SQUARE & STRETCH MULTIGRID FOR STOCHASTIC MATRIX EIGENPROBLEMS

ERAN TREISTER AND IRAD YAVNEH

Abstract. A novel multigrid algorithm for computing the principal eigenvector of column stochastic matrices is developed. The method is based on the *Exact Interpolation Scheme* multigrid approach of Brandt and Ron [3], whereby the prolongation is adapted to yield a better and better coarse representation of the sought eigenvector. The main novelty of the present approach is in the squaring of the stochastic matrix—followed by a stretching of its spectrum—just prior to the coarse-grid correction process. This procedure is shown to yield good convergence properties, even though a cheap and simple aggregation is used for the restriction and prolongation matrices, which is important for maintaining competitive computational costs. A further contribution of this paper is a novel bottom-up procedure for defining the coarse-grid aggregates.

1. Introduction

Fast numerical solvers for eigenproblems are in demand in many disciplines. Multigrid methods are efficient in such problems for certain types of matrices, especially for discretized elliptic partial differential operators [2, 8, 9] and also more general types of M-matrices [1]. Recently, Brandt and Ron [3] have suggested an adaptive multigrid approach whereby the solution itself is approximated on the coarser levels, rather than the error as in classical multigrid. This approach was dubbed *Exact Interpolation Scheme* (EIS), because it requires that the prolongation operator be consistently improved as the iterations progress until, ultimately, one obtains an arbitrarily accurate solution by prolongating a smaller vector.

In this paper, we present an efficient algorithm, based on EIS, for computing the principal eigenvector of column-stochastic matrices. This problem has drawn great recent attention, largely due to its relevance in web search application (via Google's PageRank algorithm) and in Markov chain processes. Among the various approaches that have addressed this problem [6, 7, 10], the multigrid approach of [4, 5] is the closest approach to the one presented here.

1.1. **The Stochastic Matrix Eigenproblem.**

Let $B = (B_{ij})_{i,j=1}^{n}$ be a given irreducible sparse column-stochastic matrix, that is, for every column j, $\sum_{i=1}^{n} B_{ij} = 1$, and all the elements of B are non–negative. By the Perron–Frobenius theorem there exists a unique vector $\mathbf{x}$ with strictly positive entries which satisfies $B\mathbf{x} = \mathbf{x}$. Note that $\rho(B) = 1$, where ρ denotes the spectral radius. Our objective is to compute the null–space vector of $B - I$, i.e., the vector $\mathbf{x}$ which satisfies

$$B\mathbf{x} = \mathbf{x}. \tag{1.1}$$

In the analysis below, we assume real eigenvalues (as in symmetric matrices), hence all in the interval $[-1, 1]$. Nevertheless, our experiments demonstrate efficiency also for nonsymmetric problems.

Department of Computer Science, Technion—Israel Institute of Technology, eran@tx.technion.ac.il. Research supported by the Israeli Science Foundation, grant number 795/08.

2. Algorithm Description

2.1. **The EIS Approach.** Suppose that we could construct some prolongation operator P of size $n \times n_c$, with $n_c < n$, such that the solution $\mathbf{x}$ is in its range, i.e., $\mathbf{x} = P\mathbf{x}_c$, for some vector $\mathbf{x}_c$ of size n_c. Then, defining a suitable restriction operator R of size $n_c \times n$, we obtain by substituting $P\mathbf{x}_c$ for $\mathbf{x}$ in equation (1.1) and multiplying through by R,

$$RBP\mathbf{x}_c = RP\mathbf{x}_c \,. \tag{2.1}$$

If we choose the operators such that $RP = I_c$, where I_c is the $n_c \times n_c$ identity matrix, then the coarse grid problem (2.1) is of the same form as the original fine grid problem:

$$B_c\mathbf{x}_c = \mathbf{x}_c, \tag{2.2}$$

where $B_c = RBP$ is the coarse grid operator. After solving (2.2), we obtain the sought solution by prolongation: $\mathbf{x} = P\mathbf{x}_c$. This motivates the following algorithm.

2.2. **Basic EIS Two-Level Algorithm.** Given an initial guess, $\mathbf{x}^0$, do for $k = 1, 2, \ldots$, until some convergence criterion is satisfied:

(1) Apply ν_1 *pre-relaxations* to (1.1): $\mathbf{x}^k \leftarrow Relax(\mathbf{x}^k, B, \nu_1)$
(2) Define an n by n_c prolongation matrix, P, and an n_c by n restriction matrix, R, such that $RP = I_c$ and $PR\mathbf{x}^k = \mathbf{x}^k$.
(3) Compute $B_c = RBP$, and solve equation (2.2), obtaining $\mathbf{x}_c$.
(4) Compute $\mathbf{x}^{k+1} = P\mathbf{x}_c$.
(5) Apply ν_2 *post-relaxations*: $\mathbf{x}^{k+1} \leftarrow Relax(\mathbf{x}^{k+1}, B, \nu_2)$

