modes, and is done by solving a coarse-grid problem $A_c\mathbf{e}_c = \mathbf{r}_c$. In most AMG methods, this problem is defined by the (Petrov-)Galerkin coarsening

$$A_c = RAP, \quad \mathbf{r}_c = R(\mathbf{b} - A\mathbf{x}), \tag{1}$$

which is a projection of the error equation onto the subspace defined by the full-rank prolongation and restriction operators, $P \in \mathbb{R}^{n\times n_c}$ and $R \in \mathbb{R}^{n_c\times n}$, respectively, with $n_c < n$. The process is applied recursively, resulting in a hierarchy of successively coarser problems and their associated operators.

One drawback of the (Petrov-)Galerkin coarsening is that the control over the sparsity of $A_c$ is quite limited and is dictated by $P$ and $R$. This might lead to a reduced sparsity of the Galerkin coarse-grid operators and high overall computational complexity of the multigrid algorithm. In the case of large-scale parallel computing, this usually leads to high communication overhead on coarse grids [15, 14] and loss of scalability, especially in 3D. Thus, one must often compromise between the quality of these operators and the aggressiveness of coarsening, which affects the rate of convergence and the operator complexity of the algorithm. Recently, there has been an effort to develop efficient multigrid algorithms that explicitly control the sparsity pattern of the multigrid hierarchy [20, 22], or sparsify the Galerkin AMG operators [14]. These ideas have yet to reach their full potential, and were only applied for symmetric problems. We follow a similar framework in this paper.

*Date*: January 1, 2013.

Dept. of Computer Science, Technion—Israel Institute of Technology. email: eran@cs.technion.ac.il.

In this paper we consider non-symmetric problems, focusing on the convection-diffusion equation

$$-\epsilon\Delta u + \mathbf{v}\cdot\nabla u = f \tag{2}$$

that appears in flow simulations. One popular approach to treating this problem is by using aggregation-based AMG methods [19, 7, 1, 2, 13, 9, 10, 11], where the coarsening is done by clustering (aggregating) the grid unknowns. In its simplest form of pure (non-smoothed) aggregation (AGG), $P$ and $R$ are sparser than those obtained by most other AMG approaches, and the operator complexity of the multigrid hierarchy is usually well-bounded and attractive. However, it is difficult to obtain grid independent convergence using this approach, and therefore, the approach of *Smoothed Aggregation* (SA) [19, 1, 2, 13, 8] is often preferred over AGG. In SA we smooth the simple aggregation operators by a relaxation operator. This improves the convergence properties of the multigrid solver, but it also increases the operator complexity of the multigrid hierarchy. Therefore, when using SA we must make sure that our coarsening is aggressive enough to prevent exaggerated stencil growth on coarse-grids. However, for a convection-dominated problem (2), such aggressive aggregation coarsening (say $3\times 3$ for 2D problems on structured meshes) significantly harms the quality of the aggregation operators compared to moderate coarsening (say, $2\times 2$) [7], whereas the latter leads to an unbounded operator complexity using SA. This behavior is just another example of the tradeoff between using quality transfer operator vs. maintaining low operator complexity.

In a very recent paper [11], impressive results are achieved for solving (2) using AGG with moderate coarsening (coarsening factor of about 4). To overcome the slow convergence caused by using pure aggregation, the multigrid process includes acceleration on all levels of the hierarchy (known as K-cycle), requiring a more elaborate recursive structure (usually W-cycles). A similar technique is also used in [12]. However, such cycles may be costly, even though their storage complexity is well-bounded, especially when considering parallel multigrid computations [15, 5].

In this paper, we present a new AMG algorithm that controls the sparsity pattern of the multigrid hierarchy, using ideas related to [22, 14]. The new algorithm is developed for non-symmetric problems but it preserves the symmetry property of the coarse operators for symmetric fine operators. Our algorithm uses the aggregation framework as a basis platform, by applying smoothed aggregation to define the transfer operators and simple aggregation Galerkin coarse-grid operators for defining a sparse non-zero pattern of $A_c$. Once the transfer operators and the target sparsity pattern are set, our algorithm sparsifies the Galerkin SA coarse operator to match the chosen sparsity pattern of $A_c$.

## 2. Algorithm Description

We follow the rationale of [20, 22], where the main idea is to separate the issue of the coarse-grid operator sparsity pattern from the transfer operators $R$ and $P$. We fix the sparsity pattern of $A_c$ and, independently, use high quality transfer operators $P$ and $R$. Furthermore, suppose that $A_c$ is constructed in such a way that it yields an exact approximation of the Galerkin operator with respect to certain basis vectors. Assume that the current error, $\mathbf{e}$, is in the range of $P$, i.e., $\mathbf{e} = P\mathbf{e}_c$. Also, assume that $A_c$ is constructed such that $A_c\mathbf{e}_c = RAP\mathbf{e}_c$, implying that $\mathbf{e}_c$ can be represented as a linear combination of the basis vectors that are used to define $A_c$. Then, the two-level cycle of such an algorithm eliminates the error $\mathbf{e}$:

$$\begin{aligned}\mathbf{e}_{new} &= \left[I - P(A_c)^{-1}RA\right]\mathbf{e} &&= [I - P(A_c)^{-1}RA]P\mathbf{e}_c \\ &= [P - P(A_c)^{-1}RAP]\mathbf{e}_c &&= P(A_c)^{-1}[A_c - RAP]\mathbf{e}_c \;=\; 0.\end{aligned} \tag{3}$$

This property motivates our work, as well as the work of [20, 22].

