LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-JRNL-231442

A Generalized Eigensolver based on Smoothed Aggregation (GES-SA) for Initializing Smoothed Aggregation Multigrid (SA)

M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, G. Sanders, P. S. Vassilevski

June 4, 2007

Numerical Linear Algebra with Applications

A GENERALIZED EIGENSOLVER BASED ON SMOOTHED AGGREGATION (GES-SA) FOR INITIALIZING SMOOTHED AGGREGATION MULTIGRID (SA) *

MARIAN BREZINA, TOM MANTEUFFEL, STEVE MCCORMICK,
JOHN RUGE, GEOFFREY SANDERS, PANAYOT VASSILEVSKI †

Abstract. Consider the linear system $A\mathbf{x} = \mathbf{b}$, where A is a large, sparse, real, symmetric, and positive definite matrix and $\mathbf{b}$ is a known vector. Solving this system for unknown vector $\mathbf{x}$ using a smoothed aggregation multigrid (SA) algorithm requires a characterization of the algebraically smooth error, meaning error that is poorly attenuated by the algorithm's relaxation process. For relaxation processes that are typically used in practice, algebraically smooth error corresponds to the near-nullspace of A. Therefore, having a good approximation to a minimal eigenvector is useful to characterize the algebraically smooth error when forming a linear SA solver. This paper discusses the details of a generalized eigensolver based on smoothed aggregation (GES-SA) that is designed to produce an approximation to a minimal eigenvector of A. GES-SA might be very useful as a stand-alone eigensolver for applications that desire an approximate minimal eigenvector, but the primary aim here is for GES-SA to produce an initial algebraically smooth component that may be used to either create a black-box SA solver or initiate the adaptive SA (αSA) process.

Key words. generalized eigensolver, smoothed aggregation, multigrid, near-kernel, GES-SA, adaptive, SA,

AMS subject classifications.

1. Introduction.

In the spirit of algebraic multigrid (AMG; [4, 9, 7]), smoothed aggregation multigrid (SA; [10, 12]) has been designed to solve linear systems of equations with little or no prior knowledge regarding the geometry or material coefficients of the problem. Therefore, SA is often well-suited for problems discretized on unstructured meshes, with varying coefficients, or with no underlying geometry at all. Consider solving sparse linear systems that arise from discretizing elliptic PDE problems. The relaxation processes commonly used in multigrid are computationally cheap, but fail to adequately reduce certain types of error, which we call error that is *algebraically smooth* with respect to the given relaxation. If a characterization of algebraically smooth error is known, in the form of a small set of vectors, the SA framework is ideally suited for construction of intergrid transfer operators that eliminate such error components. For example, in a three-dimensional elasticity problem, six such components (the so-called rigid body modes) form a characterization of the algebraically smooth error. These are often available from discretization packages. However, for certain problems, such a characterization may not be readily available, even for scalar problems.

Adaptive SA (αSA [3]) was designed specifically to create a representative set of vectors in cases where a characterization of algebraically smooth error is not known. Initially, simple relaxation is performed on a homogeneous version of the problem for all levels of the multigrid hierarchy being constructed. These coarse-level approximations are used to achieve a global-scale update that serves as our first algebraically smooth vector component. Using this one resulting component, the SA framework

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

†Department of Applied Math, University of Colorado at Boulder (sandersg@colorado.edu, , tmantuef@colorado.edu, stevem@colorado.edu, ruge@colorado.edu) and CASC, Lawrence Livermore National Laboratory (vassilevski1@llnl.gov).

is employed to construct a multigrid solver, and the whole process can be repeated with the updated solver playing the role of relaxation. At each step, the adequacy of the solver is assessed by monitoring convergence factors, and if the current solver is deemed adequate, then the adaptive process is terminated and the current solver is retained.

Adaptive SA has proven successful for a class of problems. However, in certain difficult cases, an impractically large number of relaxation steps is required to compute the first algebraically smooth component. The eigensolver presented in this paper is intended to accurately calculate the initial component for such cases, expanding the class of problems that αSA successfully solves.

As with any multigrid solver, the success of the method depends on designing coarse spaces capable of eliminating algebraically smooth error, complementing the relaxation process. To motivate the use of an eigensolver, we recall the concept of near-kernel error components. Consider the linear system $A\mathbf{x} = \mathbf{b}$, where A is real, symmetric, positive definite, and scaled so diagonal entries are all ones. For a given approximate solution $\tilde{\mathbf{x}}$, define the associated error vector to be $\mathbf{e} := \mathbf{x} - \tilde{\mathbf{x}}$. If the error is such that

$$\|A\mathbf{e}\|_2 << \|A\|_2\|\mathbf{e}\|_2, \tag{1.1}$$

we say that $\mathbf{e}$ is *near-kernel* or *near-nullspace*. Simple relaxation processes that we consider, such as Richardson, damped-Jacobi, and Gauss-Seidel, are based on residual correction. If the associated residual vector, $\mathbf{r} := \mathbf{b} - A\tilde{\mathbf{x}} = A\mathbf{e}$, is small in the sense of (1.1), then the error correction is negligible for such methods. Therefore, we use algebraically smooth and near-kernel synonymously in the context of such simple relaxation schemes. Because A is usually ill-conditioned, for the eigenvector $\mathbf{v}_1$ of A that corresponds to the smallest eigenvalue, λ_1, or *minimal eigenvector*, we have

$$\|A\mathbf{v}_1\|_2 = \lambda_1\|\mathbf{v}_1\|_2 << \|A\|_2\|\mathbf{v}_1\|_2. \tag{1.2}$$

In other words, $\mathbf{v}_1$ is a near-kernel component that is algebraically smooth with respect to a simple relaxation scheme. Ideally, this would be the first vector that an adaptive SA method computes to produce an SA solver with good performance. Computing a minimal eigenvector is the goal of the generalized eigensolver based on smoothed aggregation (GES-SA) that we present in this paper.

It is known from the literature (eg. [1, 2]) that a sufficient condition for optimal two-level convergence is that, for a given interpolation operator, P, and for any $\mathbf{u}$ on the fine-grid, there exists a $\mathbf{v}$ from the coarse-grid, such that

$$\|\mathbf{u} - P\mathbf{v}\|_2^2 \leq \frac{C}{\|A\|_2}(A\mathbf{u}, \mathbf{u}), \tag{1.3}$$

for a constant C. This requirement is known in the literature as *the weak approximation property*, and reflects the observation noted in [7, 8] that the minimal eigenvector needs to be interpolated with accuracy inversely proportional to the size of its eigenvalue. (Note that $\|A\|_2 = \mathcal{O}(1)$ due to the scaling requirement on the diagonal of A.) This emphasizes the need for accurate computation of the minimal eigenvector.

The GES-SA algorithm performs a series of subspace corrections that minimize the Rayleigh quotient over various subspaces. It uses local subspace corrections that are analagous to a nonlinear relaxation process, and global subspace corrections that are analagous to a nonlinear coarse-grid correction. In short, GES-SA is a variant

of algebraic Rayleigh quotient multigrid (RQMG [5]) that uses overlapping block-Rayleigh quotient Gauss-Seidel for its relaxation process and multiplicative smoothed aggregation corrections for its coarse-grid corrections. Although the actual eigenproblem that we wish to solve on the finest grid is not a generalized one, we phrase GES-SA as a generalized eigensolver to accommodate recursive application of the eigensolver on coarse subspaces.

The rest of section 1 gives a simple example and a background on smoothed aggregation multigrid. Section 2 introduces the components of GES-SA. Section 3 presents how the components introduced in section 2 are put together to form the full GES-SA algorithm. Section 4 presents a numerical example with results that demonstrate how the linear SA solvers produced with GES-SA have desirable performance for particular problems. Finally, section 5 makes concluding remarks.

1.1. The Model Problem. EXAMPLE 1. Consider the linear problem $A\mathbf{x} = \mathbf{b}$ and its associated generalized eigenvalue problem $A\mathbf{x} = \lambda B\mathbf{x}$. Up to a scaling, A is the 1D Laplacian with Dirichlet boundaries, discretized with equidistant, linear finite elements or second order central differences:

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \tag{1.4}$$

an $n \times n$ tridiagonal matrix. Matrix B for this example is I_n, the identity operator on $\mathbb{R}^n$. The *full set of nodes* for this problem is $\Omega_n = \{1, 2, ..., n\}$. The problem size, $n = 9$, is used throughout this paper to illustrate various concepts regarding the algorithm. *Note that the 1D problem is used merely to display concepts and is not of further interest, as its tridiagonal structure is treated with optimal computational complexity using a direct solver.* However, the example is useful in the sense that it captures the concepts we present in their simplest form.

1.2. Smoothed Aggregation Multigrid. In this section we briefly recall the smoothed aggregation multigrid (SA) framework for constructing a multigrid hierarchy. Like any algebraic multilevel method, SA requires a setup phase. Here, we follow the SA setup phase presented in [10, 12, 11].

