NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2007; **07**:1–6 *Prepared using* nlaauth.cls *[Version: 2002/09/18 v1.02]*

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. Vassilevski[‡]

[†] *Department of Applied Mathematics, University of Colorado at Boulder, UCB 526, Boulder, CO, 80309-0526, USA*
[‡] *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, 7000 East Avenue, Mail Stop L-560, Livermore, CA, 94550, USA*

SUMMARY

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 many common relaxation processes, 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. We discuss 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 may be applied as a stand-alone eigensolver for applications that desire an approximate minimal eigenvector, but the primary purpose here is to apply an eigensolver to the specific application of forming robust, adaptive linear solvers. This paper reports the first stage in our study of incorporating eigensolvers into the existing adaptive SA framework.

KEY WORDS: generalized eigensolver; smoothed aggregation; multigrid; adaptive solver;

1. Introduction

In the spirit of algebraic multigrid (AMG; [1, 2, 5, 14, 15]), smoothed aggregation multigrid (SA; [17]) has been designed to solve a linear system of equations with little or no prior knowledge regarding the geometry or physical properties of the underlying problem. Therefore, SA is often an efficient solver for problems discretized on unstructured meshes, with varying

*Correspondence to: G. Sanders, Department of Applied Mathematics, University of Colorado at Boulder, UCB 526, Boulder, CO, 80309-0526, USA. E-mail: sandersg@colorado.edu
[†]University of Colorado at Boulder and Front Range Scientific Computing
[‡]The 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

Grants

Received 17 May 2007
Revised 16 November 2007

coefficients, or with no associated geometry. The relaxation processes commonly used in multigrid solvers are computationally cheap, but commonly 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 prototype vectors, the SA framework constructs intergrid transfer operators that allow such error to be eliminated on coarser grids, where relaxation is more economical. For example, in a three-dimensional elasticity problem, six such components (the so-called rigid body modes) form an adequate characterization of the algebraically smooth error. Rigid body modes are often available from discretization packages, and a solver can be produced with these vectors in the SA framework [17]. However, such a characterization is not always readily available (even for some scalar problems) and must be developed in an adaptive process.

Adaptive SA (αSA), as presented in [4], was designed specifically to create a representative set of vectors for 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 prototype vector that is algebraically smooth with respect to relaxation. Using this one resulting component, the SA framework is employed to construct a linear multigrid solver, and the whole process can be repeated with the updated solver playing the role of relaxation on each multigrid level. 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.

We consider applying SA to an algebraic system of equations $A\mathbf{x} = \mathbf{b}$, where $A = (a_{ij})$ is an $n \times n$ symmetric, positive definite matrix that is symmetrically scaled so that its diagonal entries are all ones. For simplicity, we use damped-Jacobi for our initial relaxation. The SA framework provides an interpolation operator, P, that is used to define a coarse level with standard Galerkin variational corrections. If the relaxation process is a convergent iteration, then it is known from the literature (e.g. [1, 11]) that a sufficient condition for two-level convergence factors bounded from one is that for any $\mathbf{u}$ on the fine-grid, there exists a $\mathbf{v}$ from the coarse-grid such that

$$\|\mathbf{u} - P\mathbf{v}\|_2^2 \leq \frac{C}{\|A\|_2}(A\mathbf{u}, \mathbf{u}), \tag{1}$$

with some constant C. The quality of the bound on convergence factor depends on the size of C, as shown in [3]. This requirement is known in the literature as *the weak approximation property*, and reflects the observation noted in [13, 11] that any *minimal eigenvector* (an eigenvector associated with the smallest eigenvalue) of A needs to be interpolated with accuracy inversely proportional to the size of its eigenvalue. For this reason, this paper proposes a Generalized EigenSolver based on Smoothed Aggregation (GES-SA) to approximate a minimal eigenvector of A.

Solving an eigenvalue problem as an efficient means to developing a linear solver may appear counterintuitive. However, we aim to compute only an appropriately accurate approximation of the minimal eigenvector to develop an efficient linear solver with that approximation at $\mathcal{O}(n)$ cost. In this context, many existing efficient methods for generating a minimal eigenvector are appealing (see [9] and [12] for short lists of such methods). Here, we propose GES-SA because it takes advantage of the same data-structures as the existing αSA framework. Our intention is to eventually incorporate GES-SA into the αSA framework to enhance robustness of our adaptive solvers for difficult problems that may benefit from such enhancement (such

as systems problems, corner-singularity problems, or problems with geometrically oscillatory near-kernel).

The GES-SA algorithm performs a series of iterations that minimize the Rayleigh quotient over various subspaces, as discussed in the later sections. In short, GES-SA is a variant of algebraic Rayleigh quotient multigrid (RQMG [6]) that uses overlapping block Rayleigh quotient Gauss-Seidel for its relaxation process and smoothed aggregation Rayleigh quotient minimization for coarse-grid updates. In [8], Hetmanuik developed an algebraic RQMG algorithm that performs point Rayleigh quotient Gauss-Seidel for relaxation and coarse-grid corrections based on a hierarchy of static intergrid transfer operators that are supplied to his algorithm. This supplied hierarchy is assumed to have adequate approximation properties. In contrast, GES-SA initializes the hierarchy of intergrid transfer operators and modifies it with each cycle, with the goal of developing a hierarchy with adequate approximation properties, as in the setup phase of αSA. This is discussed in more detail in section 3.2.

This paper is organized as follows. 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 $A\mathbf{x} = \lambda B\mathbf{x}$. Matrix A is the 1D Laplacian with Dirichlet boundary conditions, discretized with equidistant, second-order central differences, symmetrically scaled so that the diagonally entries are all ones:

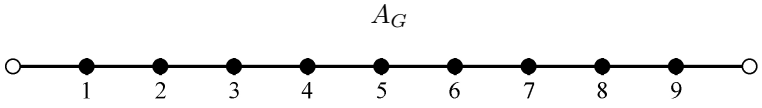
$$A = \frac{1}{2}\begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & & \ddots & & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}, \tag{2}$$

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 $\Omega_n = \{1, 2, \ldots, n\}$. 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 version presented in [17, 16]. Given a relaxation process and a set of vectors $\mathcal{K}$ characterizing algebraically smooth error, the SA setup phase produces a multigrid hierarchy that defines a linear solver.

Figure 1. Graph of matrix A_G from example 1 with $n = 9$. The nine nodes are enumerated, edges of the graph represent nonzero off-diagonal entries in A, and the Dirichlet boundary conditions are represented with the hollow dots at the end points.



For symmetric problems, such as those we consider here, standard SA produces a coarse grid using interpolation operator P and restriction operator, $R = P^T$. This gives the variational (or Galerkin) coarse-grid operator, $A_c = P^T AP$, commonly used in AMG methods. This process is repeated recursively on all grids, constructing a multigrid hierarchy. The interpolation operator is produced by applying a smoothing operator, S, to a tentative interpolation operator, $\hat{P}$, that satisfies the weak approximation property.

At the heart of forming $\hat{P}$ is a discrete partitioning of fine-level nodes into a disjoint covering of *the full set of nodes*, $\Omega_n = \{1, 2, ..., n\}$. Members of this partition are locally grouped based on matrix A_G, representing the *graph of strong connections* [17]. A_G is created by filtering the original problem matrix A with regard to strength of coupling. For the scalar problems considered here, we define node i to be strongly connected to node j with respect to the parameter $\theta \in (0, 1)$ if

$$|a_{ij}| > \theta \sqrt{a_{ii} a_{jj}}. \tag{3}$$

Any connection that violates this requirement is a *weak connection*. Entry $(A_G)_{ij} = 1$ if the connection between i and j is strong, and $(A_G)_{ij} = 0$ otherwise.