2.1. **The Sparsified Smoothed Aggregation (SpSA) Framework.** Our approach is based on aggregation, which is defined by a partitioning of the fine-grid index set $\{1, ..., n\}$ into $n_c$ disjoint subsets $\{C_j\}_{j=1}^{n_c}$, called aggregates. Given these aggregates, a piece-wise constant tentative prolongation operator, which we denote by $P_a$ ($a$ for aggregation), is defined:

$$(P_a)_{i,j} = \begin{cases} 1 & i \in C_j, \\ 0 & otherwise. \end{cases} \tag{4}$$

For the tentative restriction operator $R_a$ we use $R_a = P_a^T$. In this work, we define our aggregation by the Bottom-Up approach used in [16, 17], but other aggregation-based coarsening methods as in [19, 10, 11] may also be suitable for our algorithm.

As mentioned above, we use SA to improve the convergence properties of AGG, by smoothing the aggregation operators $P_a$ and $R_a$. More precisely, general SA operators $P_s$ and $R_s$ ($s$ for smoothed aggregation) are often defined by

$$P_s = (I - \omega Q A^F) P_a, \qquad R_s = R_a (I - \omega A^F Q). \tag{5}$$

where the matrix $Q$ is a diagonal preconditioner of $A$, $\omega$ is a damping parameter, and $A^F$ is a filtered version of the matrix $A$. For $Q$, the inverse of the diagonal of $A$ is usually chosen. Then $I - \omega QA$ is the error propagation matrix associated with the damped Jacobi relaxation. In this work, we use the SPAI diagonal preconditioner [4] for $Q$, i.e., we choose $Q$ as the diagonal matrix that minimizes $\|I - QA\|_F$ which leads to

$$Q_{ii} = \frac{A_{i,i}}{\sum_j A_{i,j}^2}, \quad i = 1, ..., n. \tag{6}$$

This is a more sophisticated relaxation operator than Jacobi, and in the context of smoothing $P_a$ it is related to the energy minimization diagonal preconditioner (EMIN) [13]. As in [13], we also found that such a $Q$ is much more efficient for solving (2) than the Jacobi diagonal preconditioner. We also found that additional damping is worthwhile, at least with the moderate Bottom-Up aggregation [17] that we use (in [13], a much more aggressive aggregation is used). We note that the diagonal preconditioner (6) is defined using $A$, and not using $A^F$.

The filtering in (5) aims at removing small entries from $P_s$ and $R_s$, to prevent unnecessary stencil growth. In the context of convection-dominated problem (2), the small diffusion entries may have little influence on the quality of $P_s$ but still cause a severe stencil growth. Such is also the case in anisotropic diffusion. We use the standard filtering of [19]: let $N_i(\epsilon) = \{j : |A_{i,j}| \geq \epsilon\sqrt{A_{i,i}A_{j,j}}\}$, then $A^F$ is defined by

$$A^F_{i,j} = \left\{ \begin{array}{cc} A_{i,j} & \text{if } j \in N_i(\epsilon) \\ 0 & otherwise \end{array} \right\}, \quad A^F_{i,i} = A_{i,i} - \sum_{j \notin N_i(\epsilon)} A_{i,j}. \tag{7}$$

In this work we use $\epsilon = 0.02$.

Our next step is to define two Galerkin operators:

$$A_c^a = R_a A P_a, \quad \text{and} \quad A_c^s = R_s A P_s, \tag{8}$$

that are based on the non-smoothed and smoothed aggregation operators, respectively. Next we perform the "sparsening" process, where we sparsify $A_c^s$ to achieve the sparsity pattern of $A_c^a$. This setup process is described in Algorithm 1.

2.1.1. ***Sparsity Patterns in the Aggregation Framework.*** The transfer operators in (5) and (8) have some unique properties that are related to their sparsity patterns. These will be used in our sparsening algorithm, described later. We denote the sparsity pattern of any matrix $A$ as

$$\mathcal{S_P}(A) = \{(i, j) : A_{i,j} \neq 0\}. \tag{9}$$

**Algorithm: SpSA-Setup**

```
(1) Define the tentative prolongation $P_a$ and restriction $R_a = P_a^T$.
(2) Define $SA$ operators $P_s$, $R_s$.
(3) Apply Petrov-Galerkin Coarsening: $A_c^a = R_a A P_a$, $A_c^s = R_s A P_s$.
(4) Sparsify $A_c^s$ onto the sparsity pattern of $A_c^a$: $A_c = \mathbf{Sparsify}(A_c^s, A_c^a)$
(5) Apply recursion on $A_c$ to generate the next levels.
```

**Algorithm 1:** Sparsified Smoothed Aggregation (SpSA) Setup

By (5), we have that

$$\mathcal{S_P}(A_c^s) \supseteq \mathcal{S_P}(A_c^a) \tag{10}$$

up to chance cancellations of elements which we ignore in our description. In addition, since $\mathcal{S_P}(A) = \mathcal{S_P}(I - QA)$ for our choice of $Q$ in (6), we have that

$$\mathcal{S_P}(A_c^a) \supseteq \mathcal{S_P}(R_s P_a) \quad \text{and} \quad \mathcal{S_P}(A_c^a) \supseteq \mathcal{S_P}(R_a P_s), \tag{11}$$

with equalities in the case of no filtering in (5).