At the heart of SA coarsening is a discrete partitioning of fine-level nodes into a disjoint covering of the full set of nodes. The members of partition are locally grouped based on their strength of coupling within the graph of the problem matrix. Each of these groups is called an *aggregate*. For our purposes, we formally define aggregation as follows:

DEFINITION 1.1. *A sequence of J subsets $\{\mathcal{A}_j\}_{j=1}^J$ is an* **aggregation** *of $\Omega_n = \{1, 2, ..., n\}$ with respect to A, if the following conditions hold.*

- *Covering: $\bigcup_{j=1}^J \mathcal{A}_j = \Omega_n$.*
- *Disjoint: For any $j \neq k$, $\mathcal{A}_j \cap \mathcal{A}_k = \emptyset$.*
- *Connected: For any j, if two nodes $p, q \in \mathcal{A}_j$, then there exists a sequence of edges with endpoints in $\mathcal{A}_j$ that connects p to q within the graph of A.*

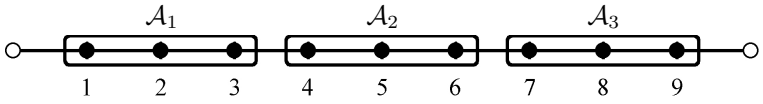
Each individual subset $\mathcal{A}_j$ within the aggregation is called an **aggregate**.

We return to example 1 to explain this concept. An acceptable aggregation of Ω_9 with respect to A would be $J = 3$ aggregates, each of size 3, defined as follows:

$$\mathcal{A}_1 = \{1, 2, 3\}, \quad \mathcal{A}_2 = \{4, 5, 6\}, \quad \mathcal{A}_3 = \{7, 8, 9\}. \tag{1.5}$$

It is easily verified that this partitioning satisfies definition 1.1. This aggregation is pictured in figure 1.2. Two-dimensional examples are present in section 4.

FIG. 1.1. *Graph of A from example 1 with n = 9, aggregated into three aggregates. Each cartouche encloses the group of nodes in the respective aggregate.*



The appropriate choice of aggregates can have a dramatic impact on the performance of the resulting method. Numerous approaches for aggregation have been explored, either using pure graph theory or geometric knowledge.

We find it useful to represent an aggregation $\{\mathcal{A}_j\}_{j=1}^{J}$ with an $n \times m$ sparse, binary *aggregation matrix*, which we denote by $[\mathcal{A}]$. Each column of $[\mathcal{A}]$ represents a single aggregate, with value one in the (i, j)-th entry if point i is contained in aggregate $\mathcal{A}_j$, and value zero otherwise. In our 1D example, with $n = 9$, we represent the aggregation given in (1.5) as

$$[\mathcal{A}] = \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & & \\ & 1 & \\ & 1 & \\ & 1 & \\ & & 1 \\ & & 1 \\ & & 1 \end{bmatrix}. \tag{1.6}$$

Based on the sparsity structure of $[\mathcal{A}]$, the SA setup phase constructs a tentative interpolation operator, $\hat{P}$, with a range that represents a given, small collection of linearly independent vectors, $\mathcal{K}$. This is done by simply restricting the values of each vector in $\mathcal{K}$ to the sparsity pattern specified by $[\mathcal{A}]$. In practice, $\mathcal{K}$ should consist of prototypical near-kernel vectors. Set $\mathcal{K}$ can either be supplied or computed in an adaptive procedure as in αSA [3] or by way of GES-SA, when the set is only one vector. Either way, once $\mathcal{K}$ is known, the setup phase uses it to construct appropriate operator hierarchies that form a linear multigrid solver. In this paper, we concentrate on the case where prototypical set is composed of one vector, specifically $\mathcal{K} = \{\mathbf{k}_1\}$.

Under the above construction, prototypical vectors are ensured to be in $\mathcal{R}(\hat{P})$, the range of the tentative interpolation operator, and are therefore attenuated by a corresponding coarse-grid correction. However, $\mathcal{K}$ is only a small number of near-kernel components. Other vectors in $\mathcal{R}(\hat{P})$ may actually be quite algebraically oscillatory, which can be harmful to the coarsening process because it could lead to an ill-conditioned coarse-grid operator. Also, some algebraically smooth error components may not be well-represented by $\mathcal{R}(\hat{P})$.

To remedy the situation, SA does not use $\hat{P}$ as its interpolation operator directly, but uses instead an operator P obtained by applying a smoothing operator S to $\hat{P}$ ($P = S\hat{P}$). As a result, algebraically oscillatory components are more orthogonal to its range, while algebraically smooth components are better represented by the range of interpolation. A typical choice for the smoothing operator, S, is one step of the error propagation operator of damped-Jacobi relaxation. In this paper, we use Richardson smoothing under the assumption that the system is diagonally scaled so that diagonal elements are one.

For symmetric problems, such as those we consider here, SA produces a coarse grid using a Galerkin restriction operator, $R = P^T$, and variational coarse-grid operator, $A_c = P^T AP$, as is commonly done in AMG methods. This process is repeated recursively on all grids, constructing a multigrid hierarchy.

The GES-SA algorithm attempts to produce an accurate near-kernel vector to be the initial vector in $\mathcal{K}$. If the convergence of the resulting SA solver is slow, then prototypical near-kernel set, $\mathcal{K}$, may need to be augmented using an approach such as αSA. The additional vectors should be algebraically smooth with respect to the current linear solver and not just with respect to relaxation. This is the basic principle behind αSA and is covered in detail in [3].

2. Subspace Correction Methods for Generalized Eigenvalue Problems. Consider the generalized eigenvalue problem $A\mathbf{v} = \lambda B\mathbf{v}$, where A and B are given $n \times n$ real, symmetric, positive definite (SPD) matrices, $\mathbf{v}$ is an unknown eigenvector of length n, and λ is an unknown eigenvalue. Our target problem is stated as follows:

$$\begin{array}{c}\text{find an eigenvector, } \mathbf{v}_1 \neq \mathbf{0}, \text{ corresponding to the smallest} \\ \text{eigenvalue, } \lambda_1, \text{ in the eigenproblem } A\mathbf{v} = \lambda B\mathbf{v}.\end{array} \tag{2.1}$$

For the sake of convenience, $\mathbf{v}_1$ is called a *minimal eigenvector* and the corresponding eigenvalue, λ_1, is called the *minimal eigenvalue*.

First, we review a well-known general strategy for approximating the solution of (2.1), an approach that has been used in [5] and [6], to introduce the specific versions of this method we use. This strategy is to select a subspace of $\mathbb{R}^n$ and choose a vector in the subspace that minimizes the Rayleigh quotient. In GES-SA, we essentially do two types of subspace selection: one uses local groupings to select subspaces with local support; the other uses smoothed aggregation to select low-resolution subspaces with global support. The corrections within the local subspaces are a form of nonlinear relaxation, and those within the global subspaces are a nonlinear coarse-grid correction. These two subspace correction processes are used together in a typical multigrid way.

First, we recall the Rayleigh quotient to introduce a minimization principle that allows us to correct an iterate within a given subspace.

DEFINITION 2.1. *The* **Rayleigh Quotient(RQ)** *of a vector,* $\mathbf{v}$*, with respect to real SPD matrices* A *and* B *is the real value*

$$\rho_{A,B}(\mathbf{v}) \equiv \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T B\mathbf{v}}. \tag{2.2}$$

Because A and B are SPD, the solution we seek minimizes the RQ:

$$\rho_{A,B}(\mathbf{v}_1) = \min_{\mathbf{v}\in\mathbb{R}^n} \rho_{A,B}(\mathbf{v}) = \lambda_1 > 0. \tag{2.3}$$

If two vectors $\mathbf{w}$ and $\mathbf{v}$ are such that $\rho_{A,B}(\mathbf{w}) < \rho_{A,B}(\mathbf{v})$, then we say that $\mathbf{w}$ is a better approximate solution to (2.1) than $\mathbf{v}$. Therefore, we may restate problem (2.1) as a minimization problem:

$$\text{find } \mathbf{v}_1 \neq \mathbf{0} \text{ such that } \rho_{A,B}(\mathbf{v}_1) = \min_{\mathbf{v}\in\mathbb{R}^n} \rho_{A,B}(\mathbf{v}). \tag{2.4}$$

This gives us a minimization principle that we use to construct subspace correction methods. Generally, we take an iterate, $\tilde{\mathbf{v}}$, or current approximate solution, and then perform a subspace correction to give an updated approximate minimal eigenvector, $\tilde{\mathbf{w}}$. The updated iterate should be a better approximate solution to (2.1) in the sense that the subspace correction should lower its RQ from that of $\tilde{\mathbf{v}}$. However, care in constructing the subspace is needed to ensure that the RQ is indeed lowered.

