Efficient Multilevel Eigensolvers with Applications to Data Analysis Tasks

Dan Kushnir, Meirav Galun, and Achi Brandt

Abstract—Multigrid solvers proved very efficient for solving massive systems of equations in various fields. These solvers are based on iterative relaxation schemes together with the approximation of the "smooth" error function on a coarser level (grid). We present two efficient multilevel eigensolvers for solving massive eigenvalue problems that emerge in data analysis tasks. The first solver, a version of classical algebraic multigrid (AMG), is applied to eigenproblems arising in clustering, image segmentation, and dimensionality reduction, demonstrating an order of magnitude speedup compared to the popular Lanczos algorithm. The second solver is based on a new, much more accurate interpolation scheme. It enables calculating a large number of eigenvectors very inexpensively.

Index Terms—Eigenvalues and eigenvectors, multigrid and multilevel methods, graph algorithms, segmentation, clustering.

1 Introduction

The spectral decomposition of matrices is used in various tasks of data analysis, such as data clustering, image segmentation, and dimensionality reduction. Spectral graph methods are based on the first eigenvalues and their corresponding eigenvectors of an $N \times N$ matrix derived from the pairwise affinity matrix, where N is the number of data points. Although promising results have been shown, the computational complexity of spectral graph methods is limited by the computation of the related eigenvalue problem. Standard eigensolvers (e.g., QR [3]) comprise cubic complexity in the size of the data set, whereas the running time of other iterative methods for sparse matrices (e.g., Lanczos et al. [42]) is significantly affected by the size, sparsity measure, and spectral properties of the problem [3]. The computational bottleneck of computing the eigenvectors has been a motivation for several related multilevel approaches, e.g., [1], [2], [11], [12], [18], [35]. We specifically note approaches where eigenvectors are explicitly computed as, for example, in the spectral graph methods suggested in [11], [12], [18]. In [11], the eigenvalue problem is efficiently solved for a smaller coarse matrix obtained by sampling the data set. The Nyström extension is used to interpolate the solution calculated for the sampled coarse set to the remaining data points. The accuracy of interpolation suggested in [11] relies on how well the random sample represents the remaining data points. We also note that the approximation suggested in [11] corresponds to the eigendecomposition of full matrices. Hence, for very large matrices, it is hard to evaluate the numeric accuracy of the solution since the full matrix cannot be computed. In the context of image segmentation, Cour et al. [12] suggest a particular sampling method of the corresponding full $N \times N$ pixels affinity matrix to obtain a larger yet sparser matrix whose eigenvectors are computed faster by the Lanczos eigensolver, thus transforming the original eigenproblem to a different one. The speedup is obtained by exploiting the special structure of the sparse matrix within the Lanczos solver. The sampling method of the affinity matrix suggested by [12] can be fairly efficient in the case of image segmentation, i.e., where the pixels proximities can be easily derived from the image grid. However, for scattered high-dimensional data, such an approach will actually require the computation of the full $N \times N$ matrix, a costly operation that should be avoided in any case. In our multilevel eigensolvers for sparse matrices, the interpolation is constructed differently. An important difference is that the coarse set is chosen such that the remaining data points are strongly connected to the chosen coarse set, thus suggesting a more accurate coarse-to-fine interpolation than the one suggested in [11]. Also, the corresponding matrix is a priori sparse and corresponds to local connections in the data graph that can be obtained by using efficient methods (e.g., [43]). We also compare our solver performance with the Lanczos eigensolver, demonstrating an order of magnitude speedup. We comment on [18], [35], and other works below.

Our work relies on the multigrid solver for differential eigenproblems on geometric grids presented in [19], and on algebraic multigrid (AMG) techniques [4], [6] for solving unstructured large-scale linear systems of equations. Over the past few decades, several multigrid schemes were developed for *differential* eigenvalue problems [20], [21], [22], [23], [24], [27], [28]. We note the works of [21] and [27] that rely on the Full Approximation Scheme (see below). The multigrid solver of [21] is suggested for the Schrödinger equation in 2D and 3D with special treatment for closely clustered eigenvalues. Borzi and Borzi [27] use the AMG technique for solving unstructured differential eigenproblems in 2D and 3D for the first few eigenvectors. Three

- *D. Kushnir is with the Department of Mathematics, Yale University, 51 Prospect St., New Haven, CT 06520-8285. E-mail: dan.kushnir@yale.edu.*
- *M. Galun and A. Brandt are with the Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. E-mail: {meirav.galun, achi.brandt}@weizmann.ac.il.*