2.2. **The Sparsening Procedure.** We next describe our sparsening procedure used to approximate $A_c^s$ using only the sparsity pattern of $A_c^a$. Our process aims to generate a coarse matrix $A_c$ that, on top of its sparsity constraints, has the same product as $A_c^s$ and $(A_c^s)^T$ with two vectors $\mathbf{x}$ and $\mathbf{y}$, respectively. That is, motivated by (3), we impose $A_c\mathbf{x} = A_c^s\mathbf{x}$ and $(A_c)^T\mathbf{y} = (A_c^s)^T\mathbf{y}$. The vectors $\mathbf{x}$ and $\mathbf{y}$ are the right and left near null-space "prototypes" that also usually feature in the construction of prolongation and restriction operators in many multigrid algorithms; see [2, 17] and references therein. We require that $\mathbf{x}$ and $\mathbf{y}$ are strictly positive, which is the case for M-matrices for example.

We start our process by copying the values in the entries of $A_c^s$ that belong to $\mathcal{S_P}(A_c^a)$, i.e.,

$$\textbf{if} \quad (A_c^a)_{k,i} \neq 0 \quad \textbf{then} \quad (A_c)_{k,i} \leftarrow (A_c^s)_{k,i}. \tag{12}$$

Otherwise, we have an entry satisfying $(A_c^a)_{k,i} = 0$ and $(A_c^s)_{k,i} \neq 0$; this is an entry that we wish to eliminate (set $(A_c)_{k,i} = 0$), while maintaining the product of $A_c^s$ and $(A_c^s)^T$ with the right and left near null-spaces $\mathbf{x}$ and $\mathbf{y}$ respectively. That is, we impose

$$\textstyle\sum_\ell (A_c^s)_{k,\ell} x_\ell = \sum_\ell (A_c)_{k,\ell} x_\ell \quad \text{and} \quad \sum_\ell (A_c^s)_{\ell,j} y_\ell = \sum_\ell (A_c)_{\ell,j} y_\ell. \tag{13}$$

Now, if we set $(A_c)_{k,i} = 0$, and do not have a $(k,i)$ entry on the right-hand-sides of (13), we break these equalities, and thus we need to correct them by changing other entries as well. We elaborate on the choice of these entries in the next section.

2.2.1. ***Obtaining a surrogate path for eliminating the*** $(k,i)$ ***entry.*** First, we remark that we must avoid the simplest choice of entries to make up for an elimination of a $(k,i)$ entry—the diagonal and "mirror" entries, $(k,k)$, $(i,i)$ and $(i,k)$. Considering that we might need to eliminate both $(A_c^s)_{k,i}$ and $(A_c^s)_{i,k}$, we have to satisfy four equations: the equalities (13) for both $i$ and $k$. Since we have only the two diagonal entries $(k,k)$, $(i,i)$ at our disposal (the off-diagonals are zeroed), this task is impossible, since we have four equations and only two variables to satisfy them. Thus, we conclude that additional or other entries need to be changed.

We next describe a set of entries in $A_c^a$ that are safe to use for eliminating $(A_c^s)_{k,i}$. In Petrov-Galerkin coarsening, every non-zero entry satisfies

$$(A_c^s)_{k,i} = \sum_j \sum_\ell (R_s)_{k,j} A_{j,\ell} (P_s)_{\ell,i}, \tag{14}$$

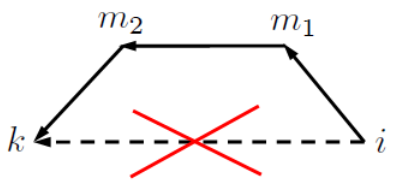

FIGURE 1. A surrogate path of $(A_c^s)_{k,j}$. The dashed arrow represents the eliminated entry, while the surrogate path is in solid arrows.

which means that every entry in $A_c^s$ is generated by a sum of three-term multiplications. Now, if $(A_c^a)_{k,i} = 0$ and $(A_c^s)_{k,i} \neq 0$, then we have at least one non-zero term $(R_s)_{k,j} A_{j,\ell} (P_s)_{\ell,i}$ in the right hand side of (14), in which $\ell$ belongs to the aggregate $\mathcal{C}_{m_1}$ and $j$ belongs to the aggregate $\mathcal{C}_{m_2}$, and at least one of them is different from $k$ and $j$ (note that $j$ and $\ell$ are fine-grid variables while the rest are coarse-grid variables). Also,

$$(R_s P_a)_{k,m_2} \neq 0, \quad (R_a P_s)_{m_1,i} \neq 0, \quad \text{and} \quad (A_c^a)_{m_2,m_1} \neq 0. \tag{15}$$

Furthermore, by (11), all these entries also belong to $\mathcal{S}_{\mathcal{P}}(A_c^a)$, and hence can be used together with their associated diagonal entries to eliminate $(A_c^s)_{k,i}$. Overall, in order to eliminate $(A_c^s)_{k,i}$ we add to $A_c^s$ a submatrix of the form

$$\begin{array}{c} \\ i \\ m_1 \\ m_2 \\ k \end{array} \begin{array}{c} \begin{array}{cccc} i & m_1 & m_2 & k \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ -(A_c^s)_{k,i} & 0 & \times & 0 \end{pmatrix} \end{array} \tag{16}$$

where $\times$ denotes a non-zero entry (we excluded $(i,i)$ and $(k,k)$ because they are singles in their row and column, respectively, and hence cannot be changed). Since the marked non-zero entries constitute a distance-three path $i \to m_1 \to m_2 \to k$ in $A_c^a$, we will denote $(i, m_1, m_2, k)$ as the "surrogate path" for eliminating the $(k,i)$ entry. Figure 1 demonstrates this path.