Given $\tilde{\mathbf{v}}$, we construct a subspace, $\mathcal{V} \subset \mathbb{R}^n$, such that $\dim(\mathcal{V}) = m << n$ and $\mathcal{V}$ contains a vector with lower RQ:

$$\min_{\mathbf{v}\in\mathcal{V}} \rho_{A,B}(\mathbf{v}) \leq \rho_{A,B}(\tilde{\mathbf{v}}). \tag{2.5}$$

If we select the corrected approximation, $\tilde{\mathbf{w}}$, to be a vector within $\mathcal{V}$ of minimal RQ, then we have improved our approximate solution. So we must solve a restricted minimization problem within the subspace of dimension m,

$$\text{find } \tilde{\mathbf{w}} \neq \mathbf{0} \text{ such that } \rho_{A,B}(\tilde{\mathbf{w}}) = \min_{\mathbf{v}\in\mathcal{V}} \rho_{A,B}(\mathbf{v}). \tag{2.6}$$

This restricted minimization problem is solved for updated iterate $\tilde{\mathbf{w}}$ by restating the minimization problem within the lower-dimensional vector space, $\mathbb{R}^m$ and then mapping the low-dimensional solution to the corresponding vector in $\mathcal{V}$. To do so, we construct an $n \times m$ matrix, P, whose m column vectors are a basis for $\mathcal{V}$. Matrix P is called the *iterpolation matrix*, and it maps $\mathbb{R}^m$ onto $\mathcal{V}$. Note that, for any $\mathbf{v} \in \mathcal{V}$, there exists a unique $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{v} = P\mathbf{y}$. Moreover, the RQ of $\mathbf{v}$ with respect to A and B and the RQ of $\mathbf{y}$ with respect to restricted versions of A and B are equivalent:

$$\rho_{A,B}(\mathbf{v}) = \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T B\mathbf{v}} = \frac{\mathbf{y}^T P^T AP\mathbf{y}}{\mathbf{y}^T P^T BP\mathbf{y}} = \rho_{P^T AP,\, P^T BP}(\mathbf{y}) = \rho_{A_\mathcal{V},B_\mathcal{V}}(\mathbf{y}), \tag{2.7}$$

for $A_\mathcal{V} := P^T AP$ and $B_\mathcal{V} := P^T BP$. Thus, the solution to restricted minimization problem (2.6) is found by solving a low-dimensional minimization problem:

$$\text{find } \mathbf{y}_1 \neq \mathbf{0} \text{ such that } \rho_{A_\mathcal{V},B_\mathcal{V}}(\mathbf{y}_1) = \min_{\mathbf{y}\in\mathbb{R}^m} \rho_{A_\mathcal{V},B_\mathcal{V}}(\mathbf{y}), \tag{2.8}$$

or, equivalently

$$\begin{gathered}\text{find an eigenvector, } \mathbf{y}_1 \neq \mathbf{0}, \text{ corresponding to the smallest} \\ \text{eigenvalue, } \mu_1, \text{ in the eigenproblem } A_\mathcal{V}\mathbf{y} = \mu B_\mathcal{V}\mathbf{y}.\end{gathered} \tag{2.9}$$

After either approximating the solution to low-dimensional minimization problem (2.8) or solving low-dimensional eigenvalue problem (2.9) for $\mathbf{y}_1$ with a standard solver, the solution to the minimization problem restricted to $\mathcal{V}$ defined in (2.6) is recovered by interpolation: $\tilde{\mathbf{w}} \leftarrow P\mathbf{y}_1$.

Vector $\tilde{\mathbf{w}}$ is our updated iterate, and the whole process is repeated. We first update $\tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{w}}$, use $\tilde{\mathbf{v}}$ to form a new subspace, $\mathcal{V}$, with its respective interpolation

matrix, P, solve a low-dimensional problem for a new $\mathbf{y}_1$, and then interpolate via $\tilde{\mathbf{w}} \leftarrow P\mathbf{y}_1$.

The specific methods we use for constructing interpolation operators, P, are the features of GES-SA and are explained in the following three sections. Before we discuss how subspaces are selected in RQ-relaxation and coarse-grid correction, we first focus on how a reasonable initial approximation is obtained using a version of the subspace correction algorithm in section 2.1. Section 2.2 presents how we phrase global subspaces corrections or nonlinear coarse-grid corrections, and section 2.3 presents how we phrase the local subspace corrections or nonlinear relaxation processes.

2.1. Initial Guess Development via Aggregation. Since the subspace corrections used in this eigensolver are nonlinear iterations is helpful to develop a fairly accurate initial approximation to a minimal eigenvector. The development of initial approximate minimal eigenvectors is very similar to local subspace correction as presented later in section 2.3. The exception is that we form an additive local subspace correction that starts with the zero-vector. To do so, we require that an aggregation, $\{\mathcal{A}_j\}_{j=1}^{J}$, is provided. Each aggregate induces a subspace, $\mathcal{S}_j \subset \mathbb{R}^n$, defined by all vectors $\mathbf{v}$ whose support is contained entirely in $\mathcal{A}_j$. (Subspaces $\mathcal{S}_j$ constitute the $\mathcal{V}$ used in general subspace correction at the beginning of section 2.) For each subspace, we choose $\tilde{\mathbf{w}}_j$ to minimize the RQ over $\mathcal{S}_j$. We then set the initial guess to be the sum over j of these vectors to create a vector that has its RQ minimized over each disjoint aggregate individually. This creates a vector that has a relatively low RQ and is, therefore, close to being a minimal eigenvector.

For each aggregate, we form a local interpolation matrix, P_j, that maps $\mathbb{R}^{m_j}$ onto $\mathcal{S}_j$, where m_j is the number of nodes in the j-th aggregate. This interpolation matrix is given by

$$P_j = \begin{bmatrix} \top & & \top \\ \hat{\mathbf{e}}_{p_1} & \cdots & \hat{\mathbf{e}}_{p_{m_j}} \\ \bot & & \bot \end{bmatrix}, \tag{2.10}$$

where $\hat{\mathbf{e}}_p$ is the p-th canonical basis vector,

$$(\hat{\mathbf{e}}_p)_i = \left\{ \begin{array}{ll} 1 & i = p \\ 0 & i \neq p \end{array} \right\}, \tag{2.11}$$

and $\{p_q\}_{q=1}^{m_j}$ are the m_j nodes in the j-th aggregation. We then restrict the principal matrices to $\mathcal{S}_j$: $A_{\mathcal{V}} \leftarrow P_j^T A P_j$ and $B_{\mathcal{V}} \leftarrow P_j^T B P_j$. A solution, $\mathbf{y}_1 \neq \mathbf{0}$, to generalized eigenvalue problem (2.9) of size m_j is then found using a standard eigensolver. Note that the choice of scaling for $\mathbf{y}_1$ is not an issue, because the subsequent subspace corrections adjust any scaling. Nodes within the j-th aggregate are initialized by interpolation: $\tilde{\mathbf{w}}_j \leftarrow P\mathbf{y}_1$. After $\tilde{\mathbf{w}}_j$ is found for each aggregate, the initial approximation is constructed by summing the local initial guesses:

$$t\tilde{\mathbf{v}} \leftarrow \sum_{j=1}^{J} \tilde{\mathbf{w}}_j. \tag{2.12}$$

We summarize initial guess development in the form of an algorithm. It is a component of the full GES-SA algorithm and is used by algorithm 3 in section 3.

ALGORITHM 1 : Initial Guess Development

- *Function:* $\tilde{\mathbf{v}} \leftarrow IGD(A, B, \{\mathcal{A}_j\}_{j=1}^J)$.
- *Input:* SPD matrices A and B, and aggregation $\{\mathcal{A}_j\}_{j=1}^J$.
- *Output:* initial approximate solution $\tilde{\mathbf{v}}$ to (2.1).

1. For $j = 1, ..., J$, do the following:
   (a) Form P_j based on $\mathcal{A}_j$ as in (2.10).
   (b) Compute $A_{\mathcal{V}} = P_j^T A P_j$ and $B_{\mathcal{V}} = P_j^T B P_j$.
   (c) Find any $\mathbf{y}_1 \neq \mathbf{0}$ by solving (2.9) via a standard eigensolver.
   (d) Interpolate $\tilde{\mathbf{w}}_j \leftarrow P_j \mathbf{y}_1$.
2. Output $\tilde{\mathbf{v}} \leftarrow \sum_{j=1}^J \tilde{\mathbf{w}}_j$.

