Square & Stretch Multigrid for Stochastic Matrix Eigenproblems**

Eran Treister and Irad Yavneh

Department of Computer Science, Technion—Israel Institute of Technology.

SUMMARY

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, whereby the prolongation is adapted to yield a better and better coarse representation of the sought eigenvector. A special feature of the present approach is 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 second special feature is a novel bottom-up procedure for defining coarse-grid aggregates. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: Algebraic Multigrid, Smoothed Aggregation, Markov Chains,

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 [1, 2, 3]. Recently, a solver for more general types of M-matrices was developed, based on classical algebraic multigrid techniques [4].

Brandt and Ron [5] 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

Contract/grant sponsor: Israeli Science Foundation; contract/grant number: 795/08

*Correspondence to: eran@cs.technion.ac.il, irad@cs.technion.ac.il

**This is the pre-peer reviewed version of the article, which has been published in final form at Journal of Linear Algebra with Applications.

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 applications (via Google's PageRank algorithm) and in Markov chain processes. Several papers have addressed this problem using multigrid approaches, in particular the classical [6], and the recent [7, 8], which are based on Algebraic Multigrid. The aggregation-based multigrid approaches of [9, 10] are the closest approach to the one presented here. For recent papers focusing on this problem using other types of numerical methods, see [11, 12].

1.1. The Stochastic Matrix Eigenproblem

Let $B \in R^{n\times n}$ 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 $B\mathbf{x} = \mathbf{x}$. Furthermore, $\rho(B) = 1$, where ρ denotes the spectral radius. Our objective is to compute the principal eigenvector of B, 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.

2. The Basic Algorithm

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 $RP = I_c$, 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*: $\mathbf{x}^{k+1} \leftarrow \mathit{Relax}(\mathbf{x}^{k+1}, B, \nu_2)$

2.3. Relaxation

Let D denote the diagonal matrix comprised of the diagonal part 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 solver 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 its real part. (The reason why it is only the real part that is important here is the damping.) 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 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:

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

Evidently, $RP = X_c^k (X_c^k)^{-1} = I_c$, 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 off-diagonal 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, i.e., their sum.

2.5. Aggregation—a bottom-up approach

The aggregation procedure should aim to approximately optimize the tradeoff between convergence rate and operator complexity. The latter is defined as the total number of non-zero elements divided by the number of nonzero elements of the fine-level operator. In general, bigger aggregates (aggressive coarsening) lead to slower convergence, but also to smaller operator complexity. However, experiments show, as expected, that the convergence rates are determined by the size of the biggest aggregates, even if there are only a few such big ones. Therefore, it makes sense to strive to select homogeneously-sized aggregates, and this motivates our approach.

The bottom-up aggregation approach we next describe is an alternative to the (more typical) top-down approach of [9, 10]. Both are motivated by the same notion of strength of connection. We refer as "popular" to variables with many strongly connected neighbors regardless of the directions of the dependencies. In contrast to common practice, our approach is to first make sure that the "less popular" variables are aggregated properly, and only later tend to the "more popular" ones. The reason for this is that "popular" variables will naturally be selected to aggregates anyway, whereas "unpopular" 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, with $X^k = diag(\mathbf{x}^k)$ as above. Here,

$$\hat{S}_{ij} = \begin{cases} B_{ij}\mathbf{x}_j^k & \text{if } i \neq j \text{ and } B_{ij}\mathbf{x}_j^k \geq \theta \max_{l\neq i} B_{il}\mathbf{x}_l^k\,, \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

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 [9, 10].

Our goal is to form aggregates that contain elements which are strongly connected. Since there is no "preferred element" in the aggregate, we do not build our aggregates around a seed. For reasons that are described later, we symmetrize our connectivity matrix as follows:

$$S = \frac{1}{2}\left(\hat{S} + \hat{S}^T\right)\,. \tag{2.6}$$

Our algorithm aims to find the most strongly connected aggregates of size equal to or less than a given integer 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 (2.6) that if $S_{ij} > 0$ then i and j form a circle of size two, so such circles are ubiquitous. We also define the weight of an aggregate as the sum of the elements of the connectivity submatrix corresponding to its members:

$$w(\mathcal{C}_J) = \sum_{i,j \in \mathcal{C}_J} S_{ij}. \tag{2.7}$$

Our second aim is to limit the operator complexity. The tradeoff of low complexity versus better convergence rate 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.

The reason we look for *undirected* circular dependencies is that such dependencies largely result in a transferring of the strength (the value) of the members in a circle within the aggregate as a result of transitions brought about by the relaxation. This property leads to a mutual growth or reduction of the whole aggregate, as required by our algorithm for fast convergence. (Recall that the coarse grid correction shifts the entire aggregate by the same proportion.) For example, when considering a pair of variables as an aggregate, it clearly does not matter which one influences the other.

```
Algorithm: Bottom-up($s$)
repeat
    Among the unassigned elements, find the element $i$ with the least remaining
    neighbors in $S$.
    if $i$ has more than one neighbor then
        Find all circles of length $s$ or less that contain $i$.
        % At least one circle will always be found.
        Choose the circles of maximal length, and amongst those aggregate the circle of
        maximal weight as defined in (2.7).
    else
        Let $p$ be the (only) neighbor of $i$.
        if other neighbors of $p$ with only one neighbor besides $p$ exist then
            Add up to $(s-2)$ of them to $i$'s aggregate.
        end
    end
    Remove the chosen aggregate's members from $S$.
    if After removing the chosen aggregate there are members with no neighbors then
        Add them to $i$'s aggregate.
    end
until all elements are assigned to aggregates.
```