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

- Covering: $\bigcup_{j=1}^m \mathcal{A}_j = \Omega_n$.
- 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_G.

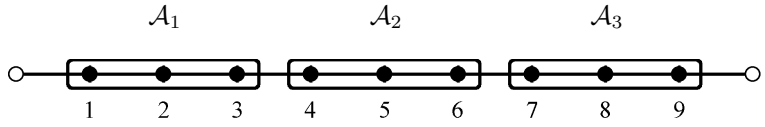
Each individual subset $\mathcal{A}_j$ within the aggregation is called an *aggregate*.

The method we use to form aggregations is given in [17], where each aggregate has a *central node*, or *seed*, numbered i, and covers this node's entire *strong neighborhood* (the support of the i-th row in graph of A_G). This is a very common way of forming aggregations for computational benefits, but is not mandatory. We return to example 1 to explain the aggregation concept. An acceptable aggregation of Ω_9 with respect to A would be $m = 3$ aggregates, each of size 3, defined as follows:

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

It is easily verified that this partitioning satisfies definition 1.1. This aggregation is pictured in figure 2. Two-dimensional examples are presented in section 4.

Figure 2. Graph of matrix A_G from example 1 with $n = 9$, split into three aggregates. Each box encloses a group of nodes in its respective aggregate.



We find it useful to represent an aggregation $\{\mathcal{A}_j\}_{j=1}^m$ 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 a one in the (i, j)-th entry if point i is contained in aggregate $\mathcal{A}_j$, and a zero otherwise. In our 1D example, with $n = 9$, we represent the aggregation given in (4) as

$$[\mathcal{A}] = \begin{bmatrix} 1 & & \\ 1 & & \\ 1 & & \\ & 1 & \\ & 1 & \\ & 1 & \\ & & 1 \\ & & 1 \\ & & 1 \end{bmatrix}. \tag{5}$$

Based on the sparsity structure of $[\mathcal{A}]$, the SA setup phase constructs $\hat{P}$ 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 $[\mathcal{A}]$.

Under the above construction, the vectors in $\mathcal{K}$ are ensured to be in $\mathcal{R}(\hat{P})$, the range of the tentative interpolation operator, and are therefore well 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 may lead to a coarse-grid operator with higher condition number than desired. This degrades the effect of coarse-grid relaxation on vectors that are moderately algebraically smooth. Of greater importance, some algebraically smooth vectors are typically not well-represented by $\mathcal{R}(\hat{P})$ and are therefore not reduced by coarse-grid corrections. To remedy the situation, SA does not use $\hat{P}$ as its interpolation operator directly, but instead utilizes a smoothed version, $P = S\hat{P}$, where S is an appropriately chosen polynomial smoothing operator. As a result, a much richer set of algebraically smooth error is accurately represented by the coarse grid. A typical choice for S is one step of the error propagation operator of damped-Jacobi relaxation. In this paper, we use damped-Jacobi smoothing under the assumption that the system is diagonally scaled so that diagonal elements are one.

The underlying set, $\mathcal{K}$, that induces a linear SA solver can either be supplied, as in standard SA, or computed, as in adaptive SA methods. We now describe a new approach to constructing $\mathcal{K}$ that can be used within the existing αSA framework.

2. Rayleigh Quotient Minimization within Subspaces

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:

$$\begin{array}{c} \text{find an eigenvector, } \mathbf{v}_1 \neq \mathbf{0}, \text{ corresponding to the} \\ \text{smallest eigenvalue, } \lambda_1, \text{ in the problem } A\mathbf{v} = \lambda B\mathbf{v}. \end{array} \tag{6}$$

For 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 (6), an approach that has been used in [6] and [7], to introduce our method. 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 local subspaces that update our approximations locally; the other uses smoothed aggregation to select low-resolution subspaces that use coarse grids to update our approximation globally. These two minimization schemes are used together in a typical multigrid way.

We recall the Rayleigh quotient to introduce a minimization principle that we use to update an iterate within a given subspace.

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

$$\rho_{A,B}(\mathbf{v}) \equiv \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T B\mathbf{v}}. \tag{7}$$

Since we restrict ourselves to the case when A and B are SPD, the RQ is always a real and positive valued. The solution we seek minimizes the RQ:

$$\rho_{A,B}(\mathbf{v}_1) = \min_{\mathbf{v}\in\mathbb{R}^n} \rho_{A,B}(\mathbf{v}) = \lambda_1 > 0. \tag{8}$$

If two vectors $\mathbf{w}$ and $\mathbf{v}$ are such that $\rho_{A,B}(\mathbf{w}) < \rho_{A,B}(\mathbf{v})$, then $\mathbf{w}$ is considered to be a better approximate solution to (6) than $\mathbf{v}$. Therefore, problem (6) is restated as a minimization problem:

$$\text{find } \mathbf{v}_1 \neq \mathbf{0} \text{ such that } \rho_{A,B}(\mathbf{v}_1) = \min_{\mathbf{v}\in\mathbb{R}^n} \rho_{A,B}(\mathbf{v}). \tag{9}$$

Given a current approximation, $\tilde{\mathbf{v}}$, we use the minimization principle to construct a subspace, $\mathcal{V} \subset \mathbb{R}^n$, such that $\dim(\mathcal{V}) = m << n$ and

$$\min_{\mathbf{v}\in\mathcal{V}} \rho_{A,B}(\mathbf{v}) \leq \rho_{A,B}(\tilde{\mathbf{v}}). \tag{10}$$

The new approximation, $\tilde{\mathbf{w}}$, is a vector in $\mathcal{V}$ with minimal RQ. Note that if $\tilde{\mathbf{v}}$ is already of minimal RQ, then lowering the RQ is not possible. In general, we must carefully construct the subspace to ensure that the RQ is indeed lowered.

To select $\tilde{\mathbf{w}}$, we must solve a restricted minimization problem within $\mathcal{V}$:

$$\text{find } \tilde{\mathbf{w}} \neq \mathbf{0} \text{ such that } \rho_{A,B}(\tilde{\mathbf{w}}) = \min_{\mathbf{v}\in\mathcal{V}} \rho_{A,B}(\mathbf{v}). \tag{11}$$

This restricted minimization problem is solved for $\tilde{\mathbf{w}}$ 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, Q, whose m column vectors are a basis for $\mathcal{V}$. Note that, for any $\mathbf{v} \in \mathcal{V}$, there exists a unique $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{v} = Q\mathbf{y}$. Moreover, the RQ of $\mathbf{v}$ with respect to A and B and the RQ of $\mathbf{y}$ with respect to coarse versions of A and B are equivalent:

$$\rho_{A,B}(\mathbf{v}) = \frac{\mathbf{v}^T A\mathbf{v}}{\mathbf{v}^T B\mathbf{v}} = \frac{\mathbf{y}^T Q^T AQ\mathbf{y}}{\mathbf{y}^T Q^T BQ\mathbf{y}} = \rho_{Q^T AQ,\, Q^T BQ}(\mathbf{y}) = \rho_{A_\mathcal{V}, B_\mathcal{V}}(\mathbf{y}), \tag{12}$$

for $A_\mathcal{V} = Q^T AQ$ and $B_\mathcal{V} = Q^T BQ$. Thus, the solution of restricted minimization problem (11) is found by solving a low-dimensional minimization problem:

$$\text{find } \mathbf{y}_1 \neq \mathbf{0} \text{ such that } \rho_{A_\mathcal{V}, B_\mathcal{V}}(\mathbf{y}_1) = \min_{\mathbf{y} \in \mathbb{R}^m} \rho_{A_\mathcal{V}, B_\mathcal{V}}(\mathbf{y}), \tag{13}$$

or, equivalently, a low-dimensional eigenproblem:

$$\begin{gathered}\text{find an eigenvector, } \mathbf{y}_1 \neq \mathbf{0}, \text{ corresponding to the smallest} \\ \text{eigenvalue, } \mu_1, \text{ in the eigenproblem } A_\mathcal{V}\mathbf{y} = \mu B_\mathcal{V}\mathbf{y}.\end{gathered} \tag{14}$$