2.3. **Relaxation.** Let D denote the diagonal matrix comprised of the diagonal of the singular M-matrix $I - B$. Then weighted Jacobi relaxation is defined by the iteration

$$\mathbf{x}^{k+1} = \left[I - \omega D^{-1}(I - B)\right]\mathbf{x}^k \,, \tag{2.3}$$

where ω is a positive weighting parameter (typically damping, that is, $\omega < 1$.) The effectiveness of the relaxation as a *s*olver for (1.1) is generally determined by the ratio between the largest eigenvalue of B, which is 1, and the modulus of the eigenvalue that is second-largest in magnitude. Typically, especially for large problems of interest, this ratio is very close to 1, so the relaxation converges slowly. More generally, damped Jacobi relaxation is highly effective for reducing eigenvectors with relatively large eigenvalues of $I - B$, corresponding to negative and small positive eigenvalues of B. These are called the *rough* eigenmodes. On the other hand, the *smooth* eigenmodes—those corresponding to eigenvalues of B that are close to 1—are hardly affected by the relaxation.

2.4. **Prolongation and Restriction operators.** Our task is to construct a prolongation matrix, P, such that the exact solution, $\mathbf{x}$, is approximately in its range, and this approximation should improve as the solution process progresses. Contrary to classical algebraic multigrid, here the "setup" is not done just once—P and R and B_c need to be updated during each multigrid cycle. Hence, it is imperative to maintain low-cost operators, and we therefore aim to employ simple aggregation. We first partition the fine grid index set $\mathcal{N} = \{1, \ldots, n\}$, into n_c aggregates $\{\mathcal{C}_I\}_{I=1}^{n_c}$ (which are simply disjoint subsets of $\mathcal{N}$.) The formation of these aggregates is explained later.

2.4.1. *Transfer Operators.* The prolongation (disaggregation) matrix P (of size $n \times n_c$), and the restriction (aggregation) matrix R (of size $n_c \times n$) are defined by:

$$R_{J,i} = \begin{cases} 1 & \text{if } i \in \mathcal{C}_J \\ 0 & \text{otherwise} \end{cases} \qquad P_{i,J} = \begin{cases} \mathbf{x}_i^k / \left(\mathbf{x}_c^k\right)_J & \text{if } i \in \mathcal{C}_J \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

with $\left(\mathbf{x}_c^k\right)_J = \left(R\mathbf{x}^k\right)_J = \sum_{r \in \mathcal{C}_J} \mathbf{x}_r^k$.

Denote by X^k and X_c^k the diagonal square matrices whose diagonals are given by $\mathbf{x}^k$ and $\mathbf{x}_c^k$, respectively. Observe that $X_c^k = RX^kR^T$. Using this notation, we can write:

$$P = X^k R^T (X_c^k)^{-1} .$$

Clearly, $RP = X_c^k(X_c^k)^{-1} = I_c$, and

$$PR\mathbf{x}^k = P\mathbf{x}_c^k = X^k R^T (X_c^k)^{-1}\mathbf{x}_c^k = X^k R^T \mathbf{1}_c = X^k \mathbf{1} = \mathbf{x}^k,$$

where $\mathbf{1}$ and $\mathbf{1}_c$ are vectors of ones of size n and n_c, respectively. Thus, P and R satisfy the conditions of step 2 of the basic EIS algorithm described above.

Note in (2.4) that the disaggregation operator P distributes each aggregate's value amongst the fine-grid elements belonging to the aggregate, with relative weights proportional to the corresponding elements in the current approximation $\mathbf{x}^k$. The main heuristic observation regarding the *smoothing effect* of damped Jacobi relaxation is that, for a sparse matrix B, the *ratio* between "strongly connected neighbors" in the current solution $\mathbf{x}^k$ quickly tends to the corresponding ratio in the exact solution $\mathbf{x}$. That is, if $B_{i,j}$ is relatively large in comparison to other terms in the ith row of B (i and j are strongly connected) then, after several relaxations, the ratio between the ith and jth elements of the current approximation tends to be close to the corresponding ratio in $\mathbf{x}$. We strive to create aggregates comprised of strongly connected elements. Thus, by this smoothing effect of damped Jacobi, the relaxation and coarse grid correction fulfill complementary roles: relaxation causes the values within each aggregate to tend to the correct relations, and coarse-grid correction corrects the value of each aggregate.

2.4.2. *Aggregation—a bottom-up approach.* The bottom-up aggregation approach we now present is an alternative to the (more typical) top-down approach of [4, 5]. Both are motivated by the same notion of strength of connection. However, our approach is to first make sure that the small-valued variables are represented well in the coarse system, and only later tend to the large ones. The reason for this is that strong variables will naturally be selected to aggregates, whereas weak ones will not, unless we explicitly require it. We therefore do this early on, to avoid the situation where many unattached variables need to be added to existing aggregates at the end of the process.