The difference between this subspace correction and subsequent subspace corrections is output is produced additively instead of multiplicatively, allowing for easier parallelization. To demonstrate the effect of algorithm 1, we return to example 1. The interpolation matrices are

$$P_1 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \\ & & \\ & & \\ & & \\ & & \end{bmatrix}, \; P_2 = \begin{bmatrix} & & \\ 1 & & \\ & 1 & \\ & & 1 \\ & & \\ & & \\ & & \end{bmatrix}, \text{ and } P_3 = \begin{bmatrix} & & \\ & & \\ & & \\ 1 & & \\ & 1 & \\ & & 1 \\ & & \end{bmatrix}. \tag{2.13}$$

Here, for all aggregates, $j = 1, 2, 3$, the restricted matrices are identical:

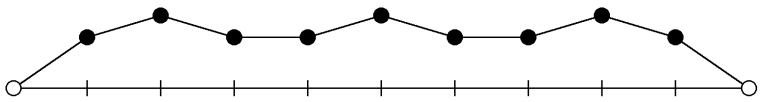
$$A_{\mathcal{V}} = \begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix} \quad \text{and} \quad B_{\mathcal{V}} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}. \tag{2.14}$$

Hence, solutions to the restricted eigenproblems are all of the form $\tilde{\mathbf{y}}_1 = \omega_j [\frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}]^T$, with a scaling term $\omega_j \neq 0$. So the initial guess developed is the vector

$$\tilde{\mathbf{v}} = [\frac{\omega_1}{\sqrt{2}}, \omega_1, \frac{\omega_1}{\sqrt{2}}, \frac{\omega_2}{\sqrt{2}}, \omega_2, \frac{\omega_2}{\sqrt{2}}, \frac{\omega_3}{\sqrt{2}}, \omega_3, \frac{\omega_3}{\sqrt{2}}]^T. \tag{2.15}$$

Note that the type of subspace corrections we use in the following section allows arbitrary scaling values ω_1. For the case $\omega_j = 1$ for all three aggregates, the initial guess is seen in figure 2.1.

FIG. 2.1. *Initial guess for the 1D model problem produced by the initial guess development algorithm; the RQ has been minimized over each aggregate individually.*



In the context of multigrid, initial guess development is used in place of pre-relaxation for the first GES-SA multigrid cycle performed. Subsequent pre-relaxations

and post-relaxations are presented as multiplicative local subspace correction in section 2.3. We now describe how smoothed aggregation is used to phrase global subspace corrections.

2.2. Global Coarse-Grid Correction using Smoothed Aggregation. Typically, smoothed aggregation has been used to form grid transfer operators within multigrid schemes for linear systems, as in [3], [10], and [12]. Here, we use smoothed aggregation in a similar fashion to form coarse, global subspaces of lower dimension that are used to form subspace corrections with lower RQ.

Smoothed aggregation defines a sparse $n \times m$ interpolation operator, P_c^f, that maps from a coarse set of m variables to the original fine set of n variables. We first partition the domain using the same aggregation that was used for linear SA solvers in the section 1. Note here, the number of coarse degrees of freedom, m, is equal to the number of aggregates, J, in the aggregation that we use. As in the general subspace correction method introduced in the beginning of section 2, P_c^f is designed so that its range, $\mathcal{V}$, has vectors with lower RQ than the current iterate, $\tilde{\mathbf{v}}$. Using a subspace correction with this interpolation improves the iterate. This section describes the process of forming P_c^f and $\mathcal{V}$ using smoothed aggregation.

The aggregation matrix, $[\mathcal{A}]$, defined in section 1.2 is used as a template for the sparsity structure of the interpolation operator we create, as in the linear solver case. Assume that we have our iterate $\tilde{\mathbf{v}}$, or current approximate solution. To assure the range of interpolation contains vectors with a RQ that is less than or equal to that of $\tilde{\mathbf{v}}$, we first form tentative interpolation, $\hat{P}$, that has $\tilde{\mathbf{v}}$ in its range. To do so, we use $[\mathcal{A}]$ to define the sparsity pattern of $\hat{P}$ by injecting $\tilde{\mathbf{v}}$ into that sparsity pattern:

$$\hat{P}_{ij} := \left\{ \begin{array}{ll} v_i, & [\mathcal{A}]_{ij} = 1 \\ 0, & [\mathcal{A}]_{ij} = 0 \end{array} \right\}, \tag{2.16}$$

where v_i denotes the i-th entry of $\tilde{\mathbf{v}}$. For our $n = 9$, 1D example, this injection is the following:

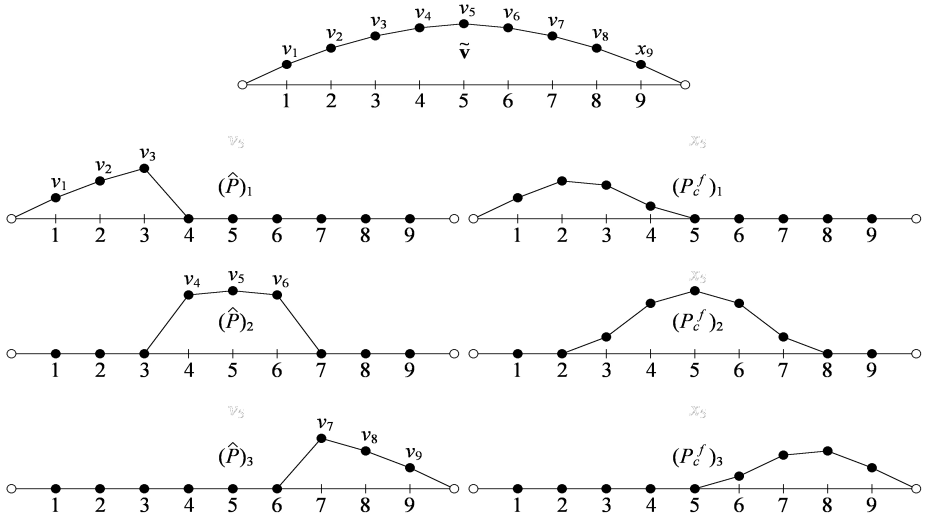
$$\tilde{\mathbf{v}} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix} \quad \longrightarrow \quad \hat{P} = \begin{bmatrix} v_1 & & \\ v_2 & & \\ v_3 & & \\ & v_4 & \\ & v_5 & \\ & v_6 & \\ & & v_7 \\ & & v_8 \\ & & v_9 \end{bmatrix}. \tag{2.17}$$

Operator $\hat{P}$ is such that $\tilde{\mathbf{v}} \in \mathcal{R}(\hat{P})$. Specifically, $\tilde{\mathbf{v}} = \hat{P}\,\mathbf{1}_m$, where $\mathbf{1}_m$ is the column vector of all ones with length m. This of course means that that we are guaranteed to have a vector within $\mathcal{R}(\hat{P})$ with no larger a RQ than that of $\tilde{\mathbf{v}}$:

$$\min_{\mathbf{v} \in \mathcal{R}(\hat{P})} \rho_{A,B}(\mathbf{v}) \leq \rho_{A,B}(\tilde{\mathbf{v}}). \tag{2.18}$$

To attempt to improve the minimal RQ within $\mathcal{V}$ with significantly lower RQ, we smooth interpolation. A smoothing operator, S, or smoother, is introduced for this purpose. Note that this smoothing process is just a local relaxation process to be applied to the columns of the interpolation. Many types of smoothing could be

FIG. 2.2. *Iterate* $\tilde{\mathbf{v}}$ *is aggregated and smoothed to form the columns of* P_c^f *for example 1 with* $n = 9$. *The top vector in the figure is a typical approximate minimal eigenvector* $\tilde{\mathbf{v}}$. *Each of the lower-left three vectors,* $(\hat{P})_j$, *is the* j*-th column of tentative interpolation matrix* $\hat{P}$ *formed by restricting* $\tilde{\mathbf{v}}$ *to the* j*-th aggregate. Each of the lower-right three vectors,* $(P_c^f)_j$, *is the* j*-th column of smoothed interpolation matrix* P_c^f *formed by smoothing* $\hat{P}$.



employed here, but it is best to use a smoother that does not significantly expand the support of the columns of $\hat{P}$. Typically, one application of the error propagation operator of Richardson of damped-Jacobi smoothing is performed, as in smoothing for the linear case. In this study, we use Richardson smoothing:

$$S := (I_n - \alpha A), \tag{2.19}$$

where I_n is the identity operator on $\mathbb{R}^n$ and α is chosen to be $\frac{4}{3\|A\|_2}$. This is the same value of α that is chosen for linear SA in [11], and our experience shows that it works well for the nonlinear subspace corrections as well.

Normalization of the columns of $\hat{P}$ is also performed. This does not change the range of interpolation, but does control the scaling of the coarse-grid problems. We choose a diagonal normalization matrix, N, so that the restricted, right-hand matrix, $B_{\mathcal{V}}$, has a diagonal that is the identity matrix of dimension m:

$$N_{ii} := \frac{1}{\|S(\hat{P})_i\|_B}. \tag{2.20}$$

We now define the interpolation matrix by

$$P_c^f := S\hat{P}N. \tag{2.21}$$

For our 1D example with $n = 9$, the sparsity structure of P_c^f is as follows:

$$P_c^f = \begin{bmatrix} \times & & \\ \times & & \\ \times & \times & \\ \times & \times & \\ & \times & \\ & \times & \times \\ & \times & \times \\ & & \times \\ & & \times \end{bmatrix}, \tag{2.22}$$

where each $\times$ represents a nonzero entry at its corresponding location. Figure 2.2 gives a full picture of how $\tilde{\mathbf{v}}$ is used to form the columns of $\hat{P}$ and P_c^f for the 1D example.