After either approximating the solution to low-dimensional minimization problem (13) or solving low-dimensional eigenvalue problem (14) for $\mathbf{y}_1$ with a standard eigensolver, the solution to the minimization problem restricted to $\mathcal{V}$ defined in (11) is $\tilde{\mathbf{w}} \leftarrow Q\mathbf{y}_1$. The whole process is then repeated: update $\tilde{\mathbf{v}} \leftarrow \tilde{\mathbf{w}}$, use $\tilde{\mathbf{v}}$ to form a new subspace, $\mathcal{V}$, and corresponding Q, solve (14) for $\mathbf{y}_1$, and set $\tilde{\mathbf{w}} \leftarrow Q\mathbf{y}_1$.

The specific methods we use for constructing subspaces are the defining features of GES-SA and are explained in the following three sections. In section 2.1, we focus on how a reasonable initial approximation is obtained using a non-overlapping version of the subspace minimization algorithm. In section 2.2, we present the global subspace minimization based on SA that serves as our nonlinear coarse-grid update. In section 2.3, we describe the local subspace minimizations that play the role of nonlinear relaxation.

2.1. Initial Guess Development

Because the RQ minimization problem we wish to solve is nonlinear, it is helpful to develop a fairly accurate initial approximation to a minimal eigenvector. The algorithm presented in this section is very similar to the local subspace iteration that is presented later in section 2.3. The difference is that here we perform non-overlapping, additive updates with the zero-vector as an initial iterate.

First, we require that an aggregation, $\{\mathcal{A}_j\}_{j=1}^m$, be provided. Each aggregate induces a subspace, $\mathcal{V}_j \subset \mathbb{R}^n$, defined by all vectors $\mathbf{v}$ whose support is contained entirely in $\mathcal{A}_j$. We form a local selection matrix, Q_j, that maps $\mathbb{R}^{m_j}$ onto $\mathcal{V}_j$, where m_j is the number of nodes in the j-th aggregate. This matrix is given by

$$Q_j = \begin{bmatrix} \top & & \top \\ \hat{\mathbf{e}}_{p_1} & \cdots & \hat{\mathbf{e}}_{p_{m_j}} \\ \bot & & \bot \end{bmatrix}, \tag{15}$$

where $\hat{\mathbf{e}}_p$ is the p-th canonical basis vector, and $\{p_q\}_{q=1}^{m_j}$ are the nodes in the j-th aggregate. We then form local principal submatrices, $A_j \leftarrow Q_j^T AQ_j$ and $B_j \leftarrow Q_j^T BQ_j$. A solution, $\mathbf{y}_1 \neq \mathbf{0}$, to generalized eigenvalue problem (14) of size m_j is then found using a standard eigensolver.

Nodes within the j-th aggregate are set as $\tilde{\mathbf{w}}_j \leftarrow Q_j \mathbf{y}_1$. After $\tilde{\mathbf{w}}_j$ is found for each aggregate, the initial approximation is the sum of disjoint, locally supported vectors: $\tilde{\mathbf{v}} \leftarrow \sum_{j=1}^{m} \tilde{\mathbf{w}}_j$.

REMARK 2.1. There is no guarantee that $\tilde{\mathbf{w}}_j$ is of the same sign as the $\tilde{\mathbf{w}}_k$ that are supported within adjacent aggregates. For example, $\tilde{\mathbf{w}}_j$ may have all negative entries on $\mathcal{A}_j$ and $\tilde{\mathbf{w}}_k$ may have all positive entries on an adjacent aggregate. In fact, discrepancies in the sign of entries on neighboring aggregates usually occur in practice because $\beta \mathbf{y}_1$ is still a solution to the local eigenproblem, for any $\beta \neq 0$. However, this is not an issue of concern, because the subsequent coarse-grid update presented in section 2.2 uses the same aggregation as the initial guess development. The coarse space is invariant to such scaling, so the result of coarse-grid update is independent as well. In any case, we emphasize that this may only occur for the initial guess development phase of the algorithm. Example 2 in section 4 is designed to display the invariance of the success of GES-SA with respect to these sign changes.

We summarize initial guess development in the form of an algorithm. This algorithm is used on every level in the full GES-SA (algorithm 3 of section 3) as pre-relaxation for only the first multigrid cycle.

ALGORITHM 1. Initial Guess Development.

- *Function:* $\tilde{\mathbf{v}} \leftarrow IGD(A, B, \{\mathcal{A}_j\}_{j=1}^{m})$.
- *Input:* SPD matrices A and B, and aggregation $\{\mathcal{A}_j\}_{j=1}^{m}$.
- *Output:* initial approximate solution $\tilde{\mathbf{v}}$ to (6).

1. For $j = 1, ..., m$, do the following:
   (a) Form Q_j based on $\mathcal{A}_j$ as in (15).
   (b) Compute $A_j \leftarrow Q_j^T A Q_j$ and $B_j \leftarrow Q_j^T B Q_j$.
   (c) Find any $\mathbf{y}_1$, $\|\mathbf{y}_1\|_2 = 1$, by solving (14) with a standard eigensolver.
   (d) Interpolate $\tilde{\mathbf{w}}_j \leftarrow Q_j \mathbf{y}_1$.
2. Output $\tilde{\mathbf{v}} \leftarrow \sum_{j=1}^{m} \tilde{\mathbf{w}}_j$.

Algorithm 1 is demonstrated through example 1. The selection matrices are

$$Q_1 = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \\ & & \\ & & \\ & & \end{bmatrix}, \; Q_2 = \begin{bmatrix} & & \\ 1 & & \\ & 1 & \\ & & 1 \\ & & \\ & & \end{bmatrix}, \text{ and } Q_3 = \begin{bmatrix} & & \\ & & \\ & & \\ 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}. \tag{16}$$

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

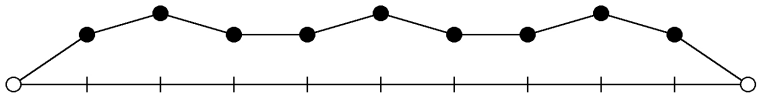
$$A_j = \frac{1}{2} \begin{bmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ & -1 & 2 \end{bmatrix} \quad \text{and} \quad B_j = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}. \tag{17}$$

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

$$\tilde{\mathbf{v}} = \left[\frac{\omega_1}{2}, \frac{\omega_1}{\sqrt{2}}, \frac{\omega_1}{2}, \frac{\omega_2}{2}, \frac{\omega_2}{\sqrt{2}}, \frac{\omega_2}{2}, \frac{\omega_3}{2}, \frac{\omega_3}{\sqrt{2}}, \frac{\omega_3}{2}\right]^T. \tag{18}$$

For the case $\omega_j = 1$ for all three aggregates, the initial guess is seen in figure 3. We reiterate what is stated in remark 2.1: if, for example $\omega_1 = \omega_3 = 1$ and $\omega_2 = -1$, then the initial guess causes no difficulty, even though the RQ of this vector is much higher than the vector formed from $\omega_1 = \omega_2 = \omega_3 = 1$. For either vector, the subsequent coarse-grid update uses the same subspace to find a set of coefficients that correspond to some new vector of minimal RQ within that subspace.