We denote by $\hat{S}$ the connectivity matrix of BX^k, where $X^k = diag(\mathbf{x}^k)$, as defined above:

$$\hat{S}_{ij} = \begin{cases} B_{ij}\mathbf{x}_j^k & \text{if } i \neq j \text{ and } B_{ij}\mathbf{x}_j^k \geq \theta \max_{l \neq i} B_{il}\mathbf{x}_l^k \, , \\ 0 & \text{otherwise,} \end{cases}$$

where $\theta \in [0, 1]$ is a threshold parameter. In this paper, we choose $\theta = 0.1$. Note that a binary version of this matrix is used in the coarsening procedure of [4, 5].

Our goal is to form aggregates that contain elements which are strongly connected. Since there is no "preferred element" in the aggregate, we choose to disregard the directions of the dependencies within the aggregate. Therefore, we symmetrize our connectivity matrix as follows.

$$S = \frac{1}{2}\left(\hat{S} + \hat{S}^T\right) . \tag{2.5}$$

Our algorithm aims to find the most strongly connected aggregates of size equal to or less than a given parameter s. To this end, it searches for groups of s (or less) points which are closest to forming a clique, and at least have a circular dependency of all the members of the aggregate. We define a circle of length s as a set of s members $\{i_1, ..., i_s\}$ where $S_{i_j,i_{j+1}} > 0$ for all $1 \leq j \leq s - 1$ and $S_{i_s,i_1} > 0$. Note by (2.5) that if $S_{ij} > 0$ then i and j form a circle of size 2. We also define the weight of an aggregate as the sum of the elements of the connectivity submatrix corresponding to its members:

$$w(\mathcal{C}_J) = \sum_{i,j \in \mathcal{C}_J} S_{ij}. \tag{2.6}$$

Our second aim is to limit the operator complexity, defined as the total number of non-zero elements divided by the number of nonzero elements of the fine-level operator. The tradeoff of low complexity versus better convergence is controlled by the parameter s. Small s means higher complexity and, in general, better convergence rates per cycle, and vice versa. Our approach in this paper is to simply choose for each problem the smallest value of s that keeps the operator complexity sufficiently low.

```
Algorithm: Bottom-up(s)
repeat
    Among the unassigned elements, find the index $i$ of the smallest element in the current
    approximation $\mathbf{x}^k$.
    Find all circles of length $s$ or less that contain $i$.
    if Circles found then
        Choose the circles of maximal length, and amongst those aggregate the circle of
        maximal weight as defined in (2.6).
        Remove the aggregate's members from $S$.
    else
        Let $p = \arg\max_{j \neq i}\{B_{i,j}\mathbf{x}_j^k\}$.
        Assign $i$ to $p$'s aggregate, and remove it from $S$.
    end
until all elements are assigned to aggregates
```

The *Bottom-up(s)* algorithm may be implemented in various ways. We use a Depth-First-Search technique (starting from seed i) in order to seek circles of length s. As this procedure might be a bit complex, we may, following a suggestion in [4, 5], calculate aggregates only once or twice throughout the whole iterative process and then freeze them. In this work, when $s > 2$, we calculate the aggregates twice: once in the beginning of the process, and once after the current solution is stable enough (fourth cycle, for example).

2.4.3. *Coarse Grid operators.* The coarse matrix $B_c = RBP$ maintains the column-stochastic property of the fine-grid matrix B. It is easy to see that $\mathbf{1}^T P = \mathbf{1}_c^T$ and $\mathbf{1}_c^T R = \mathbf{1}^T$, and since $\mathbf{1}^T B = \mathbf{1}^T$, then

$$\mathbf{1}_c^T B_c = \mathbf{1}_c^T RBP = \mathbf{1}^T BP = \mathbf{1}_c^T.$$

In addition, since all elements in R and P are non-negative, then so are those of B_c. Thus, B_c is also a column-stochastic matrix. Furthermore, if B is irreducible, then so is B_c. This means that we can continue recursively and obtain a well-defined multi-level process.

2.5. Properties of the Basic EIS Algorithm.

The basic algorithm converges to the solution in all problems we tested. However, as observed in [5], the simple low-order prolongation produces sharp differences between aggregates, because all elements of each aggregate receive the same multiplicative correction. Rough eigenmodes are thus amplified, resulting in slow convergence in many problems, especially for discretized elliptic partial differential operators. This effect is related to the well-known requirements on the high-frequency orders of transfer operators in classical multigrid (see, e.g., [11]). The usual remedy is to use higher-order transfers. Indeed, this approach is adopted in [5], employing smoothed aggregation. While often effective, there are drawbacks to this approach, including a significant increase in the cost of constructing the coarse operators, emergence of negative elements in the coarse operator (overcome in [5] by so-called lumping), and deterioration for some common nonsymmetric problems such as convection dominated operators.