The columns of P_c^f form a basis for $\mathcal{V}$ because our construction ensures that there is at least one point in the support of each column that is not present in any other column. Forming aggregates that are at least a neighborhood in size and only smoothing interpolation once does not allow columns to ever share support with an aggregate's central node. We then compute $A_{\mathcal{V}} = (P_c^f)^T A P_c^f$ and $B_{\mathcal{V}} = (P_c^f)^T B P_c^f$. In multigrid vocabulary, restricted problem (2.9) is now the *coarse-grid problem*. The coarse-grid correction is given by interpolating the solution of the coarse-grid problem, $\tilde{\mathbf{w}} \leftarrow P_c^f \mathbf{y}_1$. This problem, $A_{\mathcal{V}}\mathbf{y} = \mu B_{\mathcal{V}}\mathbf{y}$, is either solved using a standard eigensolver or phrased as a coarse-grid minimization problem as in (2.8), where local and global subspace corrections may be applied in a recursive fashion. This offers approximate solutions to the coarse-grid problem as in a standard nonlinear multigrid method. Note that there is no algorithm presented in this subsection. However, algorithm 3 of section 3, the full GES-SA algorithm, uses coarse-grid corrections within it.

Under this construction, $S\tilde{\mathbf{v}}$ is in the range of P_c^f. Therefore, if $S\tilde{\mathbf{v}}$ has lower RQ than that of $\tilde{\mathbf{v}}$, we have guaranteed that subspace correction improves the RQ of our iterate. There is no way to ensure that our choice of smoother will lower the RQ of $\tilde{\mathbf{v}}$ (consider an eigenvector), however, our experience shows that it does for the types of vectors we use in the problems we have considered. Note that a choice of α could be computed to minimize the RQ of $\tilde{\mathbf{v}}$, a single vector in the range of interpolation. However, this choice of α may not be the best choice for all the other vectors in the range. We want to avoid having high RQ components represented by interpolation, because this causes ill-conditioned coarse-grid problems. Also, the vector of minimal RQ we select is most likely not merely $S\tilde{\mathbf{v}}$, but a vector of even lower RQ. Otherwise, the coarse-grid correction is unnecessary. The choice of $\alpha = \frac{4}{3\|A\|_2}$ aims to make both minimum and maximum RQs within $\mathcal{V}$ small, and its success is verified in section 4.

No matter how the coarse-grid problem is solved, it is designed to provide a correction that has the global traits of the actual minimal eigenvector. The local traits, however, still need to be corrected. Thus, we next develop a complementary process to selecting local subspaces for corrections that give local traits of approximate minimal eigenvectors resembling the actual minimal eigenvector.

2.3. Nonlinear Relaxation by Local Subspace Corrections. In the context of a nonlinear multilevel method, we use subspace corrections posed over locally supported subsets as our relaxation process, which is basically just nonlinear block-Gauss Seidel for minimizing the RQ. This section explains the specifics for choosing the nodes that make up each block, and presents the relaxation algorithm.

The original generalized eigenvalue problem, $A\mathbf{v} = \lambda B\mathbf{v}$, is posed over a set of n nodes, Ω_n. To choose a subspace that provides a local correction over a small cluster of n_j nodes, we construct $\mathcal{W}_j$ to be a subset of Ω_n, with cardinality n_j. Subset $\mathcal{W}_j$ should be local and connected within the graph of A. Subspace $\mathcal{V}_j$ is chosen to be the space of all vectors that only differ from a constant multiple of our current approximation, $\tilde{\mathbf{v}}$, by $\mathbf{w}$, a vector with support in the subset $\mathcal{W}_j$:

$$\mathcal{V}_j := \{\mathbf{v} \in \mathbb{R}^n \mid \mathbf{v} = w_0\tilde{\mathbf{v}} + \mathbf{w} \text{ where } w_0 \in \mathbb{R} \text{ and } \operatorname{supp}(\mathbf{w}) \subset \mathcal{W}_j\}, \tag{2.23}$$

a subspace of $\mathbb{R}^n$ with dimension $(n_j + 1)$ used to form and solve (2.6) for an updated approximation, $\tilde{\mathbf{w}}$, that has a minimum RQ within $\mathcal{V}_j$. Note that $\mathcal{W}_j$ is a subset of the full set of nodes, while $\mathcal{V}_j$ is a subspace of $\mathbb{R}^n$. We effectively change the value of current iterate $\tilde{\mathbf{v}}$ at only nodes in set $\mathcal{W}_j$ to minimize RQ, while leaving $\tilde{\mathbf{v}}$ unchanged at nodes outside of $\mathcal{W}_j$, up to a scaling factor, w_0. We now explain how subsets $\mathcal{W}_j$ are chosen, and then explain how the corrections are performed.

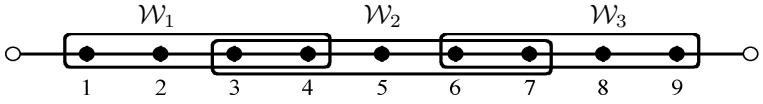
One step of the local subspace relaxation minimizes the approximate eigenvalue locally over one small portion of the full set of nodes, Ω_n. We want to construct a sequence of local subspace corrections so that the iterate is somewhat locally minimal over the entire set of nodes. To do so, we need a sequence of overlapping subsets $\{\mathcal{W}_j\}_{j=1}^J$ that form an overlapping covering of Ω_n. We then perform local subspace correction with each of these subsets in sequence until a correction has been done within each. The covering is called an overlapping subset covering of Ω_n, and is defined similarly to an aggregation, except that sufficient overlap is required in place of disjointness.

DEFINITION 2.2. *A sequence of J subsets, $\{\mathcal{W}_j\}_{j=1}^J$, is an* **overlapping subset covering** *of Ω_n with respect to A if the following properties hold.*

- *Overlap: For any $j \neq k$, if point $p \in \mathcal{W}_j$ and $A_{pq} \neq 0$ for some $q \in \mathcal{W}_k \setminus \mathcal{W}_j$, then we require $p \in \mathcal{W}_k$ as well.*
- *Covering: $\bigcup_{j=1}^J \mathcal{W}_j = \Omega_n$.*
- *Connected: For any j, if points $p, q \in \mathcal{W}_j$, then there exists a sequence of points in $\mathcal{W}_j$ that connect p to q within the graph of A.*

Figure 2.3 displays an overlapping subset covering for our 1D example.

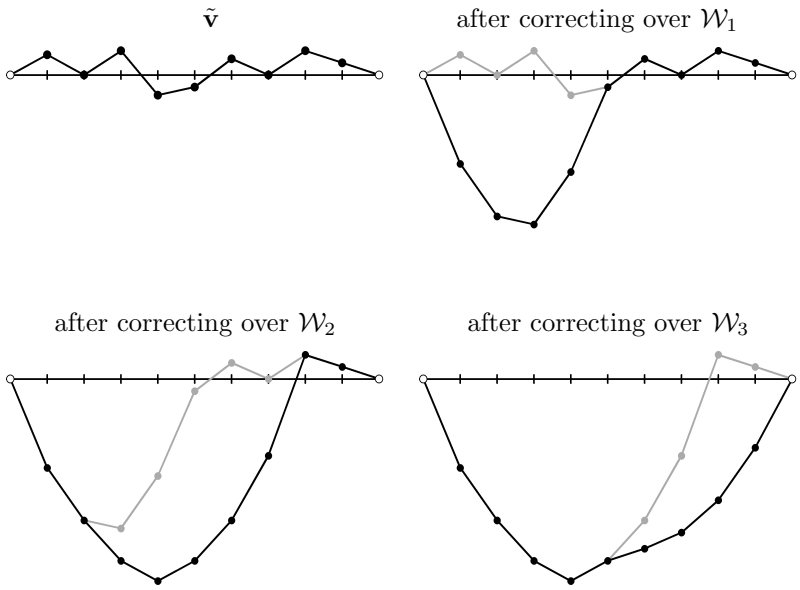
FIG. 2.3. *Graph of matrix A from example 1 with n = 9, grouped into three overlapping subsets. Each cartouche encloses the group of unknowns in the respective subset.*



Similar to aggregation matrix $[\mathcal{A}]$, we represent these subset coverings with a sparse, binary, *overlapping subset matrix*, $[\mathcal{W}]$. One way to obtain an overlapping subset decomposition is to dilate an aggregation. This is accomplished by taking each aggregate $\mathcal{A}_j$ within the aggregation and expanding $\mathcal{A}_j$ once with respect to the graph of matrix A. Let $[A]$ be an $n \times n$ binary version of A that stores connections in the graph of A, defined as

$$[A]_{ij} := \left\{ \begin{array}{ll} 1, & A_{ij} \neq 0 \\ 0, & A_{ij} = 0 \end{array} \right\}. \tag{2.24}$$

FIG. 2.4. *How a typical local subspace relaxation sweep acts on a random iterate for the 1D example with $n = 9$. The top left vector is the initial iterate, $\tilde{\mathbf{v}}$; top right shows a subspace correction on subset $\mathcal{W}_1$, bottom left shows a subsequent subspace correction over $\mathcal{W}_2$, and bottom right shows final relaxed iterate $\tilde{\mathbf{w}}$ after a subsequent subspace correction over $\mathcal{W}_3$.*