Figure 3. 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 applied as local subspace relaxation as presented in section 2.3. We now describe how smoothed aggregation is used for global subspace updates.

2.2. Global Coarse-Grid RQ Minimization

Typically, smoothed aggregation has been used to form intergrid transfer operators within multigrid schemes for linear systems, as in [4] and [17]. Here, we use smoothed aggregation in a similar fashion to form coarse subspaces of lower dimension that are used to compute iterates with lower RQ.

Smoothed aggregation defines a sparse $n \times m$ interpolation operator, P, that maps from a coarse set of m variables to the original fine set of n variables. Here, we use the same aggregation that was used for initial guess development in section 2.1. This is essential for the initial guess to be a suitable one, as stated in remark 2.1. Given a current iterate, $\tilde{\mathbf{v}}$, we form a space $\mathcal{V}$ that is designed to contain a vector with a RQ that is less than or equal to that of $\tilde{\mathbf{v}}$. Our construction is to first form tentative interpolation, $\hat{P}$, that has $\tilde{\mathbf{v}}$ in its range. This is done in the usual way by restricting the values of $\tilde{\mathbf{v}}$ to individual aggregates according to the sparsity pattern defined by the aggregation matrix, $[\mathcal{A}]$:

$$\hat{P} := \text{diag}(\tilde{\mathbf{v}})[\mathcal{A}]. \tag{19}$$

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 means that that we are guaranteed to have a vector within $\mathcal{R}(\hat{P})$ with no larger a RQ than that of $\tilde{\mathbf{v}}$:

$$\min_{\mathbf{v} \in \mathcal{R}(\hat{P})} \rho_{A,B}(\mathbf{v}) \leq \rho_{A,B}(\tilde{\mathbf{v}}). \tag{20}$$

Many of the vectors in $\mathcal{R}(\hat{P})$ are of high RQ, because the columns of $\hat{P}$ have local support and are not individually algebraically smooth with respect to relaxation. Therefore, as in standard SA, we apply a polynomial smoothing operator of low degree, S, to $\hat{P}$, and use the resulting operator, instead of $\hat{P}$, as a basis for our coarse space. This gives a coarse space with better approximation to the sought eigenvector at reasonable increase in computational complexity. This smoothing consists of just one application of the error propagation operator of damped-Jacobi:

$$S := (I_n - \alpha D^{-1} A), \tag{21}$$

where I_n is the identity operator on $\mathbb{R}^n$ and $\alpha = \frac{4}{3\|D^{-1}A\|_2}$.

Normalization of the columns of interpolation is also performed, which does not change the range of interpolation, but does control the scaling of the coarse-grid problems. This scaling is used so that the diagonal entries of coarse-grid matrix A_c are all one. The scaling is done by multiplying with diagonal matrix, N, whose entries are given by

$$N_{ii} := \frac{1}{\|S(\hat{P})_i\|_A}, \tag{22}$$

where $(\hat{P})_i$ is the i-th column of $\hat{P}$. Note that we must assume that $\tilde{\mathbf{v}}$ is nonzero on every aggregate. The interpolation matrix is

$$P := S\hat{P}N. \tag{23}$$

Under this construction, $S\tilde{\mathbf{v}}$ is in the range of P. Therefore, if $S\tilde{\mathbf{v}}$ has lower RQ than that of $\tilde{\mathbf{v}}$, we have guaranteed that a vector in $\mathcal{V}_c = \mathcal{R}(P)$ improves the RQ of our iterate. The vector of minimal RQ we select from $\mathcal{V}_c$ is typically a vector of much lower RQ than that of $S\tilde{\mathbf{v}}$ due to the localization provided by prolongation.

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 best for all other vectors in the range. Therefore, we retain the standard choice of α, known from the literature [16].

The columns of P form a basis for $\mathcal{V}_c$ 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 using damped-Jacobi smoothing does not allow columns to ever share support with an aggregate's central node. Therefore, under the assumption $\tilde{\mathbf{v}}$ is nonzero on every aggregate, $A_c \leftarrow P^T A P$ and $B_c \leftarrow P^T B P$ are both SPD. In the multigrid vocabulary, restricted problem (14) is now the *coarse-grid problem.* A coarse-grid update is given by interpolating the solution of the coarse-grid problem: $\tilde{\mathbf{w}} \leftarrow P\mathbf{y}_1$. This problem, $A_c\mathbf{y} = \mu B_c \mathbf{y}$, is either solved using a standard eigensolver or posed as a coarse-grid minimization problem as in (13), where local and global updates may be applied in a recursive fashion. This process forms the coarse-grid update step of algorithm 3 of section 3, the full GES-SA algorithm.

As in linear multigrid, the coarse-grid update needs to be complemented by an appropriately chosen relaxation process, on which we next focus.

2.3. Local Subspace RQ Relaxation

In the context of a nonlinear multilevel method, we use subspace minimization updates posed over locally supported subsets as our relaxation process, which is a form of nonlinear,

overlapping-block Gauss-Seidel method 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 update over a small cluster of m_j nodes, we construct $\mathcal{W}_j$ to be a subset of Ω_n, with cardinality m_j. Subset $\mathcal{W}_j$ should be local and connected within the graph of A. Subspace $\mathcal{V}_j^{\tilde{\mathbf{v}}}$ 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 $\mathcal{W}_j$:

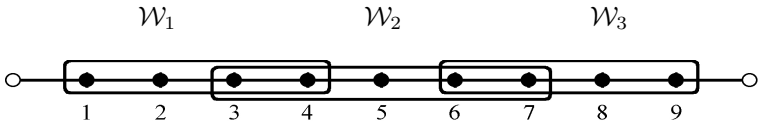
$$\mathcal{V}_j^{\tilde{\mathbf{v}}} := \{\mathbf{v} \in \mathbb{R}^n \mid \mathbf{v} = w_0\tilde{\mathbf{v}} + \mathbf{w} \text{ where } w_0 \in \mathbb{R} \text{ and } \operatorname{supp}(\mathbf{w}) \subset \mathcal{W}_j\}, \tag{24}$$

a subspace of $\mathbb{R}^n$ with dimension $(m_j + 1)$ used to form and solve (11) for an updated approximation, $\tilde{\mathbf{w}}$, that has a minimum RQ within $\mathcal{V}_j^{\tilde{\mathbf{v}}}$. We allow changes to the entries of current iterate $\tilde{\mathbf{v}}$ only at 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.

REMARK 2.2. If $\tilde{\mathbf{v}}$ has a relatively high RQ, then a vector in $\mathcal{V}_j^{\tilde{\mathbf{v}}}$ that has minimal RQ may have $w_0 = 0$. Essentially, the subspace iteration throws away all information outside of $\mathcal{W}_j$. This is potentially disastrous to our algorithm because, for typical problems, minimal eigenvectors are globally supported. Avoiding this situation is the primary reason we develop initial guesses with algorithm 1 instead of randomly. Our current implementation does not update the iterate for subspaces in which $w_0 = 0$. However this situation did not occur for the problems presented in the numerical results in section 4.

We now explain how subsets $\mathcal{W}_j$ are chosen, and then explain the iteration procedure. One step of the local subspace relaxation scheme minimizes the approximate eigenvalue locally over one small portion of the full set of nodes, Ω_n. We utilize a sequence of subsets $\{\mathcal{W}_j\}_{j=1}^m \subset \Omega_n$ that form an overlapping covering of Ω_n. We then perform local subspace relaxation with each of these subsets in a multiplicative fashion.