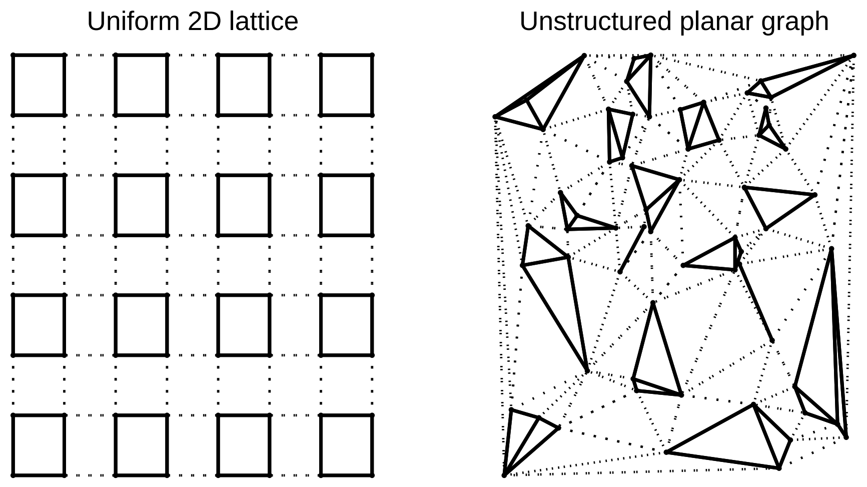

Figure 1. BottomUp algorithm using $s = 4$. Bold/dashed lines refer to connections within/between the aggregates. Left—2D lattice with uniform weights. Right—Unstructured planar graph .

The *Bottom-up*(s) algorithm may be implemented in various ways. We use a Depth-First-Search technique (starting from i) in order to recognize circles of length s, and s buckets in order to sort the members according to the number of neighbors. As this procedure might be a bit complex, we may, following a suggestion in [9, 10], calculate aggregates only once throughout the whole iterative process and then freeze them. In order to do that, we first need to obtain a locally smooth approximation to $\mathbf{x}$ in order to define the matrix S, and that can be achieved by applying a few relaxations to a random guess.

One nice property that this *Bottom-Up* algorithm has, is that it produces structured aggregation for structured problems. Figure (1) shows two examples of aggregations with $s = 4$, where in the case of the structured graph we get the optimal 2×2 aggregation, and in the case of the unstructured graph we find that most of the aggregates are indeed of size 4.

2.6. 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.

2.7. Properties of the Basic EIS Algorithm

The basic algorithm converges to the solution in all problems we tested. However, as observed in [9], 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, and it is explained in section 3. The usual remedy is to use higher-order transfers. Indeed, this approach is adopted in [9], 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 [9] 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 plain aggregation transfer operators) and appears to yield quite competitive performance, though a detailed comparison between the approaches would require first optimizing them on a common platform, and this has not been performed.

3. The Square & Stretch Algorithm Description

Before presenting the new algorithm, we motivate it by explaining why low-order transfer operators such as simple aggregation lead to slow convergence. Denote the eigenpairs of B by $\{\lambda_k, \mathbf{v}_k\}_{k=1}^n$ where[†] $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n$, with $\mathbf{v}_1 = \mathbf{x}$. The coarse-grid problem, $B\mathbf{x}_c = \mathbf{x}_c$ can be written as a minimization problem,

$$\mathbf{x}_c = \text{argmin}_{\mathbf{x}'_c} ||B_c\mathbf{x}'_c - \mathbf{x}'_c|| = \text{argmin}_{\mathbf{x}'_c} ||R(B-I)P\mathbf{x}'_c||, \tag{3.1}$$

and since the coarse operator is column stochastic, this minimization functional has a unique solution with a value of 0. Suppose next that we can write the prolongated coarse-grid vector as a linear combination of the eigenvectors of B:

$$P\mathbf{x}'_c = \sum_{k=1}^{n} \alpha_k \mathbf{v}_k \, .$$

Then the argument in the minimization problem becomes

$$\sum_{k=2}^{n} \alpha_k(\lambda_k - 1)R\mathbf{v}_k \, .$$

Recall now that the purpose of the coarse-grid correction is to reduce efficiently smooth eigenvectors, v_k, with $\lambda_k \approx 1$, while the relaxation reduces the rough eigenvectors, with negative or small positive eigenvalues. We see here that the factor $(\lambda_k - 1)$, which is small for smooth eigenvectors, will cause these eigenvectors to be featured far less prominently in the functional than the rough eigenvectors. This can be compensated by high-order R and P, which strongly reduce the generation of rough eigenvectors, and this is precisely what smoothed

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

aggregation achieves. Here we offer an alternative strategy that allows us to continue using the low-order aggregation transfers.