Then define $[\mathcal{W}]$ by creating a binary version of the matrix product $[A][\mathcal{A}]$, a *dilation*,

$$[\mathcal{W}]_{ij} := \left\{ \begin{array}{ll} 1, & \big([A][\mathcal{A}]\big)_{ij} \neq 0 \\ 0, & \big([A][\mathcal{A}]\big)_{ij} = 0 \end{array} \right\}. \tag{2.25}$$

We see no reason to insist that the overlapping subsets have anything to do with the aggregates, except for simplicity and convenience. Under this construction, the overlapping subset matrix, $[\mathcal{W}]$, has the same sparsity structure as the smoothed interpolation operator associated with the global coarse-grid correction, P_c^f. This construction gives us the characteristic of the covering having a 2-point overlap for every neighboring pair of local subsets. For our 1D example in section 2, $[\mathcal{W}]$ looks exactly like (2.22) with a 1 in the place of each $\times$:

$$[\mathcal{W}] = \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & 1 & \\ 1 & 1 & \\ & 1 & \\ & 1 & 1 \\ & 1 & 1 \\ & & 1 \\ & & 1 \end{bmatrix}, \tag{2.26}$$

In practice, each local correction is accomplished by rephrasing minimization problem (2.6) as a generalized eigenvalue problem of low dimension, as in (2.9), and

solving for minimal eigenvector $\mathbf{y}_1$ with a standard eigensolver. Note: here, we use Q_j to represent interpolation matrices that span each subspace, $\mathcal{V}_j$, to distinguish from the P_j and $\mathcal{S}_j$ used in the initial guess section. To construct the basis, Q_j, for $\mathcal{V}_j$ that we need to phrase (2.9), recall there are n_j nodes represented by $\mathcal{W}_j$. We construct a $n \times (n_j + 1)$ matrix, Q_j, so that its columns are an orthogonal basis for subspace $\mathcal{V}$. To define Q_j explicitly, first define vector $\mathbf{v}_0$ by

$$(\mathbf{v}_0)_i := \left\{ \begin{array}{ll} v_i, & i \notin \mathcal{W}_j \\ 0, & i \in \mathcal{W}_j \end{array} \right\}. \tag{2.27}$$

For each point $p \in \mathcal{W}_j$, define elementary basis vectors $\hat{\mathbf{e}}_p$ as in (2.11). Then, we form Q_j by appending these $(n_j + 1)$ vectors in a matrix of column vectors:

$$Q_j = \left[\begin{array}{c|ccc} \top & \top & & \top \\ \mathbf{v}_0 & \hat{\mathbf{e}}_{p_1} & \cdots & \hat{\mathbf{e}}_{p_{n_j}} \\ \bot & \bot & & \bot \end{array} \right], \tag{2.28}$$

where the sequence of points, $\{p_i\}_{i=1}^{n_j}$, is a list of all points within local subset $\mathcal{W}_j$. This makes the columns of Q_j orthogonal, a matrix that maps from $\mathbb{R}^{n_j+1}$ onto $\mathcal{V}_j$. For the 1D example, with $\mathcal{W}_2 = \{3, 4, 5, 6, 7\}$, the interpolation operator is given by

$$Q_j = \begin{bmatrix} v_1 & & & & & & \\ v_2 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ v_8 & & & & & & \\ v_9 & & & & & & \end{bmatrix}. \tag{2.29}$$

Next, we compute

$$A_{\mathcal{V}} \leftarrow Q_j^T A Q_j \qquad \text{and} \qquad B_{\mathcal{V}} \leftarrow Q_j^T B Q_j. \tag{2.30}$$

Although Q_j is not sparse, due to its first column, all J of the triple products in (2.30) can be efficiently implemented in $\mathcal{O}(n) + \sum_{j=1}^{J} \mathcal{O}(n_j)$ operations. Then (2.9) is solved for $\mathbf{y}_1$ normalized such that $(\mathbf{y}_1)_1 = 1$. This normalization is the same as requiring $w_0 = 1$, which leaves all nodes outside of $\mathcal{W}_j$ unchanged by the correction. Then, the local subspace correction is given by interpolating $\tilde{\mathbf{w}} \leftarrow P\mathbf{y}_1$. This problem has dimension $(n_j + 1) << n$, so it is solved cheaply with an standard eigensolver for a sufficiently small subspace.

Local subspace relaxation is summarized in the following algorithm, a component of the full GES-SA algorithm referred to in algorithm 3 of section 3.

ALGORITHM 2 : Local Subspace Relaxation
Function: $\tilde{\mathbf{w}} \leftarrow LSR(A, B, \tilde{\mathbf{v}}, \{\mathcal{W}_j\}_{j=1}^{J})$.
Input: SPD matrices A and B, current approximation to the minimal eigenvector $\tilde{\mathbf{v}}$, and overlapping subset covering $\{\mathcal{W}_j\}_{j=1}^{J}$.
Output: corrected iterate $\tilde{\mathbf{w}}$.

1. For $j = 1, ..., J$, do the following:

(a) Form Q_j based on $\tilde{\mathbf{v}}$ and $\mathcal{W}_j$ as in (2.28).
(b) Form $A_{\mathcal{V}} = Q_j^T A Q_j$ and $B_{\mathcal{V}} = Q_j^T B Q_j$.
(c) Find $\mathbf{y}_1$ by solving (2.9) via a standard eigensolver.
(d) Normalize and interpolate $\tilde{\mathbf{w}} \leftarrow Q_j \mathbf{y}_1$.
(e) Update $\tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{w}}$.

2. Output $\tilde{\mathbf{w}}$.

Figure 2.3 shows how a single sweep of local subspace relaxation acts on a random initial guess, $\tilde{\mathbf{v}}$. Although the guess is never really random in the actual algorithm, we show this case so it is clear how the algorithm behaves. This algorithm gives relaxed iterate $\tilde{\mathbf{w}}$ local characteristics of the actual minimal eigenvector. For problems with large numbers of nodes, the global characteristics of the iterate are far from those of the actual minimal eigenvector. This is where the coarse-grid correction complements local subspace relaxation. When done in an alternating sequence, as in a standard multigrid method, the complementary processes correct both local and global characteristics of the approximate minimal eigenvector, forming an eigensolver. Their explicit use is presented in the next section.

3. The full GES-SA algorithm. Because GES-SA is a multilevel method, to describe it, we change to multilevel notation. Any symbol with subscript l refers to an object on grid l, with $l = 1$ the finest or original grid and $l = L$ the coarsest. The matrix on the l-th grid is written A_l; in particular, $A_1 = A$, the matrix from our original problem. Interpolation from grid $l+1$ to grid l is written P_{l+1}^l instead of P_c^f, and restriction from grid l to grid $l+1$ is denoted P_l^{l+1}. Again, we use $P_l^{l+1} = (P_{l+1}^l)^T$, because A_l is SPD. The dimension of A_l is written n_l. Several other level l objects are denoted with subscript and superscripts l, as well.

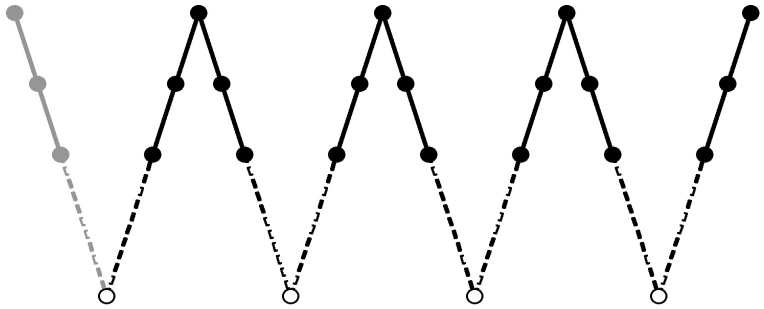
The full GES-SA algorithm is a nonlinear multigrid method that uses standard components, relaxation and coarse-grid correction, in the way they are used for linear multgrid solvers with some differences. First, both are deigned to minimize RQ over a subspace, versus minimizing the error. Another crucial difference is that interpolation is redefined for each and every coarsening, making each cycle more expensive than those of linear solvers. Therefore, we intend to do a very small number of cycles. In fact, the results in this paper are all produced using one GES-SA cycle.

Here, the relaxation process is accomplished by using the local subspace correction of section 2.3, while the coarse-grid correction is done using global corrections formed with smoothed aggregation interpolation, as in section 2.2. Also, no initial guess is provided to the eigensolver. Instead, during the first cycle, at each level, an initial guess is developed by the solver in place of pre-relaxation with local subspace correction. For the post-relaxations, and pre-relaxations of subsequent cycles, local subspace correction is used, as pictured in figure 3.1.

Several parameters are involved with the GES-SA algorithm: ν dictates how many relaxations are done at a given stage of the algorithm, with $\nu = 2$ as our standard; η is the total number of cycles; and γ is the number of sub-iterations, with $\gamma = 1$ giving a V-cycle (as in figure 3.1), and $\gamma = 2$ giving a W-cycle.