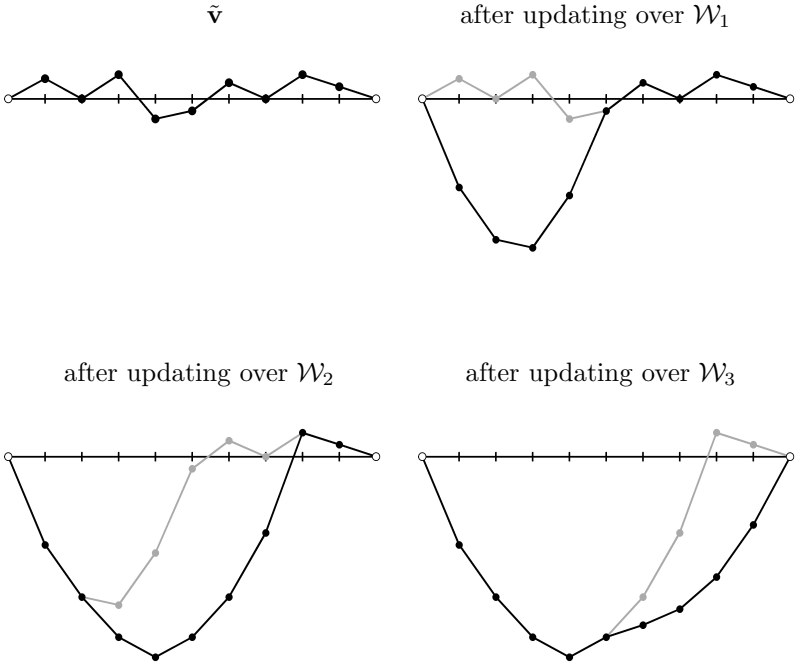
Figure 4. Graph of matrix A_G from example 1, with $n = 9$, grouped into three overlapping subsets. Each box encloses a group of nodes in a respective subset.



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 covering is by dilating aggregates. 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_G. Let $[A_G]$ be an $n \times n$ binary version of A_G that stores strong connections in the graph of A, defined as

$$[A_G]_{ij} := \left\{ \begin{array}{ll} 1, & (A_G)_{ij} \neq 0 \\ 0, & (A_G)_{ij} = 0 \end{array} \right\}. \tag{25}$$

Figure 5. A typical local subspace relaxation sweep 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 update on subset $\mathcal{W}_1$, bottom left shows a subsequent update over $\mathcal{W}_2$, and bottom right shows final relaxed iterate $\tilde{\mathbf{w}}$ after a subsequent subspace update over $\mathcal{W}_3$.



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

$$[\mathcal{W}]_{ij} := \left\{ \begin{array}{ll} 1, & \left([A_G][\mathcal{A}]\right)_{ij} \neq 0 \\ 0, & \left([A_G][\mathcal{A}]\right)_{ij} = 0 \end{array} \right\}. \tag{26}$$

Our choice of the overlapping subsets is not limited to this construction; however, we make this choice for simplicity and convenience.

In practice, each local RQ minimization is accomplished by rewriting minimization problem (11) as a generalized eigenvalue problem of low dimension, as in (14), and solving for minimal eigenvector $\mathbf{y}_1$ with a standard eigensolver. Note that here we use $Q_j^{\tilde{\mathbf{v}}}$ to represent matrices that span each subspace, $\mathcal{V}_j^{\tilde{\mathbf{v}}}$, to distinguish from the Q_j used in the initial guess section. We construct an $n \times (m_j + 1)$ matrix, $Q_j^{\tilde{\mathbf{v}}}$, so that its columns are an orthogonal basis for subspace $\mathcal{V}_j^{\tilde{\mathbf{v}}}$. To define $Q_j^{\tilde{\mathbf{v}}}$ explicitly, first define vector $\mathbf{v}_0$ by

$$(\mathbf{v}_0)_i := \left\{ \begin{array}{ll} v_i, & i \notin \mathcal{W}_j \\ 0, & i \in \mathcal{W}_j \end{array} \right\}. \tag{27}$$

For each point $p \in \mathcal{W}_j$, define canonical basis vectors $\hat{\mathbf{e}}_p$. Then, we form $Q_j^{\tilde{\mathbf{v}}}$ by appending

these $(m_j + 1)$ vectors in a matrix of column vectors:

$$Q_j^{\tilde{\mathbf{v}}} = \left[\begin{array}{c|ccc} \top & \top & & \top \\ \mathbf{v}_0 & \hat{\mathbf{e}}_{p_1} & \cdots & \hat{\mathbf{e}}_{p_{m_j}} \\ \perp & \perp & & \perp \end{array}\right], \tag{28}$$

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

$$Q_2^{\tilde{\mathbf{v}}} = \begin{bmatrix} v_1 & & & & & & \\ v_2 & & & & & & \\ 0 & 1 & & & & & \\ 0 & & 1 & & & & \\ 0 & & & 1 & & & \\ 0 & & & & 1 & & \\ 0 & & & & & 1 & \\ v_8 & & & & & & \\ v_9 & & & & & & \end{bmatrix}. \tag{29}$$

Next, we compute $A_j^{\tilde{\mathbf{v}}} \leftarrow (Q_j^{\tilde{\mathbf{v}}})^T A Q_j^{\tilde{\mathbf{v}}}$ and $B_j^{\tilde{\mathbf{v}}} \leftarrow (Q_j^{\tilde{\mathbf{v}}})^T B Q_j^{\tilde{\mathbf{v}}}$. Then (14) is solved with a standard eigensolver for $\mathbf{y}_1$, which is normalized so 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 update. Then, updated iterate is then given by $\tilde{\mathbf{w}} \leftarrow Q_j^{\tilde{\mathbf{v}}} \mathbf{y}_1$.

Local subspace relaxation is summarized in the following algorithm.

ALGORITHM 2. Local Subspace Relaxation.

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

1. For $j = 1, ..., m$, do the following:
   (a) Form $Q_j^{\tilde{\mathbf{v}}}$ based on $\tilde{\mathbf{v}}$ and $\mathcal{W}_j$ as in (28).
   (b) Form $A_j^{\tilde{\mathbf{v}}} \leftarrow (Q_j^{\tilde{\mathbf{v}}})^T A Q_j^{\tilde{\mathbf{v}}}$ and $B_j^{\tilde{\mathbf{v}}} \leftarrow (Q_j^{\tilde{\mathbf{v}}})^T B Q_j^{\tilde{\mathbf{v}}}$.
   (c) Find $\mathbf{y}_1$ by solving (14) via a standard eigensolver.
   (d) If $w_0 \neq 0$, normalize and $\tilde{\mathbf{v}} \leftarrow Q_j^{\tilde{\mathbf{v}}} \mathbf{y}_1$.
2. Output $\tilde{\mathbf{v}}$.

Figure 5 shows how a single sweep of local subspace relaxation acts on a random initial guess for the 1D example. Although the guess is never really random in the actual algorithm, due to the initial guess development, 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 iteration complements local subspace relaxation. When done in an alternating sequence, as in a standard multigrid method, the complementary processes achieve both local and global characteristics of the approximate minimal eigenvector, forming an eigensolver. Their explicit use is presented in the next section.