To summarize, the slow convergence brought about by low-order transfers is due to the generation and relative amplification of rough eigenmodes by P (and lack of sufficient damping of such modes by R). These rough eigenmodes generate large residuals in Equation (3.1), and their coupling with the smooth eigenmodes leads to relatively poor coarse-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. Thus, the "roughest" eigenmodes of B (with eigenvalues close to -1) no longer generate large residuals in Equation (3.1). Indeed, they are smooth eigenmodes 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 simple P and R aggregation-based 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 new 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 with the matrix $\hat{B}_c$ defined as follows:

$$\hat{B}_c = R\hat{B}P\,, \quad \text{with } \hat{B} = \left(\frac{1}{1-d}\right)B^2 - \left(\frac{d}{1-d}\right)I\,, \tag{3.2}$$

where $d > 0$ 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.

3.1. 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. In practice, we found that in real spectrum problems, we can safely use $d = 0.5$, which stretches the segment $[0, 1]$ (of B^2) to $[-1, 1]$. Depending on the problem and the aggregates, in some cases of complex spectrum, the multiplications of R and P tend to eliminate the imaginary parts of the eigenvalues and hence we can also use $d = 0.5$. However, in general, we need to use a lower value, otherwise convergence may be compromised.

A safe alternative is given by $d = \min_J\{(RB^2P)_{J,J}\}$. Evidently, the new coarse grid matrix $\hat{B}_c$ then remains irreducible and column stochastic. In some cases, 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.

Another related option we will use in practice is the average of the diagonal entries, and that is in order to prevent excessively small values of d (which lead to slow convergence) because of only a few small diagonal members.

3.2. Limiting the operator complexity—a lumping approach

As indicated earlier, our method of aggregation aims at keeping the operator complexity from growing. However, the dependency matrix S defined in Equation (2.6) is defined only by strong connections. Therefore, if B contains many weak connections, they are not taken into consideration by the aggregation process. Since we square the matrix B, those weak connections multiply and increase the operator complexity with far weaker connections: if $B = B_1 + \epsilon B_2$, then $B^2 = B_1^2 + \epsilon(B_1B_2 + B_2B_1) + \epsilon^2 B_2^2$. Those ϵ^2 terms are significantly smaller than both the strong connections and the weak connections that become important at coarser grids. In order to prevent such complexity growth, we use a lumping process on the coarse grid operator. First we define:

$$L_i(\epsilon) = \{l : \epsilon^2(\max_{j\neq i} B_{ij}x_j^k) > B_{il}x_l^k\},$$

then we redefine the coarse grid operator:

$$(\hat{B}_c)_{ij} = \begin{cases} (B_c)_{ij} & i \neq j \text{ and } j \notin L_i(\epsilon)\,, \\ (B_c)_{ii} + \frac{1}{x_i^k}\left(\sum_{j\in L_i(\epsilon)} (B_c)_{ij}x_j^k\right) & i = j\,, \\ 0 & j \in L_i(\epsilon). \end{cases} \tag{3.3}$$

It should be noted that, although this definition may look simple, carrying it out in each cycle has a computational cost of a few relaxations, even if no lumping is actually carried out and B_c remains the same. Therefore, we avoid using it, unless we are dealing with a significant operator complexity growth (in extremely anisotropic problems).

3.3. Square & Stretch V-cycle 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. Using *BottomUp* aggregation, define the n by n_c prolongation matrix, P, and the n_c by n restriction matrix, R, such that $RP = I_c$ and $PR\mathbf{x}^k = \mathbf{x}^k$.
3. Compute $B_c = RB^2P$.
4. Lump B_c as in (3.3) if necessary.
5. Stretch B_c as in (3.2) by a parameter d.
6. Solve Equation (2.2) recursively, obtaining $\mathbf{x}_c$.
7. Compute $\mathbf{x}^{k+1} = P\mathbf{x}_c$.
8. Apply ν_2 *post-relaxations*: $\mathbf{x}^{k+1} \leftarrow Relax(\mathbf{x}^{k+1}, B, \nu_2)$

4. 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 [9, 7]. 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 $\omega = 0.98$. We begin with 20 relaxation sweeps in order to obtain a relevant initial (smooth) 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 directly. For the S&S method, d represents the stretching parameter of (3.2), where *AvgDiag* refers to choosing the average of the diagonal elements of RB^2P; the second option we test is a constant value, which is dependent on the problem. In real spectrum problems we use $d = 0.5$. This constant is motivated by (3.2), 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. The effective convergence is defined as 'γ_{eff}' $= \gamma^{1/C_{op}}$ and is a measure which compares the effectiveness of the V-cycle and the F-cycle. All structured problems are described by the 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. In all the test problems, the aggregates are calculated only in the first multigrid cycle, and then frozen for the rest of the cycles. The aggregate size parameter s was chosen as the smallest value that keeps the operator complexity bounded (for S&S), and that was generally found to be the best choice. As for Aggr, we choose the value of s that maximizes the effective convergence γ_{eff}.