ALGORITHM 3 : Generalized Eigensolver Based on Smoothed Aggregation
Function: $\tilde{\mathbf{v}}_l \leftarrow GESSA(A_l, B_l, \nu, \eta, \gamma, l, L)$.
Input: SPD matrices A_l and B_l, number of relaxations to perform ν, number cycles η, number of coarse-grid problem iterations γ, current level l, and coarsest level L. Assume that an aggregation and an overlapping subspace covering are available for each level l.

FIG. 3.1. *Diagram of how V-cycles are done in GES-SA for $\gamma = 1$. We follow the diagram from left to right as the algorithm progresses. Gray dots represent the initial guess development phase of the algorithm, only done on the first cycle where pre-relaxation is usually done. Hollow dots represent solve steps done with a standard eigensolver on the coarsest eigenproblem. Black dots represent local subspace pre- and post-relaxation steps. A dot on top stands for a step on the finest grid and a dot on bottom stands for a step on the coarsest grid.*



Output: approximate minimal eigenvector $\tilde{\mathbf{v}}_l$ to the level l problem.

0. If no aggregation of Ω_{n_l} is provided, compute $\{\mathcal{A}_j^l\}_{j=1}^{J_l}$. Also, if no overlapping subset covering is provided, compute $\{\mathcal{W}_j^l\}_{j=1}^{J_l}$.
1. For $\zeta = 1, ..., \eta$, do the following:
   (a) If $\zeta = 1$, form an initial guess, $\tilde{\mathbf{v}}_l \leftarrow IGD(A_l, B_l, \{\mathcal{A}_j^l\}_{j=1}^{J_l})$. Otherwise, pre-relax the current approximation, $\tilde{\mathbf{v}}_l \leftarrow LSR(A_l, B_l, \tilde{\mathbf{v}}^l, \nu, \{\mathcal{W}_j^l\}_{j=1}^{J_l})$.
   (b) Form P_{l+1}^l with SA based on $\tilde{\mathbf{v}}_l$ and $\{\mathcal{A}_j^l\}_{j=1}^{J_l}$ as in (2.21).
   (c) Form matrices $A_{l+1} \leftarrow (P_{l+1}^l)^T A_l P_{l+1}^l$ and $B_{l+1} \leftarrow (P_{l+1}^l)^T B_l P_{l+1}^l$.
   (d) If $(l+1) = L$, solve (2.9) for $\mathbf{y}_1$ with a standard eigensolver, and set $\tilde{\mathbf{v}}_{l+1} \leftarrow \mathbf{y}_1$. Else, $\tilde{\mathbf{v}}_{l+1} \leftarrow GESSA(A_{l+1}, B_{l+1}, \nu, \gamma, \gamma, l+1, L)$.
   (e) Interpolate the coarse-grid correction, $\tilde{\mathbf{v}}_l \leftarrow P_{l+1}^l \tilde{\mathbf{v}}_{l+1}$.
   (f) Post-relax the current approximation, $\tilde{\mathbf{v}}_l \leftarrow LSR(A_l, B_l, \tilde{\mathbf{v}}_l, \nu, \{\mathcal{W}_j^l\}_{j=1}^{J_l})$.
2. Output $\tilde{\mathbf{v}}_l$.

To describe the creation of a simple adaptive linear solver, assume that GES-SA has produced an approximate eigenvector, $\tilde{\mathbf{v}}_1$. Then, the setup phase of SA is called with this vector as a characterization of the near-kernel, $\mathcal{K} = \{\tilde{\mathbf{v}}_1\}$. The solve phase of SA is then run, starting with a random initial guess for $\mathbf{x}$. If the solver is still not adequate, more GES-SA cycles are run (without initial guess development) to lower the RQ of $\tilde{\mathbf{v}}_1$. Then, the setup phase is run again, and a new solver is formed. This one-vector process is repeated until the solver is adequate, or until the RQ of $\tilde{\mathbf{v}}_1$ is not being improved by much. If the latter case occurs, then it may be that more than one prototypical smooth error component should be developed. As of now, this is not a feature covered by the GES-SA approach, although an algorithm for developing secondary kernel components is developed in [3].

4. Numerical Results. Many linear systems that come from the discretization of scalar PDEs are solved with SA, with the vector of all ones as a near-kernel component, where the linear solver has decent convergence rates. However, we present examples of matrices where the vector of all ones is not a near-kernel component, and using it as one with SA produces a linear solver with unacceptable convergence rates. For all of these problems, one cycle of the GES-SA algorithm gives an approximate

minimal eigenvector that may be use to produce an acceptable linear SA solver.

EXAMPLE 2. *A random-signed discrete Laplacian.* Consider the d-dimensional Dirichlet-Poisson problem

$$\begin{aligned} -\Delta u &= f \quad \text{in} \quad \Omega = (0,1)^d \\ u &= 0 \quad \text{on} \quad \delta\Omega. \end{aligned} \tag{4.1}$$

We first discretize (4.1) with linear finite elements, or second-order finite differences, on equidistant rectilinear grids. For finite elements, in one-dimension, we use piecewise linear functions on equidistant intervals giving a 3-point stencil; in two dimensions, we use bilinears on equidistant squares giving a 9-point stencil; and in three dimensions, we use trilinears on equidistant cubes giving a 27-point stencil. For finite differences, in one dimension, central differences give a 3-point stencil; in two dimensions, they give a 5-point stencil; and in three dimensions, they give a 7-point stencil. The one-dimensional finite-difference case problem is neglected because it is identical to the one-dimensional finite element problem. Either way we discretize the problem, we have a sparse $n \times n$ matrix $\hat{A}$. We then define the diagonal, *random-signed* matrix $D_{\pm}$ to have randomly assigned positive and negative ones for entries. Finally, we form the random-signed discrete Laplacian matrix A by

$$A \leftarrow D_{\pm}\hat{A}D_{\pm}. \tag{4.2}$$

In our results, we also symmetrically scale the matrix A to have ones on its diagonal.

TABLE 4.1

Asymptotic convergence factors for the one-, two-, and three-dimensional finite element versions of the random-signed Dirichlet Laplacian problem. Aggregation was done geometrically. All factors were calculated using a geometric average of the last 5 of 25 linear SA multigrid $V(2,2)$-cycles. Factors in the column labeled "ones" correspond to solvers created using the vectors of all ones as algebraically smooth error; factors in the "ges-sa" column correspond to solvers that use our approximate minimal eigenvector computed with one RQ V-cycle with $\nu = 2$; and factors in the "eigen" column correspond to solvers that use the minimal eigenvector computed with a standard method. The last column, "comp", displays the operator complexity for the linear solver based on the GES-SA vector.

	prob. size	dimensions	ones	ges-sa	eigen	comp
1D	81	$n = 81^1$	0.987	0.204	0.193	1.461
	243	$n = 243^1$	0.984	0.218	0.214	1.484
	729	$n = 729^1$	0.994	0.218	0.218	1.494
	2187	$n = 2187^1$	0.991	0.219	0.220	1.498
	6561	$n = 6561^1$	0.989	0.221	0.220	1.499
	19683	$n = 19683^1$	0.989	0.221	0.221	1.500
	59049	$n = 59049^1$	0.990	0.221	0.221	1.500
2D	81	$n = 9^2$	0.620	0.074	0.074	1.078
	729	$n = 27^2$	0.892	0.176	0.179	1.108
	6561	$n = 81^2$	0.965	0.193	0.196	1.119
	59049	$n = 243^2$	0.977	0.215	0.214	1.123
3D	729	$n = 9^3$	0.598	0.114	0.111	1.054
	19683	$n = 27^3$	0.934	0.188	0.189	1.112

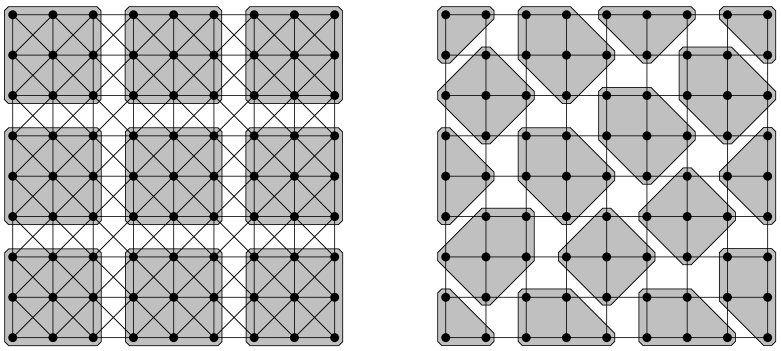
Now consider solving $A\mathbf{x} = \mathbf{b}$ given vector $\mathbf{b}$. Note that the vector of all ones is no longer algebraically smooth with respect to standard relaxation methods. As

TABLE 4.2

Asymptotic convergence factors for the two- and three-dimensional finite difference versions of the random-signed Dirichlet Laplacian problem. Aggregation was done algebraically. All factors were calculated using a geometric average of the last 5 of 25 linear SA multigrid $V(2,2)$-cycles. Factors in the column labeled "ones" correspond to solvers created using the vectors of all ones as algebraically smooth error; factors in the "ges-sa" column correspond to solvers that use our approximate minimal eigenvector computed with one RQ V-cycle with $\nu = 2$; and factors in the "eigen" column correspond to solvers that use the minimal eigenvector computed with a standard method. The last column, "comp", displays the operator complexity for the linear solver based on the GES-SA vector.