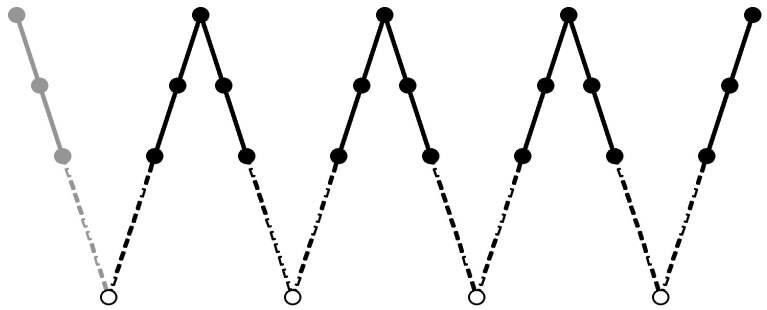
3. GES-SA

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. For example, the matrix associated with the problem on level l is denoted by A_l; in particular, $A_1 = A$, the matrix from our original problem. Interpolation from level $l+1$ to level l is denoted by by P_{l+1}^l instead of P, and restriction from level l to level $l+1$ is denoted $(P_{l+1}^l)^T$. The dimension of A_l is written n_l. Other level l objects are denoted with a subscript and superscript l, as appropriate.

3.1. The full GES-SA algorithm

GES-SA performs multilevel cycles that are structured in a format similar to standard multigrid. The very first cycle of the full GES-SA algorithm differs from the subsequent cycles: on each level, the initial guess development given in algorithm 1 is used in place of pre-relaxation. Subsequent cycles use local subspace relaxation given in algorithm 2. Coarse-grid updates are given by the process presented in section 2.2. A typical GES-SA cycling scheme is illustrated in figure 6.

Figure 6. 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 in the first cycle. 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.



ALGORITHM 3. Generalized Eigensolver Based on Smoothed Aggregation

Function: $\tilde{\mathbf{v}}_l \leftarrow GESSA(A_l, B_l, \nu, \eta, \gamma, l)$.
Input: SPD matrices A_l and B_l, number of relaxations to perform ν, number of cycles η, number of coarse-grid problem iterations γ, and current level l.
Output: approximate minimal eigenvector $\tilde{\mathbf{v}}_l$ to the level l problem.

0. If no aggregation of Ω_{n_l} is provided, compute $\{\mathcal{A}_j^l\}_{j=1}^{m_l}$. Also, if no overlapping subset covering is provided, compute $\{\mathcal{W}_j^l\}_{j=1}^{m_l}$. Step 0. is only performed once per level.

1. For $\zeta = 1, ..., \eta$, do the following:

   (a) If $\zeta = 1$, form an initial guess, $\tilde{\mathbf{v}}_l \leftarrow IGD(A_l, B_l, \{\mathcal{A}_j^l\}_{j=1}^{m_l})$. Otherwise, pre-relax the current approximation, $\tilde{\mathbf{v}}_l \leftarrow LSR(A_l, B_l, \tilde{\mathbf{v}}_l, \nu, \{\mathcal{W}_j^l\}_{j=1}^{m_l})$.
   (b) Form P_{l+1}^l with SA based on $\tilde{\mathbf{v}}_l$ and $\{\mathcal{A}_j^l\}_{j=1}^{m_l}$ as in (23).
   (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 n_{l+1} is small enough, solve (14) for $\mathbf{y}_1$ with a standard eigensolver, and set $\tilde{\mathbf{v}}_{l+1} \leftarrow \mathbf{y}_1$. Else, $\tilde{\mathbf{v}}_{l+1} \leftarrow GESSA(A_{l+1}, B_{l+1}, \nu, \gamma, \gamma, l+1)$.
   (e) Interpolate the coarse-grid minimization, $\tilde{\mathbf{v}}_l \leftarrow P_{l+1}^l \tilde{\mathbf{v}}_{l+1}$.
   (f) Post-relax the current approximation, $\tilde{\mathbf{v}}_l \leftarrow LSR(A_l, B_l, \tilde{\mathbf{v}}_l, \nu, \{\mathcal{W}_j^l\}_{j=1}^{m_l})$.

2. Output $\tilde{\mathbf{v}}_l$.

3.2. A Qualitative Comparison with Rayleigh-Quotient Multigrid

The GES-SA algorithm differs from RQMG [6] and algebraic versions of RQMG [8] in three main aspects. Iterations in RQMG are performed as corrections, whereas iterations in GES-SA are replacements or updates. In terms of cost, cycles of RQMG are cheaper than those of GES-SA. For one, the updates of the hierarchy that GES-SA creates are not performed with each iteration of RQMG. Also, the version of GES-SA we present here uses block relaxation, compared to the point relaxation used by the RQMG methods in the literature.

Perhaps more significant is that while the RQMG methods are supplied with a fixed hierarchy of interpolation operators, assumed to have good approximation for the minimal eigenvector, GES-SA starts with no multigrid hierarchy and creates one, changing the entries of the interpolation operators with each cycle. This is similar in spirit to running several initialization setup phase cycles of the original, relaxation-based αSA. The GES-SA multigrid hierarchy is iteratively improved to have coarsening and good approximation properties tailored for the problem at hand.

These differences suggests that GES-SA or a similar adaptive process may also be used to initialize RQMG by supplying it with an initial hierarchy. Of even more appeal is that RQMG could be used in subsequent cycles to develop several eigenvectors at once, which is currently not a feature of GES-SA. This would be a useful approach to initialize linear solvers for systems problems. This study does not quantitatively investigate the use of RQMG in the context of an adaptive process. These possible expansions of the current adaptive methodology are under consideration for our future research.

3.3. Simple Adaptive Linear Solvers

Our primary purpose is to use GES-SA to create an adaptive linear SA solver for the problem $A\mathbf{x} = \mathbf{b}$. We first consider problems that only require one near-kernel vector for a successful solver. Applications of GES-SA are repeated until the RQ improvement slows. This gives an approximate minimal eigenvector, $\tilde{\mathbf{v}}$. Then, the setup phase of SA is run to form a solver that accurately represents $\mathcal{K} = \{\tilde{\mathbf{v}}\}$. This solver is tested on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$. Section 4 presents results only for such one-vector solvers.

However, if the current, one-vector solver is not adequate, then we must develop a vector that represents error that is algebraically smooth with respect to this solver. Currently, our approach is to use the general setup phase of αSA in [4] to develop a secondary component, $\mathbf{k}_2$.

(A study regarding RQ optimization approaches for computing these secondary components is underway). Then, the setup phase of SA is run to form a solver that accurately represents $\mathcal{K} = \{\tilde{\mathbf{v}}, \mathbf{k}_2\}$. The updated solver is again tested on the homogeneous problem. If the updated solver is also inadequate, the αSA process can be repeated until an adequate solver is built.

4. Numerical Results

Many linear systems that come from the discretization of scalar PDEs are solved efficiently with SA, with the vector of all ones as near-kernel, 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 may not produce a linear solver with acceptable convergence rates.

All the results in this section display the result of running one GES-SA V-cycle ($\eta = 1, \gamma = 1$) and $\nu = 2$ post-relaxation steps. Our implementation for GES-SA is currently in `MATLAB` and we therefore make no rigorous timing comparisons with competing eigensolvers. In further investigations, we intend to explore these details. The small eigenproblems involved in GES-SA were all solved using the `eigs()` function with flags set for real and symmetric matrices, which implements `ARPACK` [10] routines. No 2D problem used more than 5 iterations to solve small eigenproblems; no 3D problem used more than 10 iterations.