4.1. 1D uniform chain

The first problem is the simplest 1D uniform Markov chain, with. operator stencil is given by

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

Each of the two boundary nodes has only one outgoing edge with a weight of 1. Here, it is sufficient to use aggregates size of $s = 2$, and since the spectrum is real, we can use the aggressive stretch of $d = 0.5$. Table I summarizes the results. Here, using the safe stretch (*AvgDiag*) and the aggressive stretch produced similar results, because, due to the nature of this problem, they are similar (the diagonal entries are approximately 0.5). Evidently, the S&S method yields superior results for this problem.

4.2. Uniform 2D lattice

The next problem is a two dimensional lattice with uniform weights and stencil given by

$$H_{\text{Uniform2D}} = \frac{1}{4}\begin{pmatrix} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{pmatrix}.$$

In this problem, when using the S&S method, we select $s = 4$, as a smaller aggregation size leads to relatively high operator complexity. Here, the BottomUp algorithm produces 2×2 aggregates (if indeed the mesh size is even and $s = 4$). Since this problem has a real spectrum, we use a constant $d = 0.5$. Table II summarizes these results. Here, the S&S performance is moderate (compared to the 1D case), however, it is still better than that of Aggr.

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
4096	Aggr	2	—	9	>20(>20)	1.99(3.95)	0.86(0.78)	0.93(0.94)
4096	S&S	2	0.50	9	7(6)	1.82(3.53)	0.08(0.04)	0.24(0.40)
4096	S&S	2	*AvgDiag*	9	7(6)	1.99(3.95)	0.08(0.04)	0.27(0.44)
16384	Aggr	2	—	11	>20(>20)	2.00(3.98)	0.86(0.79)	0.93(0.94)
16384	S&S	2	0.50	11	7(6)	1.82(3.57)	0.08(0.04)	0.24(0.40)
16384	S&S	2	*AvgDiag*	11	7(6)	2.00(3.98)	0.08(0.04)	0.28(0.45)
65536	Aggr	2	—	13	>20(>20)	2.00(4.00)	0.86(0.79)	0.93(0.94)
65536	S&S	2	0.50	13	7(6)	1.82(3.58)	0.08(0.04)	0.24(0.41)
65536	S&S	2	*AvgDiag*	13	7(6)	2.00(4.00)	0.08(0.04)	0.28(0.45)
262144	Aggr	2	—	15	>20(>20)	2.00(4.00)	0.86(0.79)	0.93(0.94)
262144	S&S	2	0.50	15	7(6)	1.81(3.57)	0.08(0.04)	0.24(0.40)
262144	S&S	2	*AvgDiag*	15	7(6)	2.00(4.00)	0.08(0.04)	0.28(0.45)

Table I. 1D uniform chain

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
4096	Aggr	4	—	5	>20(>20)	1.32(1.75)	0.83(0.72)	0.87(0.83)
4096	S&S	4	0.50	5	16(12)	1.57(2.31)	0.48(0.33)	0.62(0.62)
4096	S&S	4	*AvgDiag*	5	>20(13)	1.57(2.31)	0.64(0.38)	0.75(0.65)
16384	Aggr	4	—	6	>20(>20)	1.33(1.76)	0.84(0.73)	0.87(0.84)
16384	S&S	4	0.50	7	18(12)	1.65(2.60)	0.56(0.34)	0.71(0.66)
16384	S&S	4	*AvgDiag*	7	>20(14)	1.67(2.68)	0.75(0.47)	0.84(0.75)
65536	Aggr	4	—	7	>20(>20)	1.33(1.77)	0.85(0.75)	0.88(0.85)
65536	S&S	4	0.50	8	18(12)	1.68(2.69)	0.60(0.34)	0.74(0.67)
65536	S&S	4	*AvgDiag*	8	>20(14)	1.69(2.75)	0.73(0.44)	0.83(0.74)
262144	Aggr	4	—	8	>20(>20)	1.33(1.77)	0.84(0.75)	0.88(0.85)
262144	S&S	4	0.50	9	18(12)	1.70(2.78)	0.57(0.34)	0.72(0.68)
262144	S&S	4	*AvgDiag*	9	>20(13)	1.69(2.76)	0.73(0.41)	0.83(0.73)

Table II. 2D uniform lattice

4.3. Anisotropic 2D lattice

The next 2D problem is the 2D lattice with anisotropic weights:

$$H_{\text{Anisotropic2D}} = \frac{1}{2+2\varepsilon}\begin{pmatrix} & \varepsilon & \\ 1 & 0 & 1 \\ & \varepsilon & \end{pmatrix},$$

where $\varepsilon = 10^{-6}$. In this problem, it is known that the aggregation occurs first along the strong connections (horizontal in this case) 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 of length greater than 2, so using a larger s would not lead to any significant difference.) The reason for the high complexity is the numerous weak connections. In order to overcome this, we use the lumping process described in (3.3). Note that the complexity growth does not occur in the simple aggregation, so no lumping is performed there. As in the