We next present an alternative approach, which seems simpler (retaining the simple transfer operators) and appears to yield quite competitive performance, though a detailed comparison between the approaches would require first optimizing each of them, and this has yet to be performed.

2.6. **The Square & Stretch Algorithm.** The slow convergence brought about by low-order transfers is due to the generation and amplification of rough eigenmodes by P (and lack of sufficient damping of such modes by R). These rough eigenmodes generate large residuals, and their coupling with the smooth eigenmodes leads to relatively poor carse-grid approximation of the latter. To overcome this, we simply square the matrix B before coarse-grid correction, defining $B_c = RB^2P$. Clearly, B^2 is column stochastic and retains the same principal eigenvector as B, but it has no negative eigenvalues[1]. Thus, the "roughest" eigenmodes of B (with eigenvalues close to -1) no longer generate large residuals, as they are smooth with respect to B^2 (with eigenvalues close to 1.) This leads to much better two-level convergence rates, even though we continue to use the same P and R operators.

Now, however, a new problem arises: we cannot continue this approach recursively, because the spectrum of B_c is approximately in the range $[0, 1]$, with the rough modes having small positive eigenvalues, so squaring the matrix once again will not be useful. We therefore stretch the spectrum without changing the eigenvectors by replacing B_c by the matrix $\hat{B}_c$ defined as follows:

$$\hat{B}_c = R\hat{B}P\,, \text{ with } \hat{B} = \left(\frac{1}{1-d}\right) B^2 - \left(\frac{d}{1-d}\right) I\,, \tag{2.7}$$

where d is the stretching parameter. Observe that we have stretched the spectrum, reducing the smallest eigenvalue to (approximately) $\frac{-d}{1-d}$ instead of approximately 0. Note that the solution, i.e., the eigenvector corresponding to the eigenvalue 1, has not changed.

2.6.1. *The stretching parameter.* Theoretically, the stretching parameter d should be the largest value that still keeps the spectral radius of $\hat{B}_c$ from being higher than 1. Knowing this requires a tight bound on the smallest (most negative) eigenvalue of RB^2P, which is not generally known. One option we test is a predefined constant value for all coarse grids. Note, however, that this may introduce some negative entries in $\hat{B}_c$, and thus ruin its column-stochastic properties, though the column-sum remains 1.

A safe alternative is given by $d = \min_J\{(RB^2P)_{J,J}\}$. Evidently, the new coarse grid matrix $\hat{B}_c$ remains irreducible and column stochastic. Normally, this value of d is still substantial, as the Jth diagonal term of B^2 is a weighted sum of the elements of the submatrix of B corresponding to fine variables that belong to the aggregate J. All such elements are positive and relatively large, since the aggregates are comprised of strongly connected variables. This choice of d corresponds to using the lower bound on the smallest eigenvalue calculated according to the well-known *Gerschgorin disks theorem*, applied to the columns of RB^2P.

3. Numerical Results

We compare numerically the basic EIS aggregation multigrid method (Aggr) to the Square and Stretch (S&S) algorithm for nearly all the test problems of [5]. We use a random positive $\mathbf{x}^0$ as an initializer, and damped Jacobi relaxation with $\nu_1 = 2, \nu_2 = 2$, alternating between $\omega = 0.5$ and $\omega = 1$ (but 0.98 on the finest grid). We begin with 20 relaxation sweeps in order to obtain a relevant initial approximation for deriving P, and we count this work as a cycle in our results. In the future, this will be replaced by a full multigrid approach, beginning with a uniform guess for the solution. We reduce the l_1 residual norm to 10^{-8}, but stop after 20 cycles if this goal is not achieved (marked as '>20'). We stop coarsening when $n < 16$, where we solve the problem

[1]Recall that we only consider real-valued eigenvalues in this discussion.

directly. For the S&S method, d represents the stretching parameter of (2.7), where *MinDiag* refers to choosing $d = \min_J\{(RB^2P)_{J,J}\}$; the second option we test is a constant $d = 0.5$. This constant is motivated by (2.7), where it is seen that if the eigenvalues of B are real and in the range $[-1, 1]$, then this also holds for $\hat{B}$, so $d = 0.5$ is the most aggressive possible stretching parameter that is expected not to change the spectral radius. 'γ' denotes the geometric mean convergence factor per cycle, computed over the last five cycles. 'C_{op}' is the total number of non-zero elements in the operators B on all the grids, divided by that of the fine-level operator. *Values in brackets* refer to F-cycles rather than V-cycles. All structured problems are described by their stencils of B, denoted by H with a descriptive subscript. However, the coefficients are normalized at the boundaries, in order to yield a stochastic matrix.