Table I. Asymptotic convergence factors for the 2- and 3-dimensional finite difference (FD) and finite element (FE) versions of the random-signed Laplacian problem. Factors in the column labeled "ones" correspond to solvers created using the vectors of all ones; factors in the "ges-sa" column correspond to solvers that use our approximate minimal eigenvector computed with GES-SA; and factors in the "eigen" column correspond to solvers that use the actual minimal eigenvector. The last column, "comp", displays the operator complexity for all 3 types of solvers.

	prob. size	levels	ones	ges-sa	eigen	comp
2D, FE	81	2	0.620	0.074	0.074	1.078
	729	3	0.892	0.176	0.179	1.108
	6561	4	0.965	0.193	0.196	1.119
	59049	5	0.977	0.215	0.214	1.123
2D, FD	81	2	0.849	0.219	0.219	1.317
	729	3	0.947	0.294	0.290	1.357
	6561	4	0.962	0.306	0.305	1.348
	59049	5	0.978	0.312	0.312	1.342
3D, FE	729	2	0.598	0.114	0.111	1.054
	19683	3	0.934	0.188	0.189	1.112
3D, FD	729	2	0.825	0.289	0.292	1.389
	19683	3	0.944	0.360	0.358	1.495
	64000	4	0.961	0.418	0.413	1.511

EXAMPLE 2. We present the *random-signed discrete Laplacian.* Consider the d-dimensional

Poisson problem with Dirichlet boundary conditions

$$\begin{array}{rcll} -\Delta u & = & f & \text{in} \quad \Omega = (0,1)^d \\ u & = & 0 & \text{on} \quad \delta\Omega. \end{array} \tag{30}$$

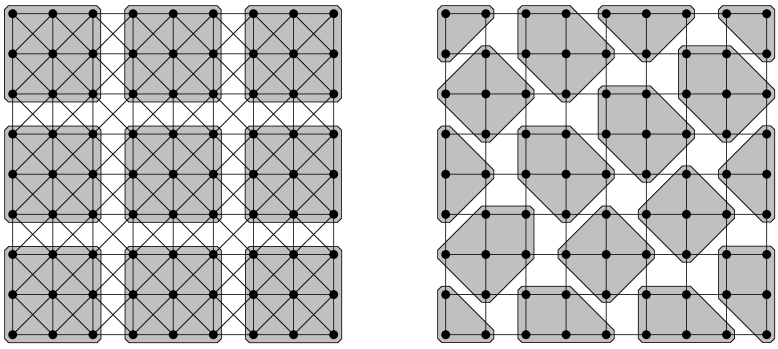
We discretize (30) with both finite element spaces with nodal bases, and second-order finite differences, on equidistant rectilinear grids.

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

$$A \leftarrow D_{\pm}\hat{A}D_{\pm}. \tag{31}$$

In our results, we also symmetrically scale the matrix A to have ones on its diagonal for example 2.

Figure 7. Aggregation examples displayed for 2D test problems of low dimension. On the left is an aggregation formed with a geometric aggregation method used for the finite element problems; on the right is an aggregation formed with an algebraic aggregation method used for finite-difference problems. Black edges represent strong connections within graph of matrix A_G; each gray box represents a separate aggregate that contains the nodes enclosed.



geometric aggregation algebraic aggregation

Now consider solving $A\mathbf{x} = \mathbf{b}$ given vector $\mathbf{b}$. Note that the vector of all ones is not algebraically smooth with respect to standard relaxation methods. As seen in table I, using the vector of all ones produces SA solvers that have unacceptable convergence factors for these problems. Instead, we use one GES-SA cycle to produce an approximate minimal eigenvector, $\tilde{\mathbf{v}}$, and use $\mathcal{K} = \{\tilde{\mathbf{v}}\}$ 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 linear SA $V(2,2)$-cycles,

$$\text{Asymptotic Convergence Factor} \approx \left(\frac{\|\mathbf{e}^{(25)}\|_A}{\|\mathbf{e}^{(20)}\|_A} \right)^{1/5} \tag{32}$$

Table II. Relative errors between the RQ of the GES-SA approximate minimal eigenvector, ρ, and the minimal eigenvalue, λ_1, for 2- and 3-dimensional finite element and finite difference versions of example 2.

	prob. size	levels	ρ	λ_1	rel. error
2D, FE	81	2	7.222e-02	7.222e-02	0.0000034
	729	3	9.413e-03	9.412e-03	0.0001608
	6561	4	1.101e-03	1.100e-03	0.0002491
	59049	5	1.243e-04	1.243e-04	0.0001224
2D, FD	81	2	4.895e-02	4.894e-02	0.0000582
	729	3	6.307e-03	6.288e-03	0.0031257
	6561	4	7.501e-04	7.338e-04	0.0222547
	59049	5	9.306e-05	8.289e-05	0.1227465
3D, FE	729	2	1.066e-01	1.066e-01	0.0000017
	19683	3	1.412e-02	1.409e-02	0.0022805
3D, FD	729	2	4.896e-02	4.894e-02	0.0003230
	19683	3	6.303e-03	6.288e-03	0.0024756
	64000	4	2.981e-03	2.934e-03	0.0158771

for the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, starting with a random initial guess. Operator complexity is also reported for the linear solver that uses the vector developed with GES-SA. We use the usual definition of *operator complexity*,

$$\text{comp} = \frac{\sum_{l=1}^{L} \text{nz}(A_l)}{\text{nz}(A_1)}, \tag{33}$$

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

Both geometric and algebraic-based aggregation was done in our tests. For the finite element problems, we took advantage of knowing the geometry of the grid and formed aggregates that were blocks of 3^d nodes. For the finite difference problems, no geometric information was employed and aggregation was done algebraically, as in [17]. Small examples in 2 dimensions of the difference between the two types of aggregations we used are shown in figure 7. Algebraic aggregations were done based on the strength-of-connection measure given in (3), with $\theta = .1$.

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 Laplacians, table II displays 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 a decent approximate minimal eigenvector can be produced with one GES-SA cycle for general problems. Note that 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.

EXAMPLE 3. We also investigate GES-SA on *"shifted" Laplacian*, or Hemholtz, problems to display the invariance of performance with respect to such shifts. Consider the d-dimensional

Poisson problem with Dirichlet boundary conditions, shifted by a parameter, $\sigma_s > 0$:

$$\begin{array}{rcll} -\Delta u - \sigma_s u & = & f & \text{in} \quad \Omega = (0,1)^d \\ u & = & 0 & \text{on} \quad \delta\Omega. \end{array} \tag{34}$$

Here, σ_s is chosen to make the continuous problem nearly singular. The minimal eigenvalue of the Laplacian operator on $(0,1)^d$ is $d\pi^2$. Therefore setting

$$\sigma_s = (1 - 10^{-s})d\pi^2, \tag{35}$$

for an integer $s > 0$, makes the shifted operator $(-\Delta - \sigma_s)$ have a minimal eigenvalue of $\mu_1 = 10^{-s}d\pi^2$. Here, we consider the $d = 2$ and $d = 3$ cases, for various shifts σ_s. We discretized the 2D case with nodal bilinear functions on square elements, with $h = \frac{1}{244}$. This gave us a system with $n = 59,049$ degrees of freedom. All aggregation done in these tests was geometric, and aggregate diameters were never greater than 3. For each shift, the solvers we developed (using both GES-SA and the actual minimal eigenvector) have operator complexity 1.119 and 5 levels with 59,049, 6561, 729, 81, and 9 degrees of freedom on each respective level. Similarly, the 3D case was discretized with nodal trilinear functions on cube elements, with $h = \frac{1}{37}$. This gave us a system with $n = 46,656$ degrees of freedom. Again, for each shift the solvers have operator complexity 1.033 and 4 levels with 46,656, 1,728, 64, and 8 degrees of freedom on each respective level. In either case, the minimal eigenvalue for the discretized matrix A is $\lambda_1 \approx 10^{-s}d\pi^2h^d$.