n	Method	s	d	*#levels*	*#cycles*	C_{op}	γ	γ_{eff}
4096	Aggr	2	—	9	>20(>20)	1.96(3.76)	0.85(0.73)	0.92(0.92)
4096	S&S	2	0.50	9	7(7)	2.82(6.48)	0.07(0.04)	0.39(0.61)
4096	S&S	2	*AvgDiag*	9	8(7)	2.82(6.50)	0.09(0.04)	0.43(0.61)
16384	Aggr	2	—	11	>20(>20)	1.98(3.86)	0.86(0.77)	0.93(0.93)
16384	S&S	2	0.50	11	8(7)	2.91(6.93)	0.07(0.04)	0.41(0.63)
16384	S&S	2	*AvgDiag*	11	7(7)	2.91(6.98)	0.07(0.04)	0.41(0.63)
65536	Aggr	2	—	13	>20(>20)	1.99(3.92)	0.86(0.78)	0.93(0.94)
65536	S&S	2	0.50	13	8(7)	2.96(7.22)	0.08(0.04)	0.42(0.64)
65536	S&S	2	*AvgDiag*	13	7(7)	2.96(7.24)	0.07(0.04)	0.41(0.64)
262144	Aggr	2	—	15	>20(>20)	1.99(3.95)	0.86(0.78)	0.93(0.94)
262144	S&S	2	0.50	15	7(6)	2.99(7.45)	0.08(0.04)	0.42(0.65)
262144	S&S	2	*AvgDiag*	15	7(6)	2.99(7.45)	0.08(0.04)	0.42(0.65)

Table III. 2D lattice with anisotropic weights, with lumping

1D problem, the S&S convergence rates are excellent, and the operator complexity is bounded. Table III summarizes these results.

4.4. Tandem queuing network

The next problem in this set is the tandem queueing network problem appearing in [9, 7]. The stencil is given by

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

where $\mu = 10$, $\mu_1 = 11$ and $\mu_2 = 10$ as in [9, 7].

In this problem, similarly to uniform 2D, we use the bottom up aggregation with $s = 4$ (which tries to form 2×2 aggregates). Although this problem has complex eigenvalues, the influence of R and P allows us to use stretching of $d = 0.5$. Table IV summarizes the results. In its best option, the S&S method produces similar results to the 2D uniform lattice (Table II) and again is better than Aggr.

4.5. Random walk on unstructured planar graph

The next test problem is a random walk on an unstructured planar undirected graph. This problem also appears in [9, 7]. The graph is generated by choosing n random points in a 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 too, we use the bottom up aggregation with $s = 4$ in S&S, as a smaller size leads to relatively high operator complexity. This problem has a real spectrum, so we can use $d = 0.5$. Table V summarizes the results. The results are again similar to those of the 2D uniform lattice problem.

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
4096	Aggr	4	—	5	>20(>20)	1.48(2.11)	0.90(0.82)	0.93(0.91)
4096	S&S	4	0.50	5	20(15)	1.60(2.41)	0.51(0.36)	0.66(0.65)
4096	S&S	4	*AvgDiag*	5	>20(19)	1.60(2.42)	0.70(0.46)	0.80(0.73)
16384	Aggr	4	—	6	>20(>20)	1.49(2.14)	0.90(0.87)	0.93(0.94)
16384	S&S	4	0.50	6	>20(14)	1.61(2.45)	0.55(0.34)	0.69(0.64)
16384	S&S	4	*AvgDiag*	7	>20(>20)	1.61(2.46)	0.78(0.49)	0.86(0.75)
65536	Aggr	4	—	7	>20(>20)	1.50(2.15)	0.92(0.89)	0.95(0.95)
65536	S&S	4	0.50	8	>20(14)	1.62(2.48)	0.60(0.33)	0.73(0.64)
65536	S&S	4	*AvgDiag*	8	>20(>20)	1.62(2.48)	0.84(0.55)	0.90(0.79)
262144	Aggr	4	—	8	>20(>20)	1.50(2.16)	0.93(0.89)	0.95(0.95)
262144	S&S	4	0.50	9	>20(14)	1.62(2.50)	0.65(0.33)	0.77(0.64)
262144	S&S	4	*AvgDiag*	9	>20(>20)	1.62(2.49)	0.89(0.66)	0.93(0.85)

Table IV. Tandem queuing network with $\mu = 10$, $\mu_1 = 11$ and $\mu_2 = 10$

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
4096	Aggr	4	—	6	>20(>20)	1.37(1.86)	0.85(0.74)	0.89(0.85)
4096	S&S	4	0.50	6	17(12)	1.81(3.05)	0.51(0.33)	0.69(0.70)
4096	S&S	4	*AvgDiag*	6	>20(12)	1.82(3.08)	0.68(0.35)	0.81(0.71)
16384	Aggr	4	—	7	>20(>20)	1.37(1.87)	0.84(0.78)	0.88(0.87)
16384	S&S	4	0.50	7	17(12)	1.87(3.31)	0.54(0.32)	0.72(0.71)
16384	S&S	4	*AvgDiag*	7	>20(13)	1.86(3.29)	0.73(0.41)	0.84(0.76)
65536	Aggr	4	—	8	>20(>20)	1.37(1.87)	0.86(0.78)	0.89(0.87)
65536	S&S	4	0.50	8	17(12)	1.88(3.42)	0.54(0.33)	0.72(0.72)
65536	S&S	4	*AvgDiag*	8	>20(13)	1.87(3.37)	0.73(0.41)	0.84(0.77)
262144	Aggr	4	—	9	>20(>20)	1.37(1.88)	0.86(0.79)	0.89(0.88)
262144	S&S	4	0.50	9	17(12)	1.90(3.53)	0.55(0.33)	0.73(0.73)
262144	S&S	4	*AvgDiag*	9	>20(13)	1.89(3.48)	0.73(0.41)	0.85(0.77)