3.1. **1D Problem Set.**

The first three problems are one-dimensional Markov chains generated by linear graphs with weighted edges. In all the examples here, each of the two boundary nodes has only one outgoing edge with a weight of 1. Here we use the bottom up aggregation method with $s = 2$, performed in every cycle, as its cost is low. We do not show a comparison of *bottom-up* to "geometric" aggregation in these problems, as both aggregations produce similar results.

3.1.1. *1D uniform chain.* The first problem is the 1D uniform chain which is the simplest one. The operator stencil is given by

$$H_{Uniform1D} = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}.$$

Table 1 summarizes the results.

n	Method	Aggregation	d	*#levels*	*#cycles*	C_{op}	γ
256	Aggr	BottomUp(2)	—	6	>20 (>20)	1.90 (3.39)	0.85 (0.73)
256	S&S	BottomUp(2)	*MinDiag*	6	7 (6)	1.89 (3.21)	0.08 (0.05)
256	S&S	BottomUp(2)	0.5	6	7 (6)	1.73 (2.97)	0.08 (0.04)
1024	Aggr	BottomUp(2)	—	7	>20 (>20)	1.96 (3.81)	0.9 (0.85)
1024	S&S	BottomUp(2)	*MinDiag*	7	7 (6)	1.96 (3.65)	0.10 (0.04)
1024	S&S	BottomUp(2)	0.5	7	7 (6)	1.79 (3.20)	0.08 (0.05)
4096	Aggr	BottomUp(2)	—	9	>20 (>20)	1.98 (3.91)	0.91 (0.88)
4096	S&S	BottomUp(2)	*MinDiag*	9	8 (6)	1.97 (3.75)	0.15 (0.05)
4096	S&S	BottomUp(2)	0.5	9	7 (6)	1.80 (3.30)	0.08 (0.04)

TABLE 1. 1D uniform chain

3.1.2. *Birth-death chain.* The second problem is a Birth-Death chain with the same constant birth and death rates as in [5]. The stencil is given by

$$H_{Birth-Death} = \begin{pmatrix} \frac{1}{1+\mu} & 0 & \frac{\mu}{1+\mu} \end{pmatrix},$$

with $\mu = 0.96$. Table 2 summarizes the results for this problem. Here, using $d = 0.5$ did not yield any improvement. Furthermore, in the bigger example (4096), this choice failed.

3.1.3. *1D uniform chain with two weak links.* The third problem is a 1D uniform chain with two weak links of weight ε in its middle. This means that there exist nodes i and $i + 1$ where the two edges between them are of weight ε. Here we use $\varepsilon = 0.001$ as in [5]. The results are summarized in Table 3.

3.2. **2D Problem Set.**

We next show results for two dimensional problems on a uniform mesh. As before, in all the examples, the outgoing edges of the boundary nodes are weighted proportionally to the stencil weights to yield a sum of 1.

n	Method	Aggregation	d	$\#levels$	$\#cycles$	C_{op}	γ
256	Aggr	BottomUp(2)	—	6	>20 (>20)	1.91 (3.53)	0.93 (0.78)
256	S&S	BottomUp(2)	*MinDiag*	6	7 (6)	1.92 (3.50)	0.08 (0.06)
256	S&S	BottomUp(2)	0.5	6	7 (6)	1.92 (3.53)	0.07 (0.05)
1024	Aggr	BottomUp(2)	—	7	>20 (>20)	1.97 (3.78)	0.95 (0.82)
1024	S&S	BottomUp(2)	*MinDiag*	7	7 (7)	1.97 (3.82)	0.08 (0.07)
1024	S&S	BottomUp(2)	0.5	7	7 (7)	1.98 (3.84)	0.08 (0.06)
4096	Aggr	BottomUp(2)	—	9	>20 (>20)	1.97 (3.78)	0.95 (0.82)
4096	S&S	BottomUp(2)	*MinDiag*	9	7 (6)	1.98 (3.88)	0.1 (0.07)

Table 2. Birth-Death chain with $\mu = 0.96$

n	Method	Aggregation	d	$\#levels$	$\#cycles$	C_{op}	γ
256	Aggr	BottomUp(2)	—	6	>20 (>20)	1.87 (3.38)	0.89 (0.77)
256	S&S	BottomUp(2)	*MinDiag*	6	7 (6)	1.89 (3.21)	0.08 (0.05)
256	S&S	BottomUp(2)	0.5	6	7 (6)	1.73 (2.92)	0.08 (0.04)
1024	Aggr	BottomUp(2)	—	7	>20 (>20)	1.97 (3.78)	0.91 (0.86)
1024	S&S	BottomUp(2)	*MinDiag*	7	7 (6)	1.79 (3.16)	0.08 (0.04)
1024	S&S	BottomUp(2)	0.5	7	7 (6)	1.78 (3.19)	0.08 (0.04)
4096	Aggr	BottomUp(2)	—	9	>20 (>20)	1.98 (3.89)	0.92 (0.86)
4096	S&S	BottomUp(2)	*MinDiag*	9	8 (6)	1.97 (3.72)	0.15 (0.04)
4096	S&S	BottomUp(2)	0.5	9	7 (6)	1.80 (3.28)	0.08 (0.05)