2.2.2. ***Setting values in the surrogate path for eliminating the*** $(k,i)$ ***entry.*** We now describe how to set the values in the submatrix (16) so that $A_c\mathbf{x} = A_c^s\mathbf{x}$ and $(A_c)^T\mathbf{y} = (A_c^s)^T\mathbf{y}$ are satisfied for the right and left strictly positive near null-space "prototypes" $\mathbf{x}$ and $\mathbf{y}$, i.e., (13) is maintained. This is satisfied if $\mathbf{x}_{[i,m_1,m_2,k]}$ and $\mathbf{y}_{[i,m_1,m_2,k]}$ subvectors are the right and left null-space vectors of (16).

Setting $(A_c)_{k,i} = 0$ first breaks the equalities (13) for the $i$-th column and $k$-th row. To satisfy them we must set:

$$\begin{aligned} (A_c)_{m_1,i} &\leftarrow (A_c)_{m_1,i} + (A_c^s)_{k,i}\tfrac{y_k}{y_{m_1}} \\ (A_c)_{k,m_2} &\leftarrow (A_c)_{k,m_2} + (A_c^s)_{k,i}\tfrac{x_i}{x_{m_2}}. \end{aligned} \tag{17}$$

Now, the corresponding equalities (13) for the $m_1$-th row and the $m_2$-th column are broken, so we must satisfy them as well by

$$\begin{aligned} (A_c)_{m_1,m_1} &\leftarrow (A_c)_{m_1,m_1} - (A_c^s)_{k,i}\tfrac{y_k}{y_{m_1}}\tfrac{x_i}{x_{m_1}} \\ (A_c)_{m_2,m_2} &\leftarrow (A_c)_{m_2,m_2} - (A_c^s)_{k,i}\tfrac{y_k}{y_{m_2}}\tfrac{x_i}{x_{m_2}}. \end{aligned} \tag{18}$$

This again breaks the equalities for the $m_1$-th column and $m_2$-th row, and to finally satisfy them both we must apply

$$(A_c)_{m_2,m_1} \leftarrow (A_c)_{m_2,m_1} + (A_c^s)_{k,i}\frac{y_k}{y_{m_2}}\frac{x_i}{x_{m_2}}. \tag{19}$$

Overall, following (17)-(19), the submatrix (16) is given by

$$
\begin{array}{c}
\\
(20)
\end{array}
\qquad
\begin{array}{c|cccc}
 & i & m_1 & m_2 & k \\
i & 0 & 0 & 0 & 0 \\
m_1 & (A_c^s)_{k,i}\frac{y_k}{y_{m_1}} & -(A_c^s)_{k,i}\frac{y_k}{y_{m_1}}\frac{x_i}{x_{m_1}} & 0 & 0 \\
m_2 & 0 & (A_c^s)_{k,i}\frac{y_k}{y_{m_2}}\frac{x_i}{x_{m_2}} & -(A_c^s)_{k,i}\frac{y_k}{y_{m_2}}\frac{x_i}{x_{m_2}} & 0 \\
k & -(A_c^s)_{k,i} & 0 & (A_c^s)_{k,i}\frac{x_i}{x_{m_2}} & 0
\end{array}
$$

and its right and left kernels are $\mathbf{x}_{[i,m_1,m_2,k]}$ and $\mathbf{y}_{[i,m_1,m_2,k]}$ respectively.

In practice, there may be several paths from $i$ to $k$, so we eliminate $(A_c^s)_{k,i}$ using all paths simultaneously, each eliminating a portion $0 < \theta_{(i,m_1,m_2,k)} < 1$ of $(A_c^s)_{k,i}$ (all portions sum to one) . The weights $\theta_{(i,m_1,m_2,k)}$ are chosen proportionally to the strength of the connection in the associated path. More specifically, we use $|(R_aP_s)_{m_1,i}(A_c^a)_{m_2,m_1}(R_sP_a)_{k,m_2}|$ as the strength of the path $(i, m_1, m_2, k)$. However, there are situations where there is a distance-two path between $i$ and $k$, featuring only one connector, i.e., $m_1 = m_2$. In such cases we choose the weight associated with the path $(i, m, m, k)$ as $|(R_aP_s)_{m,i}(R_sP_a)_{k,m}|$ and set the weight of the distance-three paths to 0. Also, the distance-two paths are computationally easier to find . We note that if we consider only symmetric problems, then our distance-two surrogate is similar to that of [14]. Lastly, our sparsening process treats the designated non-zero entries one by one, independently of their order, and hence it can be fully parallelized. A precise description of the sparsening algorithm appears in Algorithm 2.