Table V. Unstructured Planar Graph

4.6. ATM queueing network

The next test problem is the multi-class, finite buffer, priority system that, as in [13], has been applied to model an ATM queueing networks. This test was also considered in [7]. A complete description and the code used to build the transition matrices are provided in [14]. Here we used buffer sizes of 16, 32, 64 and 128 in order to generate the matrices. Since this problem has a complex spectrum, the algorithm does not converge when using $d = 0.5$ (overly aggressive stretching). A slightly less aggressive choice of $d = 0.45$ does yield good results, however, as the problem grows the complex values in its spectrum also grow so we need less aggressive stretching. In this problem, the simple aggregation surprisingly yields nice results (with F cycles), which are somewhat better than the results of S&S due to the low cost of the method. Table VI summarizes the results.

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
1940	Aggr	4	—	5	>20(16)	1.64(2.54)	0.55(0.31)	0.70(0.63)
1940	S&S	4	0.45	5	13(12)	2.73(5.34)	0.24(0.23)	0.60(0.76)
1940	S&S	4	*AvgDiag*	5	14(12)	2.69(5.20)	0.28(0.22)	0.62(0.75)
7956	Aggr	4	—	7	>20(18)	1.61(2.48)	0.70(0.45)	0.80(0.73)
7956	S&S	4	0.45	6	12(12)	2.81(5.82)	0.22(0.23)	0.59(0.78)
7956	S&S	4	*AvgDiag*	6	14(12)	2.89(5.97)	0.31(0.23)	0.67(0.78)
32276	Aggr	4	—	8	>20(17)	1.58(2.41)	0.72(0.32)	0.81(0.62)
32276	S&S	4	0.45	7	12(12)	2.93(6.36)	0.23(0.23)	0.61(0.79)
32276	S&S	4	*AvgDiag*	7	16(12)	2.78(5.87)	0.30(0.24)	0.65(0.78)
130068	Aggr	4	—	9	>20(19)	1.56(2.35)	0.74(0.49)	0.83(0.74)
130068	S&S	4	0.35	8	13(12)	2.97(6.61)	0.26(0.23)	0.64(0.80)
130068	S&S	4	*AvgDiag*	8	14(12)	2.91(6.34)	0.27(0.23)	0.64(0.79)

Table VI. ATM queueing network

4.7. Octagonal Mesh

The next test problem is an octagonal mesh, which also appears in [7]. We use the same directed graph $\mathcal{G}$, shown on Figure (2), and the same notation of probabilities: μ_+ indicates the probability of moving forward, μ_- is the probability of moving backward, and μ_0 is the probability of staying at the current node (self loop). However, here we used a stretched version of the matrix, that is, we set $\mu_0 = 0$ and stretched the other two probabilities (as in [7]), the same way as we do in Equation (3.2). This results in defining:

$$B = \mu_+ G diag(1^T G)^{-1} + \mu_- G^T diag(1^T G^T)^{-1} ,$$

where $\mu_+ = 0.94$ and $\mu_- = 0.06$. Here, since the spectrum has large imaginary parts (see Figure (2)), we need to use low values of d for the method to converge. Because of that, in big problems, the convergence with the choice of $d = AvgDiag$ becomes similar to the one when using a constant stretch. Here we also tried using aggregates of size 8 due to the nature of this problem, and this shows that, when using bigger aggregates, the spectrum of the coarse grid problem might become less complex, allowing us to use $d = 0.5$. However, using bigger aggregates deteriorates the convergence rates, although the operator complexity is much better. Table VII summarizes these results.

4.8. Triangular Lattice

The last test problem is new, taken from [15]. Consider a random walk on an $(m+1) \times (m+1)$ triangular grid, illustrated in Figure (3) for $m = 6$. The points of the grid are labeled $(j, i), (i = 0, ..., m; j = 0, ..., m - i)$. From the point (j, i), a transition may take place to one of the four adjacent points $(j \pm 1, i \pm 1)$. The probability of jumping to either of the nodes $(j-1, i)$ or $(j, i-1)$ is $(\frac{j+i}{m})$, with the probability being split equally between the two nodes when both nodes are on the grid. The probability of jumping to either of the nodes $(j+1, i)$ or $(j, i+1)$ is $(1-\frac{j+i}{m})$, with the probability again being split when both nodes are on the grid. The spectrum of this matrix is essentially real for small sizes (imaginary values of a machine precision order), however, as the number of unknowns grows, a few complex eigenvalues gradually emerge and