Table 3. 1D uniform chain with two weak links in the middle

3.2.1. *Uniform 2D lattice.* The first test is the uniform 2D lattice with stencil given by

$$H_{Uniform-2D} = \frac{1}{4}\begin{pmatrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{pmatrix}.$$

In this problem, when using the S&S method, we use the bottom up aggregation method with $s = 4$, as smaller aggregation size leads to relatively high operator complexity. The aggregation is calculated in the first and fourth multigrid iterations only. In addition, we provide results for a geometric 2D uniform aggregation of size 2×2. Table 4 summarizes these results.

3.2.2. *Anisotropic 2D lattice.* The next problem is the 2D lattice with anisotropic weights:

$$H_{Anisotropic-2D} = \frac{1}{2+2\varepsilon}\begin{pmatrix} & \varepsilon & \\ 1 & 0 & 1 \\ & \varepsilon & \end{pmatrix},$$

where $\varepsilon = 1e-6$. In this problem, the aggregation occurs first along the strong connections and only later along the weak ones. We employ the bottom up aggregation method with $s = 2$, although it might lead to relatively high complexity. (Note that there are no circles longer than 2, so using larger s would not lead to any significant difference.) As in the 1D problems, the S&S convergence rates are excellent, and the F-cycle, which increases the complexity, is not worthwhile. Table 5 summarizes these results.

n	Method	Aggregation	d	$\#levels$	$\#cycles$	C_{op}	γ
256	Aggr	BottomUp(2)	—	9	>20 (>20)	1.99 (3.58)	0.72 (0.64)
256	S&S	BottomUp(4)	*MinDiag*	4	>20 (14)	1.53 (2.22)	0.58 (0.40)
256	S&S	Geometric	*MinDiag*	3	17 (12)	1.47 (2.03)	0.47 (0.33)
256	S&S	BottomUp(4)	0.5	4	14 (12)	1.56 (2.35)	0.39 (0.29)
256	S&S	Geometric	0.5	3	13 (12)	1.48 (2.04)	0.37 (0.31)
1024	Aggr	BottomUp(2)	—	9	>20 (>20)	2.09 (3.95)	0.83 (0.70)
1024	S&S	BottomUp(4)	*MinDiag*	5	>20 (14)	1.70 (2.66)	0.70 (0.42)
1024	S&S	Geometric	*MinDiag*	4	19 (13)	1.54 (2.20)	0.57 (0.36)
1024	S&S	BottomUp(4)	0.5	5	16 (12)	1.68 (2.58)	0.49 (0.33)
1024	S&S	Geometric	0.5	4	15 (12)	1.54 (2.22)	0.45 (0.33)
4096	Aggr	BottomUp(2)	—	8	>20 (>20)	2.06 (4.27)	0.88 (0.79)
4096	S&S	BottomUp(4)	*MinDiag*	6	>20 (15)	1.73 (2.78)	0.76 (0.52)
4096	S&S	Geometric	*MinDiag*	5	20 (13)	1.53 (2.31)	0.66 (0.37)
4096	S&S	BottomUp(4)	0.5	6	19 (12)	1.70 (2.83)	0.59 (0.34)
4096	S&S	Geometric	0.5	5	16 (12)	1.57 (2.31)	0.46 (0.33)

TABLE 4. 2D uniform lattice

n	Method	Aggregation	d	$\#levels$	$\#cycles$	C_{op}	γ
256	Aggr	BottomUp(2)	—	4	>20 (>20)	1.86 (3.25)	0.61 (0.51)
256	S&S	BottomUp(2)	*MinDiag*	5	7 (6)	3.31 (8.16)	0.08 (0.05)
256	S&S	BottomUp(2)	0.5	5	7 (6)	3.46 (8.27)	0.08 (0.05)
1024	Aggr	BottomUp(2)	—	6	>20 (>20)	1.95 (3.38)	0.67 (0.48)
1024	S&S	BottomUp(2)	*MinDiag*	7	7 (6)	4.26 (10.29)	0.10 (0.08)
1024	S&S	BottomUp(2)	0.5	7	7 (6)	4.15 (10.24)	0.11 (0.08)
4096	Aggr	BottomUp(2)	—	9	>20 (>20)	2.00 (3.81)	0.88 (0.75)
4096	S&S	BottomUp(2)	*MinDiag*	8	7 (6)	5.64 (14.44)	0.10 (0.05)
4096	S&S	BottomUp(2)	0.5	8	7 (6)	5.25 (14.46)	0.08 (0.05)