## 2.3. Theoretical properties of the sparsening procedure.

**Proposition 1.** *If the fine matrix $A$ is symmetric, $R_a = P_a^T$, $R_s = P_s^T$ , and $\mathbf{x} = \mathbf{y}$, then $A_c$ is symmetric as well.*

*Proof.* By the given symmetries and (8), we get that $A_c^s$ and $A_c^a$ are symmetric, and so are their sparsity patterns. Now, if the entry $(k, i)$ needs to be eliminated, i.e., $(k, i) \in \mathcal{S}_{\mathcal{P}}(A_c^s)\backslash\mathcal{S}_{\mathcal{P}}(A_c^a)$, then the entry $(i, k)$ needs to be eliminated as well. Furthermore, if there is a path from $i$ to $k$ such that

$$(P_s^TP_a)_{k,m_2} \neq 0, \quad (P_a^TP_s)_{m_1,i} \neq 0, \quad \text{and} \quad (A_c^a)_{m_2,m_1} \neq 0, \tag{21}$$

then the same path exists from $k$ to $i$ in the opposite direction (i.e., via $m_2$ and $m_1$), because $(P_s^TP_a)^T = (P_a^TP_s)$ and $(A_c^a)$ is symmetric. Using this path, the submatrix (20) that corresponds to the elimination of $(A_c^s)_{i,k}$ is given by:

$$
\begin{array}{c}
\\
(22)
\end{array}
\qquad
\begin{array}{c|cccc}
 & i & m_1 & m_2 & k \\
i & 0 & (A_c^s)_{i,k}\frac{x_k}{x_{m_1}} & 0 & -(A_c^s)_{i,k} \\
m_1 & 0 & -(A_c^s)_{i,k}\frac{x_k}{x_{m_1}}\frac{y_i}{y_{m_1}} & (A_c^s)_{i,k}\frac{x_k}{x_{m_2}}\frac{y_i}{y_{m_1}} & 0 \\
m_2 & 0 & 0 & -(A_c^s)_{i,k}\frac{x_k}{x_{m_2}}\frac{y_i}{y_{m_2}} & (A_c^s)_{i,k}\frac{y_i}{y_{m_2}} \\
k & 0 & 0 & 0 & 0
\end{array}
.
$$

If we set $\mathbf{x} = \mathbf{y}$ and $(A_c^s)_{i,k} = (A_c^s)_{k,i}$ in (22) and in (20), we can see that they are the transpose of each other. This means, that the sum of (20) and (22) is a symmetric submatrix under these conditions. Thus, when eliminating each pair $(i, k)$ and $(k, i)$, we add a symmetric submatrix to $A_c^s$, and therefore $A_c$ remains symmetric. □

For the next proposition, we consider diagonally dominant M-matrices. A matrix $A$ is called an M-matrix if it has the form of $A = sI - B$, where $B \geq 0$ is non-negative and $s \geq \rho(B)$ ($\rho(B) = \max_i\{|\lambda_i|\}$ is the spectral radius of $B$). Furthermore, a matrix $A$ is called diagonally dominant if for every row $i$ we have $A_{i,i} \geq \sum_{j\neq i} |A_{i,j}|$.

```
**Algorithm:** $A_c \leftarrow$ **Sparsify**$(A_c^s, A_c^a, R_sP_a, R_aP_s)$
*% $A_c^s$ - a Petrov-Galerkin smoothed aggregation operator that needs to be sparsified*
*% $A_c^a$ - a Petrov-Gelerkin aggregation operator that defines the result sparsity pattern*
*% $R_sP_a, R_aP_s$ - Aggregated transfer operators. These help establishing surrogate paths.*
*% $A_c$ - SpSA operator - has the sparsity pattern of $A_c^a$ and the quality of $A_c^s$.*
**foreach** $(k,i) \in \mathcal{S}_{\mathcal{P}}(A_c^a)$ **do**
    $(A_c)_{k,i} \leftarrow (A_c^s)_{k,i}$
**end**
Define **x** and **y**: the right and left near null-space prototypes of $A_c^s$.
**foreach** $(k,i) \in \mathcal{S}_{\mathcal{P}}(A_c^s) \backslash \mathcal{S}_{\mathcal{P}}(A_c^a)$ **do**
    *% Try to define a set of distance-two surrogate paths:*
    $\mathcal{S}_{k\to i} = \{(i,m,m,k) : (R_aP_s)_{m,i} \neq 0 \quad \& \quad (R_sP_a)_{k,m} \neq 0\}$.
    **if** $\mathcal{S}_{k\to i} \neq \emptyset$ **then**
        **foreach** $(i,m,m,k) \in \mathcal{S}_{k\to i}$, associate a weight $\theta_{(i,m,m,k)} = |(R_aP_s)_{m,i}(R_sP_a)_{k,m}|$
    **else**
        *% Define a set of distance-three surrogate paths:*
        $\mathcal{S}_{k\to i} = \{(i,m_1,m_2,k) : (R_aP_s)_{m_1,i} \neq 0 \quad \& \quad (A_c^a)_{m_2,m_1} \neq 0 \quad \& \quad (R_sP_a)_{k,m_2} \neq 0\}$.
        **foreach** $(i,m_1,m_2,k) \in \mathcal{S}_{k\to i}$, associate $\theta_{(i,m_1,m_2,k)} = |(R_aP_s)_{m,i}(A_c^a)_{m_2,m_1}(R_sP_a)_{k,m}|$
    **end**
    *% Here, the set of surrogate paths $\mathcal{S}_{k\to i}$ is non-empty*
    Normalize the weights: $\theta_{(i,m_1,m_2,k)} \leftarrow \dfrac{\theta_{(i,m_1,m_2,k)}}{\sum_{(i,p,q,k)\in\mathcal{S}_{k\to i}} \theta_{(i,p,q,k)}}$
    **foreach** $(i,m_1,m_2,k) \in \mathcal{S}_{k\to i}$ **do**
        Define a portion of $(A_c^s)_{k,i}$: $\delta = (A_c^s)_{k,i} \cdot \theta_{(i,m_1,m_2,k)}$.
        Collapse $\delta$ onto the path $(i,m_1,m_2,k)$:
        $(A_c)_{m_1,i} \leftarrow (A_c)_{m_1,i} + \delta\frac{y_k}{y_{m_1}}$,
        $(A_c)_{k,m_2} \leftarrow (A_c)_{k,m_2} + \delta\frac{x_i}{x_{m_2}}$,
        $(A_c)_{m_1,m_1} \leftarrow (A_c)_{m_1,m_1} - \delta\frac{y_k}{y_{m_1}}\frac{x_i}{x_{m_1}}$,
        $(A_c)_{m_2,m_2} \leftarrow (A_c)_{m_2,m_2} - \delta\frac{y_k}{y_{m_2}}\frac{x_i}{x_{m_2}}$,
        $(A_c)_{m_2,m_1} \leftarrow (A_c)_{m_2,m_1} + \delta\frac{y_k}{y_{m_2}}\frac{x_i}{x_{m_2}}$.
    **end**
**end**
```
**Algorithm 2:** The Sparsening procedure