For all cases, we produced 2 SA solvers: the first solver was based on the actual minimal eigenvector of A and the second was based on the approximation to the minimal eigenvector created by one cycle of GES-SA. In table III, we display asymptotic convergence factors (32) for these solvers for 2D and 3D and specific shift parameters.

We assume that prolongation P from the first coarse grid to the fine grid satisfies the weak approximation property, with constant

$$C := \sup_{\mathbf{u}\in\mathbb{R}^{n_f}} \left(\min_{\mathbf{v}\in\mathbb{R}^{n_c}} \frac{\|\mathbf{u} - P\mathbf{v}\|_2^2 \|A\|_2}{(A\mathbf{u},\mathbf{u})} \right). \tag{36}$$

Based on the knowledge that A comes from a scalar PDE, we further assume that it is most essential to approximate a minimal eigenvector, $\mathbf{u}_1$. The denominator, $(A\mathbf{u},\mathbf{u})$, is smallest for this vector and other vectors that have comparable denominators are locally well-represented by $\mathbf{u}_1$. Under these assumptions, we feel it is insightful to monitor the following *measure of approximation* for any P that we develop.

$$M_1(P) := \min_{\mathbf{v}\in\mathbb{R}^{n_c}} \frac{\|\mathbf{u}_1 - P\mathbf{v}\|_2^2 \|A\|_2}{(A\mathbf{u}_1,\mathbf{u}_1)}, \tag{37}$$

where $\mathbf{u}_1$ is the minimal eigenvector of A. Note that this is a lower bound: $M_1(P) \leq C$. We compute $\min_{\mathbf{v}\in\mathbb{R}^{n_c}} \|\mathbf{u}_1 - P\mathbf{v}\|_2$ by directly projecting $\mathbf{u}_1$ onto the range of P, a computationally costly operation that is merely a tool for analyzing test problems. Table III reports $M_1(P)$ on the finest grid, for the P developed using the GES-SA method. As σ_s increases, and the problem becomes more ill-conditioned, we see an increase of $M_1(P)$ and eventually a degradation in the convergence factors for the 2D linear solvers that GES-SA produced.

We wish to investigate whether the degradation in the 2D GES-SA solver is due to GES-SA performing worse for the more ill-conditioned problems, or the approximation requirements

getting stricter. To this purpose, we monitor a second measure of approximation

$$M_2(P) := \min_{\mathbf{v}\in\mathbb{R}^{n_c}} \frac{\|\mathbf{u}_1 - P\mathbf{v}\|_2^2}{\|\mathbf{u}_1\|_2^2}. \tag{38}$$

Again, this measure is displayed in table III for each problem. As σ_s increases, we see that $M_2(P)$ is essentially constant for the linear solvers that GES-SA produced, with fixed computation, indicating that the degradation is only due to the approximation requirements getting stricter.

Table III. Asymptotic convergence factors and measures of approximation for example 3. The s values in the columns give shift sizes σ_s as in (35). The first block row is for 2D problems, the second is for 3D problems. The rows labeled "λ_1" display the minimal eigenvalue for the specific discrete problem and those labeled "ρ" display RQs of the GES-SA vectors. Rows labeled "eigen" display convergence factors for solvers based on the actual minimum eigenvector. Rows labeled "ges-sa" display convergence factors for solvers based on the approximation to the minimal eigenvector given by one GES-SA cycle. Measures of approximation, $M_1(P)$ and $M_2(P)$, are in rows with respective labels.

		$s=1$	$s=2$	$s=3$	$s=4$	$s=5$
	λ_1	3.32e-05	3.32e-06	3.36e-07	3.77e-08	7.90e-09
	ρ	3.32e-05	3.37e-06	3.88e-07	9.11e-08	6.03e-08
2D, FE	eigen	.196	.198	.198	.199	.197
	ges-sa	.197	.197	.196	.199	**.430**
(n = 59,049)	$M_1(P)$	1.14e-05	1.13e-04	1.11e-03	1.01e-02	4.83e-02
	$M_2(P)$	9.45e-11	9.37e-11	9.36e-11	9.54e-11	9.54e-11
	λ_1	5.86e-05	6.17e-06	9.32e-07	4.08e-07	3.56e-07
	ρ	5.88e-05	6.30e-06	1.06e-06	5.40e-07	4.86e-07
3D, FE	eigen	.187	.187	.190	.188	.183
	ges-sa	.188	.185	.188	.187	.185
(n=46,656)	$M_1(P)$	7.07e-05	6.67e-04	4.43e-03	1.04e-02	1.18e-02
	$M_2(P)$	3.85e-08	3.83e-08	3.84e-08	3.94e-08	3.91e-08

5. Conclusion

This paper develops a multilevel eigensolver, GES-SA, in the SA framework for the specific application of enhancing robustness of current adaptive linear SA solvers. We show preliminary numerical results that support approximate eigensolvers as potentially useful for initialization within the adaptive algebraic multigrid process. This paper serves as a proof of concept, and due to our high-level implementation, we are not making claims about the efficiency of this algorithm versus purely relaxation-based initialization given in [4]. This question will be investigated as we begin incorporating eigensolvers into our low-level adaptive software.

REFERENCES

1. A. Brandt. Algebraic multigrid theory: The symmetric case. *Appl. Math. Comput.*, 9:23–26, 1986.
2. A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. *DJ Evans (Ed.), Sparsity and its Applications*, 1984.
3. M. Brezina. *Robust iterative methods on unstructured meshes*. PhD thesis, University of Colorado, Denver, Colorado, 1997.
4. M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. Adaptive Smoothed Aggregation (αSA). *SISC*, 25:1896–1920, 2004.
5. W. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial, 2^{nd} Edition*. SIAM books, 2000.
6. Z. Cai, J. Mandel, and S. F. McCormick. Multigrid methods for nearly singular linear equations and eigenvalue problems. *SIAM J. Numer. Anal.*, 34:178–200, 1997.
7. T. F. Chan and I. Sharapov. Subspace correction multi-level methods for elliptic eigenvalue problems. *Numerical Linear Algebra with Applications*, 9:1 20, 2002.
8. U. Hetmaniuk. A Rayleigh quotient minimization algorithm based on algebraic multigrid. *Numerical Linear Algebra with Applications*, 14:563–580, 2007.
9. U. Hetmaniuk and R. B. Lehoucq. Multilevel methods for eigenspace computations in structural dynamics. *Domain Decomposition Methods in Science and Engineering, Lecture notes in Computational Science and Engineering*, 55:103–114, 2007.
10. R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1998.
11. S. F. McCormick and J. Ruge. Multigrid Methods for Variational Problems. *SIAM J. Numer. Anal.*, 19:925–929, 1982.
12. K. Neymeyr. Solving mesh eigenproblems with multigrid efficiency. *Numerical Methods for Scientific Computing, Variational Problems and Applications, editors Y. Kuznetsoz, P. Neittaanmäki, and O. Pironneau*, 2003.
13. J. Ruge. *Multigrid methods for variational and differential eigenvalue problems and unigrid for multigrid simulation*. PhD thesis, Colorado State University, Fort Collins, Colorado, 1981.
14. J. Ruge and K. Stüben. Algebraic Multigrid (AMG). *Multigrid Methods (McComrick, S.F., ed.)*, 5, 1986.
15. U. Trottenberg, C. W. Osterlee, and A. Schuller (Appendix by K. Stuben). *Multigrid (Appendix A: An Introduction to Algebraic Multigrid)*. Academic Press, 2000.
16. P. Vaněk, M. Brezina, and J. Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88:559–579, 2001.
17. P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56:179–196, 1996.