	prob. size	dimensions	ones	ges-sa	eigen	comp
2D	81	$n = 9^2$	0.849	0.219	0.219	1.317
	729	$n = 27^2$	0.947	0.294	0.290	1.357
	6561	$n = 81^2$	0.962	0.306	0.305	1.348
	59049	$n = 243^2$	0.978	0.312	0.312	1.342
3D	729	$n = 9^3$	0.825	0.289	0.292	1.389
	19683	$n = 27^3$	0.944	0.360	0.358	1.495
	64000	$n = 40^3$	0.961	0.418	0.413	1.511

FIG. 4.1. *Aggregation examples displayed for 2D test problems of low dimension. On the left is an aggregation formed with a geometric aggregation method that is used for the finite element problems; on the right is an aggregation formed with an algebraic aggregation method that is used for finite-difference problems. Each gray cartouche represents a separate aggregate that contains the nodes enclosed.*



geometric aggregation algebraic aggregation

seen in tables 4.1 and 4.2, using the vector of all ones produces SA solvers that have unacceptable convergence factors. Instead, we use one GES-SA cycle to produce an approximate minimal eigenvector, $\tilde{\mathbf{v}}$, and use $\mathbf{k}_1 = \tilde{\mathbf{v}}$ in the setup phase of SA to produce a linear SA solver. The convergence factors of the resulting solver are comparable to those obtained using the actual minimal eigenvector to build the linear SA solver. Note that convergence factors are reported as an estimation of asymptotic convergence factors by computing a geometric average of the last 5 of 25 V-cycles:

$$\text{Asymptotic Convergence Factor} \approx \left(\frac{\|\mathbf{e}^{(25)}\|_A}{\|\mathbf{e}^{(20)}\|_A} \right)^{1/5} \tag{4.3}$$

for the zero right-hand-side problem $A\mathbf{x} = \mathbf{0}$. Operator complexity is also reported for the linear solver that uses the vector developed with GES-SA. We define operator

complexity to be

$$\text{Operator Complexity} = \frac{\sum_{l=1}^{L} \text{nz}(A_l)}{\text{nz}(A_1)}, \tag{4.4}$$

where the function $\text{nz}(M)$ is the number of nonzeros in sparse matrix M.

Two types of aggregation were done. For the finite element problems, we took advantage of knowing the grid and formed aggregations that were blocks of 3^d nodes. This is fairly close to what would occur if we used algebraic aggregation with the nodes numbered lexicographically. For the finite difference problems, no geometric information was employed and aggregation was done algebraically. Small examples in 2 dimensions of the difference between the two types of aggregates we used are shown in figure 4.1 Also, we consider i to be *strongly connected* to unknown j with respect to the parameter $\theta \in (0, 1)$ if

$$|a_{ij}| > \theta \max_{k \neq i} |a_{ik}|. \tag{4.5}$$

Any connection that violates this requirement is a *weak connection*. Algebraic aggregations were done based on this strength-of-connection measure, with $\theta = .25$.

Although it is not the primary purpose of this study, it is also interesting to view GES-SA as a standalone eigensolver. For the random-signed Dirichlet-Laplacians, tables 4.3 and 4.4 display how one GES-SA V-cycle with $\nu = 2$ produces an approximate minimal eigenvector that is very close to the actual minimal eigenvector in the sense that the relative error between the RQ and the minimal eigenvalue is order 1.

All the results in this section are produced using only one GES-SA cycle. However, we do not believe that that a decent approximate minimal eigenvector can be produced with one GES-SA cycle for general problems. Note, the relative error of one cycle tends to increase as h decreases, or as the discretization error decreases. For most problems, we anticipate having to do more GES-SA cycles to achieve an acceptable approximate minimal eigenvector.

5. Conclusion. We described a multilevel generalized eigensolver, GES-SA, that uses corrections over various subspaces to minimize the RQ. This eigensolver was composed of complementary RQ subspace corrections. The process of creating the subspaces was centered around the concept of smoothed aggregation and it represents a new approach to eigenvalue problems. We also discussed how the multilevel eigensolver is used to adaptively create linear SA solvers. The numerical results in section 4 show that using GES-SA to form a linear SA solver results in one that has suitable convergence rates for the problems tested. This suggests that GES-SA is a decent initialization component for adaptively forming linear SA solvers. Our next goal is to create a version of GES-SA that is used in a fully adaptive way, creating secondary near-kernel components.

REFERENCES

[1] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput. Vol. 19, pp. 23-56, (1986)

[2] M. Brezina, *Robust iterative methods on unstructured meshes*, PhD Thesis, University of Colorado, Denver (1997)

[3] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge, *Adaptive Smoothed Aggregation (αSA)*, SISC, Volume 25, Issue 6. (2004) pages 1896-1920.

TABLE 4.3

Relative errors between the RQ of the GES-SA approximate minimal eigenvector, ρ, and the minimal eigenvalue, λ_1, for the one-, two-, and three-dimensional finite element versions of the random-signed Dirichlet-Laplacian problem. The GES-SA approximation was computed with one GES-SA V-cycle with $\nu = 2$.

	prob. size	dimensions	ρ	λ_1	rel. error
1D	81	$n = 81^1$	2.257e-02	2.256e-02	0.0002538
	243	$n = 243^1$	7.585e-03	7.584e-03	0.0001592
	729	$n = 729^1$	2.535e-03	2.535e-03	0.0000478
	2187	$n = 2187^1$	8.458e-04	8.458e-04	0.0000423
	6561	$n = 6561^1$	2.820e-04	2.820e-04	0.0000551
	19683	$n = 19683^1$	9.404e-05	9.401e-05	0.0002368
	59049	$n = 59049^1$	3.147e-05	3.134e-05	0.0040870
2D	81	$n = 9^2$	7.222e-02	7.222e-02	0.0000034
	729	$n = 27^2$	9.413e-03	9.412e-03	0.0001608
	6561	$n = 81^2$	1.101e-03	1.100e-03	0.0002491
	59049	$n = 243^2$	1.243e-04	1.243e-04	0.0001224
3D	729	$n = 9^3$	1.066e-01	1.066e-01	0.0000017
	19683	$n = 27^3$	1.412e-02	1.409e-06	0.0022805

TABLE 4.4

Relative errors between the RQ of the GES-SA approximate minimal eigenvector, ρ, and the minimal eigenvalue, λ_1, for the one-, two-, and three-dimensional finite difference versions of the random-signed Dirichlet Laplacian problem. The GES-SA approximation was computed with one GES-SA V-cycle with $\nu = 2$.

	prob. size	dimensions	ρ	λ_1	rel. error
2D	81	$n = 9^2$	4.895e-02	4.894e-02	0.0000582
	729	$n = 27^2$	6.307e-03	6.288e-03	0.0031257
	6561	$n = 81^2$	7.501e-04	7.338e-04	0.0222547
	59049	$n = 243^2$	9.306e-05	8.289e-05	0.1227465
3D	729	$n = 9^3$	4.896e-02	4.894e-02	0.0003230
	19683	$n = 27^3$	6.303e-03	6.288e-03	0.0024756
	64000	$n = 40^3$	2.981e-03	2.934e-03	0.0158771

[4] W. BRIGGS, V. E. HENSON, S. F. MCCORMICK, *A Multigrid Tutorial, 2nd Edition*, SIAM books, 2000.

[5] Z. CAI, J. MANDEL, AND S. MCCORMICK, *Multigrid methods for nearly singular linear equations and eigenvalue problems*, SIAM J. Numer. Anal. 34 (1997), pp. 178-200.

[6] T. F. CHAN, I. SHARAPOV, *Subspace correction multi-level methods for elliptic eigenvalue problems*, Numerical Linear Algebra with Applications, John Wiley and Sons, Ltd. 2002; 9:1 20

[7] S. F. MCCORMICK, J. RUGE, *Multigrid Methods for Variational Problems* SIAM J. Numer. Anal. Vol. 19, No. 5 (1982), pp. 925-929.

[8] J. RUGE, *Multigrid methods for variational and differential eigenvalue problems and unigrid for multigrid simulation*, PhD Thesis, Colorado State University, Fort Collins (1981)

[9] U. TROTTENBERG, C. W. OSTERLEE, A. SCHULLER, *Multigrid*, Academic Press, 2000.

[10] P. VANEK, J. MANDEL, M. BREZINA, *Algebraic multigrid on unstructured meshes*, in University of Colorado Technical Reports, University of Colorado at Denver, 1994

[11] P. VANEK, J. MANDEL, M. BREZINA, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, Vol. 88, pp. 559-579, (2001) ,

[12] P. VANEK, M. BREZINA, J. MANDEL, *Algebraic multigrid by smoothed agregation for second*

and fourth order elliptic problems, in Computing, Springer Wien, Volume 56, Number 3, pp. 179 - 196