**Proposition 2.** *Let $X = \operatorname{diag}(\mathbf{x})$ and $Y = \operatorname{diag}(\mathbf{y})$ be diagonal matrices whose diagonal entries are given by the vectors $\mathbf{x}$ and $\mathbf{y}$ respectively. Then, if $YA_c^sX$ is a diagonally dominant M-matrix, then $YA_cX$ is a diagonally dominant M-matrix as well.*

*Proof.* By definition, each entry $(k,i)$ of $YA_c^sX$ is given by $(A_c^s)_{k,i}y_kx_i$. Since this matrix is a diagonally dominant M-matrix, then for every row $k$: $(A_c^s)_{k,k}y_kx_k \geq -\sum_{j\neq k}(A_c^s)_{k,j}y_kx_j$, or in matrix form: $(YA_c^sX)1 \geq 0$, where 1 is the vector of ones. By the sparsening construction in Algorithm 2, $(YA_cX)1 = (YA_c^sX)1 \geq 0$. Furthermore, since any off-diagonal entry $(A_c^s)_{k,i}y_kx_i$ is non-positive, the diagonally scaled submatrix (20) for replacing $(A_c^s)_{k,i}$ is given by

$$
\begin{array}{c} \\ i \\ m_1 \\ m_2 \\ k \end{array}
\begin{array}{c} \begin{array}{cccc} i & m_1 & m_2 & k \end{array} \\ \left( \begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ -1 & 0 & 1 & 0 \end{array} \right) \end{array}
\times (A_c^s)_{k,i} y_k x_i. \tag{23}
$$

Now, except for the $(k, i)$ entry which remains zero in $A_c$, all other entries that are added to $Y(A_c^s)X$ are non-positive for an off-diagonal entry or non-negative for a diagonal entry. This means that the sign-structure of $YA_cX$ still corresponds to an M-matrix. This, together with $(YA_cX)1 \geq 0$ means that $(YA_cX)$ is diagonally dominant. Finally, by the Gerschgorin theorem, $YA_cX$ it is also positive definite and hence an M-matrix. □

## 3. Numerical Results

In this section, the sparsening approach is compared to both smoothed aggregation (SA) and simple aggregation (AGG). We consider the 2D and 3D convection-diffusion equation (2) on the unit square/cube with Dirichlet boundary conditions. The problem is discretized using the first order upwind finite differences method, leading to a five-point stencil on a discrete domain. We use the problems in [13, 11]:

$$
\begin{array}{ll}
\texttt{recirc:} & \mathbf{v} = (x(1-x)(2y-1), -(2x-1)y(1-y))^T \\
\texttt{bent-pipe:} & \mathbf{v} = (x(x-2)(1-2y), -4y(y-1)(1-x))^T, \\
\texttt{3D-1:} & \mathbf{v} = (2x(1-x)(2y-1)z, -(2x-1)y(1-y), -(2x-1)(2y-1)z(1-z))^T \\
\texttt{3D-2:} & \mathbf{v} = \left\{ \begin{array}{l} \text{if } \|(x-\frac{1}{2}, y-\frac{1}{2}, z-\frac{1}{2})\|_2 < 0.4, \\ \qquad ((y-\frac{1}{2})(z-\frac{1}{2}), (x-\frac{1}{2})(z-\frac{1}{2}), -2(x-\frac{1}{2})(y-\frac{1}{2}))^T \\ \text{otherwise } 0, \end{array} \right.
\end{array} \tag{24}
$$

and generate $f$ in (2) so that the solution is given by $u = \sin(\pi x)^2 + \sin(\pi y)^2$ for 2D or $u = \sin(\pi x)^2 + \sin(\pi y)^2 + \sin(\pi z)^2$ for 3D.

We use GMRES(5) acceleration, preconditioned with V(1,1) cycles, with one pre-smoothing of forward Gauss-Seidel and one post-smoothing of backward Gauss-Seidel for all methods. For all cases, we start with a zero initial guess and count the number of cycles required to reduce the initial residual by a factor of $10^8$. We also compare operator complexity, $C_{op}$, which is the total number of non-zero elements in the operators $A$ on all the grids, divided by that of the fine-level operator. Our coarsening is performed until $n < 100$.