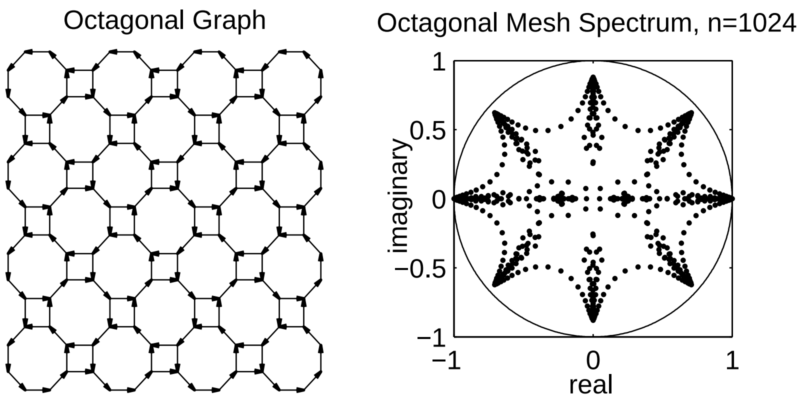

Figure 2. Left: the octagonal graph $\mathcal{G}$. Right: the spectrum of the octagonal mesh matrix B.

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
4232	Aggr	4	—	6	>20(18)	1.74(2.93)	0.85(0.49)	0.91(0.79)
4232	S&S	4	0.35	6	17(9)	2.04(3.68)	0.49(0.18)	0.70(0.62)
4232	S&S	4	*AvgDiag*	6	18(9)	2.07(3.74)	0.57(0.19)	0.76(0.65)
16200	Aggr	4	—	7	>20(>20)	1.72(2.88)	0.81(0.71)	0.88(0.89)
16200	S&S	4	0.35	7	>20(9)	2.00(3.58)	0.65(0.20)	0.81(0.63)
16200	S&S	4	*AvgDiag*	7	>20(10)	2.07(3.75)	0.70(0.30)	0.84(0.72)
66248	Aggr	4	—	8	>20(>20)	1.85(3.22)	0.83(0.82)	0.91(0.94)
66248	S&S	4	0.30	8	>20(10)	2.05(3.76)	0.71(0.27)	0.84(0.71)
66248	S&S	4	*AvgDiag*	8	>20(11)	2.07(3.77)	0.75(0.35)	0.87(0.75)
262088	Aggr	4	—	9	>20(>20)	1.79(3.05)	0.86(0.79)	0.92(0.93)
262088	S&S	4	0.20	9	>20(14)	2.01(3.87)	0.76(0.47)	0.87(0.82)
262088	S&S	4	*AvgDiag*	9	>20(15)	2.01(3.86)	0.76(0.48)	0.87(0.83)
4232	S&S	8	0.50	4	>20(>20)	1.19(1.43)	0.78(0.68)	0.81(0.76)
16200	S&S	8	0.50	5	>20(>20)	1.20(1.46)	0.78(0.67)	0.81(0.76)
66248	S&S	8	0.50	6	>20(>20)	1.21(1.48)	0.80(0.67)	0.83(0.76)
262088	S&S	8	0.50	7	>20(>20)	1.30(1.71)	0.83(0.83)	0.87(0.90)

Table VII. Octagonal Mesh

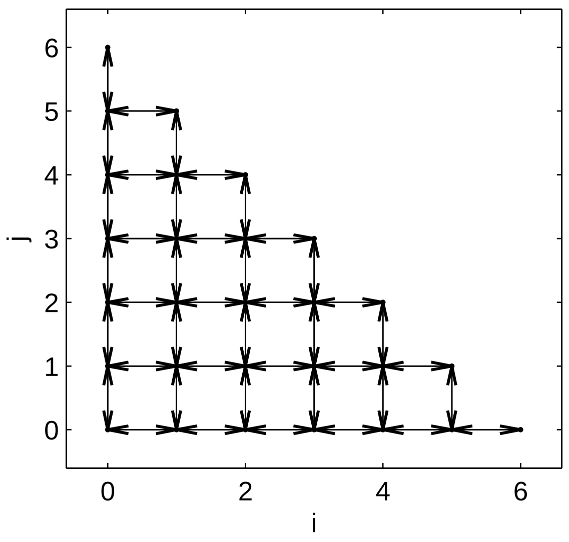

Figure 3. Triangular lattice of 28 states

n	Method	s	d	$\#levels$	$\#cycles$	C_{op}	γ	γ_{eff}
4186	Aggr	4	—	6	>20(>20)	1.49(2.17)	0.84(0.83)	0.89(0.92)
4186	S&S	4	*AvgDiag*	6	>20(>20)	1.93(3.35)	0.83(0.60)	0.91(0.86)
4186	S&S	4	0.50	6	>20(14)	1.94(3.42)	0.58(0.32)	0.75(0.72)
16471	Aggr	4	—	7	>20(>20)	1.49(2.18)	0.82(0.87)	0.87(0.94)
16471	S&S	4	*AvgDiag*	7	>20(>20)	1.97(3.52)	0.85(0.69)	0.92(0.90)
16471	S&S	4	0.50	7	>20(16)	1.98(3.56)	0.63(0.39)	0.79(0.77)
65703	Aggr	4	—	8	>20(>20)	1.49(2.17)	0.83(0.81)	0.88(0.91)
65703	S&S	4	*AvgDiag*	8	>20(>20)	1.99(3.58)	0.78(0.75)	0.88(0.92)
65703	S&S	4	0.48	8	>20(19)	2.00(3.64)	0.66(0.41)	0.82(0.79)
262450	Aggr	4	—	9	>20(>20)	1.51(2.22)	0.88(0.77)	0.92(0.89)
262450	S&S	4	*AvgDiag*	9	>20(>20)	1.99(3.59)	0.76(0.81)	0.87(0.94)
262450	S&S	4	0.40	9	>20(18)	2.02(3.69)	0.81(0.50)	0.90(0.83)