TABLE 5. 2D lattice with anisotropic weights

3.2.3. *Tandem queuing network.* The final problem in this set is the tandem queueing network problem appearing in [5]. The stencil is given by

$$H_{TandemQueue} = \frac{1}{\mu + \mu_1 + \mu_2} \begin{pmatrix} & & \mu_1 \\ \mu & 0 & \\ & \mu_2 & \end{pmatrix},$$

where $\mu = 10$, $\mu_1 = 11$ and $\mu_2 = 10$ as in [5].

In this problem, similarly to uniform 2D, we use the bottom up aggregation with $s = 4$, as smaller size leads to relatively high operator complexity. The aggregation is calculated in the first and fourth multigrid cycles only. Here too, we provide results for a geometric 2D aggregation of size 2×2. Table 6 summarizes the results.

3.2.4. *Random walk on unstructured planar graph.* The final test problem is a random walk on an unstructured planar undirected graph. This problem also appears in [5]. The graph is generated by choosing n random points in the unit square, and triangulating them using Delaunay triangulaion. The weight of an edge (i, j) is determined by the reciprocal of the outgoing degree of node i, that is, we normalize the columns of the symmetric binary matrix representing the graph to make it column-stochastic. In this problem, we use the bottom up aggregation with $s = 4$ in S&S. As for

n	Method	Aggregation	d	$\#levels$	$\#cycles$	C_{op}	γ
256	Aggr	BottomUp(2)	—	4	>20 (>20)	1.95 (3.38)	0.67 (0.48)
256	S&S	BottomUp(4)	*MinDiag*	4	19 (14)	1.67 (2.54)	0.47 (0.35)
256	S&S	Geometric	*MinDiag*	3	18 (14)	1.50 (2.11)	0.46 (0.36)
256	S&S	BottomUp(4)	0.5	4	15 (13)	1.67 (2.52)	0.39 (0.34)
256	S&S	Geometric	0.5	3	14 (12)	1.51 (2.12)	0.39 (0.34)
1024	Aggr	BottomUp(2)	—	6	>20 (>20)	2.12 (4.04)	0.8 (0.57)
1024	S&S	BottomUp(4)	*MinDiag*	5	>20 (19)	1.66 (2.62)	0.63 (0.47)
1024	S&S	Geometric	*MinDiag*	4	>20 (17)	1.57 (2.30)	0.61 (0.43)
1024	S&S	BottomUp(4)	0.5	5	21 (15)	1.69 (2.71)	0.52 (0.38)
1024	S&S	Geometric	0.5	4	18 (14)	1.57 (2.31)	0.46 (0.36)
4096	Aggr	BottomUp(2)	—	9	>20 (>20)	2.23 (4.57)	0.89 (0.64)
4096	S&S	BottomUp(4)	*MinDiag*	6	>20 (21)	1.65 (2.62)	0.74 (0.53)
4096	S&S	Geometric	*MinDiag*	5	>20 (18)	1.42 (2.40)	0.69 (0.47)
4096	S&S	BottomUp(4)	0.5	6	>20 (16)	1.67 (2.60)	0.62 (0.41)
4096	S&S	Geometric	0.5	5	18 (14)	1.60 (2.40)	0.47 (0.35)

TABLE 6. Tandem queuing network with $\mu = 10$, $\mu_1 = 11$ and $\mu_2 = 10$

n	Method	Aggregation	d	$\#levels$	$\#cycles$	C_{op}	γ
256	Aggr	BottomUp(3)	—	4	>20 (>20)	1.47 (2.10)	0.76 (0.63)
256	S&S	BottomUp(4)	*MinDiag*	4	15 (12)	1.66 (2.51)	0.46 (0.34)
256	S&S	BottomUp(4)	0.5	4	14 (12)	1.63 (2.31)	0.44 (0.36)
1024	Aggr	BottomUp(3)	—	5	>20 (20)	1.50 (2.24)	0.77 (0.65)
1024	S&S	BottomUp(4)	*MinDiag*	5	19 (13)	1.87 (3.23)	0.54 (0.36)
1024	S&S	BottomUp(4)	0.5	5	16 (12)	1.81 (3.11)	0.48 (0.33)
4096	Aggr	BottomUp(3)	—	7	>20 (>20)	1.52 (2.29)	0.84 (0.73)
4096	S&S	BottomUp(4)	*MinDiag*	6	>20 (13)	2.01 (3.68)	0.67 (0.38)
4096	S&S	BottomUp(4)	0.5	6	16 (12)	2.00 (3.75)	0.48 (0.35)

TABLE 7. Unstructured Planar Graph

Aggr, the best convergence factor is achieved by choosing $s = 3$ (triangles). The aggregation is calculated in the first and fourth cycles only. Table 7 summarizes the results.