For all the methods, we use the same Bottom-Up aggregation scheme [17], with a strong connection threshold $\theta = 0.25$. For the 2D problems we use average aggregate sizes $s = 4$, targeting aggregate size of 4, while for 3D we use $s = 6$. For the SA and SpSA methods, we calculate $Q$ according to (6). For the sparsening procedure (Algorithm 2), we use $\mathbf{x} = \mathbf{y} = 1$, that is related to the piece-wise constant operators (4). Also, we use the following damping parameters: for SA we use $\omega = 0.6$ for all tests; for SpSA we use $\omega = 0.8$ in the 2D tests, and $\omega = 0.7$ in the 3D tests. In addition, we also use a moderate overcorrection of 1.1 for all methods.

Table 1 compares the three aggregation methods AGG, SA and SpSA for the 2D convection-diffusion (2) with the first two velocity fields in (24). It is clear that the AGG method is not mesh independent and not efficient, especially for the `recirc` problem. It also struggles for the diffusion-dominated `bent-pipe` problem ($\epsilon = 10^{-2}$). The SA method shows good scalability for all combinations, but has a rather high operator complexity, especially in the more convective problems, where the aggregation becomes more moderate. The SpSA obviously has the rather low

| $\epsilon$ | $n$ | `recirc` | | | | | | `bent-pipe` | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *AGG* | | *SpSA* | | *SA* | | *AGG* | | *SpSA* | | *SA* | |
| | | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ |
| $10^{-2}$ | $256^2$ | 47 | 1.33 | 9 | 1.33 | 10 | 2.18 | 58 | 1.33 | 10 | 1.33 | 12 | 2.11 |
| | $512^2$ | 62 | 1.33 | 10 | 1.33 | 10 | 2.22 | 78 | 1.33 | 10 | 1.33 | 13 | 2.13 |
| | $1024^2$ | 79 | 1.33 | 10 | 1.33 | 11 | 2.22 | >100 | 1.33 | 11 | 1.33 | 13 | 2.13 |
| $10^{-4}$ | $256^2$ | >100 | 1.64 | 14 | 1.64 | 13 | 3.23 | 29 | 1.72 | 15 | 1.73 | 13 | 3.79 |
| | $512^2$ | >100 | 1.49 | 15 | 1.51 | 14 | 2.76 | 46 | 1.66 | 15 | 1.68 | 13 | 3.60 |
| | $1024^2$ | >100 | 1.33 | 19 | 1.35 | 16 | 2.23 | 74 | 1.59 | 16 | 1.60 | 14 | 3.19 |
| $10^{-6}$ | $256^2$ | 95 | 1.81 | 18 | 1.82 | 16 | 3.22 | 28 | 1.77 | 16 | 1.78 | 15 | 3.14 |
| | $512^2$ | >100 | 1.81 | 20 | 1.82 | 18 | 3.21 | 35 | 1.76 | 18 | 1.77 | 16 | 3.13 |
| | $1024^2$ | >100 | 1.79 | 23 | 1.80 | 19 | 3.18 | 44 | 1.75 | 21 | 1.76 | 18 | 3.10 |

Table 1. Comparison of AGG, SA and SpSA for 2D convection-diffusion. $\#it$ denotes the number of V(1,1) cycles needed for convergence, while $C_{op}$ is the operator complexity.

| $\epsilon$ | $n$ | `3D-1` | | | | | | `3D-2` | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *AGG* | | *SpSA* | | *SA* | | *AGG* | | *SpSA* | | *SA* | |
| | | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ | $\#it$ | $C_{op}$ |
| $10^{-2}$ | $64^3$ | 38 | 1.38 | 14 | 1.40 | 13 | 3.22 | 36 | 1.34 | 15 | 1.35 | 14 | 3.08 |
| | $96^3$ | 47 | 1.41 | 15 | 1.42 | 14 | 3.41 | 53 | 1.35 | 16 | 1.35 | 15 | 3.20 |
| | $128^3$ | 55 | 1.41 | 15 | 1.42 | 18 | 3.50 | 65 | 1.34 | 18 | 1.34 | 19 | 3.24 |
| $10^{-4}$ | $64^3$ | 24 | 1.61 | 12 | 1.62 | 12 | 6.49 | 34 | 1.38 | 14 | 1.39 | 12 | 3.16 |
| | $96^3$ | 31 | 1.58 | 12 | 1.59 | 12 | 6.44 | 54 | 1.39 | 15 | 1.40 | 14 | 3.27 |
| | $128^3$ | 37 | 1.56 | 12 | 1.57 | 13 | 6.07 | 72 | 1.39 | 15 | 1.40 | 17 | 3.39 |
| $10^{-6}$ | $64^3$ | 23 | 1.66 | 14 | 1.68 | 13 | 5.22 | 23 | 1.38 | 11 | 1.40 | 9 | 3.15 |
| | $96^3$ | 28 | 1.65 | 14 | 1.67 | 14 | 5.45 | 33 | 1.39 | 13 | 1.40 | 11 | 3.30 |
| | $128^3$ | 32 | 1.65 | 15 | 1.66 | 14 | 5.39 | 41 | 1.39 | 14 | 1.40 | 13 | 3.37 |

Table 2. Comparison of AGG, SA and SpSA for 3D convection-diffusion. $\#it$ denotes the number of V(1,1) cycles needed for convergence, while $C_{op}$ is the operator complexity.

operator complexity of AGG, but also enjoys the attractive and scalable convergence behavior of SA for both the diffusive and convective problems. It seems to be the best option of the three.