Table VIII. Triangular Mesh

become significant. For this reason, our method is not scalable with a constant value of d, and we need to use smaller values as the problem grows. Here, the best choice in terms of effective convergence is to use the S&S with a constant stretch with F-cycles. Table VIII summarizes the results.

While the basic aggregation method requires more than 20 cycles to converge in almost all problems, it is clear that the S&S method is better, especially with F-cycles, which seem to yield mesh-independent convergence in most cases. The S&S method is efficient, and the performance is mostly good, though occasional difficulties are encountered, mainly in complex spectrum problems.

To summarize this section, the *bottom-up* approach seems very promising, though substantial research is yet required. The use of a constant s is not optimal in general, 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 average of the diagonal elements is not scalable in general, however, it gives us a good starting point since it always leads to convergence. Note that choosing this parameter excessively 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.) Using the constant value 0.5 yields good properties in most of the tests, and is safe to use on real spectrum problems. In complex spectrum tests, other constant values suffice but might need to be reduced as the problem grows. This aspect requires further investigation as an adaptive choice for this parameter may prove to be best.

5. Conclusions and Future Work

In this paper, we introduce a novel Square and Stretch multigrid method for computing the principal eigenvector of column stochastic matrices. The S&S method exhibits better convergence properties than the basic aggregation algorithm and is only mildly 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 generally performs well in one and two dimensions. We expect that higher dimensions may introduce some challenges.

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

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. Calculating this value up to a two digits precision is an option, but it will increase the cost of the algorithm, and therefore has not been tried here.

Overall, although further research is clearly necessary, the performance is quite promising, and it seems competitive with respect to the smoothed aggregation and AMG approaches presented in [9, 7], 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, where hopefully we will be able to extend the scope of matrices that existing methods can handle.

6. 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. Brandt A, McCormick S, Ruge J. Multigrid methods for differential eigenproblems. *SIAM. J. Stat. Sci. Comput.* 1983; **4**:655–684.
2. Livne O, Brandt A. O(N log N) multilevel calculation of n eigenfunctions. *Multiscale computational methods in chemistry and physics*, *NATO Science Series: Computer and System Sciences*, vol. 177. IOS Press: Amsterdam, 2001.
3. McCormick S. Multilevel adaptive methods for elliptic eigenproblems: a two-level convergence theory. *SIAM J. Numer. Anal.* 1994; **31**(6):1731–1745.
4. Borzi A, Borzi G. Algebraic multigrid methods for solving generalized eigenvalue problems. *Int. J. Numer. Meth. Eng.* 2006; **65**(8):1186–1196.
5. Brandt A, Ron D. Multigrid solvers and multilevel optimization strategies. *Multilevel Optimization and VLSICAD*, Kluwer (Boston), 2003; 1–69.
6. Horton G, Leutenegger ST. A multi-level solution algorithm for steady-state Markov chains. *Perform. Eval. Rev.* 1994; **22**:191–200.
7. H De Sterck, Manteuffel TA, McCormick SF, Miller K, Ruge J, Sanders G. Algebraic multigrid for markov chains. *Submitted to SIAM J. Sci. Comput.* 2009; .
8. Virnik E. An algebraic multigrid preconditioner for a class of singular M-matrices. *SIAM J. Sci. Comput.* 2007; **29**(5):1982–1991.
9. H De Sterck, Manteuffel TA, McCormick SF, Miller K, Pearson J, Ruge J, Sanders G. Smoothed aggregation multigrid for markov chains. *Accepted to SIAM J. Sci. Comput.* 2009; .
10. H De Sterck, Manteuffel TA, McCormick SF, Nguyen Q, Ruge J. Multilevel adaptive aggregation for Markov chains, with application to web ranking. *SIAM J. Sci. Comput* 2008; **30**:2235–2262.
11. Kamvar S, Haveliwala T, Manning C, Golub G. Extrapolation methods for accelerating pagerank computations. *Proc. 12th Int'l World Wide Web Conf.* 2003; .
12. Sidi A. Vector extrapolation methods with applications to solution of large systems of equations and to pagerank computations. *Comp. & Maths. with applications* 2008; **56**:1–24.
13. Philippe B, Saad Y, Stewart WJ. Numerical methods in markov chain modeling. *Journal of Operations Research* 1992; **40**(6):1156–1179.
14. Stewart WJ. *MARCA_models* ; Http://www4.ncsu.edu/ billy/MARCA/marca.html.
15. Stewart G. Matrix generator *MVMRWK* ; Http://math.nist.gov/MatrixMarket/data/NEP/ mvmrwk/ mvmrwk.html.