While the basic aggregation method requires more than 20 cycles to converge in all problems, it is clear that the S&S method is far better, especially with F-cycles, which seem to yield mesh-independent convergence throughout. In all the 1D problems, the S&S method is very efficient, and the 2D performance is good, though occasional difficulties are encountered.

The *bottom-up* approach seems very promising, though substantial research is yet required. Clearly, the use of a constant s is not a good general choice, and the tradeoff of small s (good convergence, high complexity) versus large s needs to be explored.

As for the stretching parameter, the results indicate that using the minimal diagonal element, thus retaining positive matrices, is not scalable in general. Using $d = 0.5$ yields good properties in these tests. This aspect requires further investigation. Note that making this parameter too high might make the spectral radius of some coarse grid operators larger than one, which may compromise convergence (as squaring the operator will generate a positive eigenvalue bigger than 1.) An adaptive choice for this parameter may prove to be best.

4. Conclusions and Future Work

We introduce a novel Square and Stretch multigrid method for computing the principal eigenvector of column stochastic matrices. The S&S method exhibits far better convergence properties than the basic aggregation algorithm and is only slightly more complicated. The new method involves squaring the fine grid operator and then stretching its spectrum as part of the coarse operator construction. Numerical tests show that the new method performs very well in 1D problems and is also quite effective in 2D when bigger aggregates are collected.

The novel *bottom-up* aggregation method has introduces new ideas, starting with that of matching the nodes to aggregates in an order opposite to their strength or dominance. Also, classifying the candidate aggregates by their total weights and edges without giving extra meaning to the seed seems effective and deserves further study. Currently, this procedure may be of relatively high complexity (though still linear) when searching big aggregates, but it is not necessary to calculate the aggregations in every cycle—twice during the entire set of cycles suffice.

Lastly, the stretching parameter d is of high importance, and by choosing it adaptively in some way at each level we might improve the method significantly.

Overall, although further research is clearly necessary, the performance is already quite promising, and it seems competitive with respect to the smoothed aggregation approach presented in [5], though a reliable comparison requires optimizing both approaches.

It is likely that the S&S method can be extended to other types of matrices quite easily. A hint of that is indicated in the experiments where the coarse grid operators do not remain column stochastic and yet their coarse grid problems are still solved with the same efficiency. Future application to the solution of sparse linear systems is also envisaged.

5. Acknowledgements

We thank Achi Brandt, Panayot Vassilevski, Yair Koren, Hans de Sterck, Steve McCormick and Tom Manteuffel for their valuable advice and suggestions.

References

[1] A. Borzi and G. Borzi, *Algebraic multigrid methods for solving generalized eigenvalue problems*, Int. J. Numer. Meth. Eng., 65 (2006), pp. 1186–1196.

[2] A. Brandt, S. McCormick, and J. Ruge, *Multigrid methods for differential eigenproblems*, SIAM. J. Stat. Sci. Comput., 4 (1983), pp. 655–684.

[3] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization and VLSICAD, Kluwer (Boston), 2003, pp. 1–69.

[4] H. De Sterck, T. A. Manteuffel, S. F. McCormick, Q. Nguyen, and J. Ruge, *Multilevel adaptive aggregation for Markov chains, with application to web ranking*, Submitted to SIAM J. Sci. Comput., (2007).

[5] H. De Sterck, T. A. Manteuffel, S. F. McCormick, J. Pearson, and J. Ruge, *Smoothed aggregation multigrid for markov chains*, Submitted to SIAM J. Sci. Comput., (2008).

[6] G. Horton and S. T. Leutenegger, *A multi-level solution algorithm for steady-state Markov chains*, Perform. Eval. Rev., 22 (1994), pp. 191–200.

[7] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub, *Extrapolation methods for accelerating pagerank computations*, Proc. 12th Int'l World Wide Web Conf., (2003).

[8] O. Livne and A. Brandt, *O(N log N) multilevel calculation of n eigenfunctions*, in Multiscale computational methods in chemistry and physics, vol. 177 of NATO Science Series: Computer and System Sciences, IOS Press, Amsterdam, 2001.

[9] S. McCormick, *Multilevel adaptive methods for elliptic eigenproblems: a two-level convergence theory*, SIAM J. Numer. Anal., 31 (1994), pp. 1731–1745.

[10] E. Virnik, *An algebraic multigrid preconditioner for a class of singular M-matrices*, SIAM J. Sci. Comput., 29 (2007), pp. 1982–1991.

[11] I. Yavneh, *Coarse-grid correction for nonelliptic and singular perturbation problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1682–1699.