Table 2 shows the results for the 3D problems. Although we used larger aggregates, we see even larger operator complexities for SA than in 2D, especially for the convection-dominated problems ($\epsilon = 10^{-4}, 10^{-6}$). The convergence of SA is rather good and scalable, as in the 2D case. Unlike in 2D, the AGG method shows moderate convergence, albeit not mesh-independent, especially for the convective problems. Its convergence is expected to further deteriorate as the problem gets bigger. As in 2D, SpSA has the low operator complexity of AGG and convergence similar to SA. We note that the SpSA setup is more expensive in 3D than in 2D because more non-zeros need to be eliminated using more surrogate paths. Reducing the number of surrogate paths is one subject of our future research. Other than that, the SpSA method again seems to be the most efficient.

## 4. Conclusions

In this paper we have presented a new algebraic multigrid algorithm where the choice of the sparsity pattern of the coarse operators is independent of the choice of the high-quality transfer operators. This property makes the algorithm particulary worthwhile for parallel settings.

The new algorithm uses the well-known aggregation framework, adopting simple non-smoothed aggregation for determining the sparsity pattern of the coarse operators, and smoothed aggregation for high-quality transfer operators. It sparsifies the smoothed aggregation coarse operators onto the simple aggregation sparsity patterns. Numerical experiments show that the algorithm has promising capabilities for 2D and 3D convection-diffusion problems. It seems scalable and robust and may be advantageous in cases where strict sparsity constraints prevent us from using high-quality Galerkin operators. We expect that our algorithm will have similar performance for such problems also on unstructured settings.

## 5. Acknowledgements

The author would like to thank Prof. Irad Yavneh for his valuable help and guidance throughout this work. The author would also like to thank Dr. Jacob Schroder for a fruitful conversation that inspired the non-symmetric sparsening approach.

## References

[1] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation ($\alpha$ SA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[2] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders, *Towards adaptive smooth aggregation ($\alpha$ SA) for nonsymmetric problems*, SIAM J. Sci. Comput., 32 (2010), pp. 14–39.

[3] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, SIAM, second ed., 2000.

[4] O. Brker and M. J. Grote, *Sparse approximate inverse smoothers for geometric and algebraic multigrid*, Applied Numerical Mathematics, 41 (2002), pp. 61–80.

[5] E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang, *A Survey of Parallelization Techniques for Multigrid Solvers*, in Parallel Processing for Scientific Computing, 2006.

[6] R. D. Falgout, *An introduction to algebraic multigrid*, IEEE: Computing in Science and Engineering, 8 (2006), pp. 24–33.

[7] H. Guillard and P. Vaněk, *An aggregation multigrid solver for convection-diffusion problems on unstructured meshes.*, Tech. Report UCD-CCM-130, University of Colorado at Denver, CO, USA, 1998.

[8] H. De Sterck, T. A. Manteuffel, S. F. McCormick, K. Miller, J. Pearson, J. Ruge, and G. Sanders, *Smoothed aggregation multigrid for markov chains*, SIAM J. Sci. Comput., 32 (2010), pp. 40–61.

[9] Y. Notay, *Aggregation-based algebraic multilevel preconditioning*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 998–1018.

[10] ———, *An aggregation-based algebraic multigrid method*, Electronic Transactions on Numerical Analysis, 37 (2010), pp. 123–146.

[11] ———, *Aggregation-based algebraic multigrid for convection-diffusion equations*, SIAM J. Sci. Comput., 34 (2012), pp. A2288–A2316.

[12] C. W. Oosterlee and T. Washio, *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flow*, SIAM J. Sci. Comput., 21 (2000), pp. 1670–1690.

[13] M. Sala and R. S. Tuminaro, *A new petrov-galerkin smoothed aggregation preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 31 (2008), pp. 143–166.

[14] J. Schroder and R. Falgout, *Non-Galerkin coarse-grid operators for AMG*, 12th Copper Mountain Conference on Iterative Methods, (2012).

[15] H. D. Sterck, U. M. Yang, and J. J. Heys, *Reducing complexity in parallel algebraic multigrid preconditioners*, SIAM J. Matrix Anal. Appl, 27 (2006), pp. 1019–1039.

[16] E. Treister and I. Yavneh, *Square and stretch multigrid for stochastic matrix eigenproblems*, Numerical Linear Algebra with Application, 17 (2010), pp. 229–251.

[17] ———, *On-the-fly adaptive smoothed aggregation multigrid for markov chains*, SIAM Journal on Scientific Computing (SISC), 33 (2011), pp. 2927–2949.

[18] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London and San Diego, 2001.

[19] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[20] R. Wienands and I. Yavneh, *Collocation coarse approximation in multigrid*, SIAM J. Sci. Comput., 31 (2009), pp. 3643–3660.

[21] I. Yavneh, *Why multigrid methods are so efficient*, IEEE: Computing in Science and Engineering, 8 (2006), pp. 12–22.

[22] R. Zemach, E. Treister, and I. Yavneh, *Algebraic collocation coarse approximation (ACCA) multigrid*, 12th Copper Mountain Conference on Iterative Methods, (2012).