Manuscript received 6 Aug. 2008; revised 21 Nov. 2008; accepted 20 June 2009; published online 24 July 2009.
Recommended for acceptance by K. Murphy.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-08-0468.
Digital Object Identifier no. 10.1109/TPAMI.2009.147.

different algorithms based on classical AMG are suggested in [27]. Most related to our work is the first algorithm, based on the FAS with AMG (FAMG), and also employing a Full Multigrid cycle (FMG, see [9]) for initialization. FAMG does not employ the Kaczmarz relaxation method, and therefore, relaxation diverges at coarse levels when the iteration matrix is indefinite (see Section 3.2). Such coarse levels can be avoided, but avoiding these levels causes the solver convergence to be much slower. Other approaches tackling indefiniteness are suggested in several multigrid preconditioners for Helmholtz equation, such as [25], [26]. In the context of differential eigenproblems, we also note [23], which suggests an accurate algorithm for computing two eigenfunctions of the 1D Helmholtz equation by using a Full Approximation Scheme solver. The solver of [23] can solve for any two eigenfunctions (and corresponding eigenvalues) in the vicinity of a given eigenvalue $\tilde{\lambda}$, not necessarily for the lowest eigenvalues of the Helmholtz operator. In [28], the author suggests an AMG extension for Rayleigh Quotient Multigrid (RQMG) solver [29] for computing several of the first eigenvectors of a differential Laplacian. The AMG version of RQMG employs Davidson iteration with Rayleigh-Ritz projection to obtain update directions for the approximation of the *few* smallest minimizing eigenpairs of the Rayleigh Quotient. The minimization is done in each level of the multilevel solver. Another related class of multigrid solvers relies on the smooth aggregation scheme suggested in [31], [32], and in particular, the AMG eigensolvers of [30] and [35]. The generalized eigensolver in [30] minimizes the Rayleigh Quotient (RQ) for a *single* minimal eigenvector but, unlike [28], which uses a static intergrid transfer operator, it initializes and modifies the intergrid operator in each cycle by scaling the aggregation matrix with the current approximation and smoothing the operator to obtain an operator which has the current approximation in its range and spans a coarse subspace of lower RQ vectors.

In the context of stochastic matrices, we mention the algorithms of [18] and [35]. The former is based on interpolations similar to the classical AMG (had the latter been applied to a Laplacian of a graph associated with the stochastic matrix). This, however, is done just for obtaining a first approximation, without using multilevel acceleration cycles; hence, the obtained eigensolver is much less efficient. The multilevel eigensolver of [35] for stochastic matrices is closely related to previous aggregation algorithms for stochastic matrices [36], [37], [38], [39], [40], all designed for the computation of the (single) first eigenvector corresponding to the stationary distribution of a stochastic matrix. Sterck et al. [35] suggest an adaptive smoothed aggregation method [32], [33] for this particular case. Specifically, the solver uses an adaptive AMG multilevel solver involving relaxation and multiplicative corrections, in which the matrix is scaled with the current iterate so that the chosen coarse seeds correspond to the Markov chain states of highest probability as encoded in the current iterate. The solver is used for Web page ranking applications.

Unlike [23], [27], [28], [31], [32], and [35], the presentation in our paper is in the context of data analysis problems. Also, we demonstrate a particularly fast multilevel approach for simultaneously calculating *many* eigenvectors.

Multigrid solvers [5] for linear problems (structured or unstructured) are based on iterating between two processes:

1. Classical relaxation schemes that are generally slow to converge but fast to "*smooth*" the error function, i.e., approximately eliminate from it *high eigencomponents* (eigenvectors whose eigenvalue is comparable to the largest absolute eigenvalue).
2. Approximating the "smooth" error function (made primarily of low eigencomponents) on a coarser level (a coarser grid, or more generally, a smaller graph, typically having one quarter to one half the number of nodes). This is done by solving coarse-level equations derived from the fine-level system and the residuals of its current approximate solution, and then interpolating that coarse-level solution to correct the fine-level approximation. The solution of the coarse level equations is obtained by recursively using the same two processes, employing still coarser levels.

As a result, large-scale changes are effectively calculated on correspondingly coarse levels, based on information gathered from finer levels. Such multigrid solvers typically yield linear complexity, i.e., the total solution work is proportional to the number of variables (nodes) in the system.

The main issue in AMG solvers is how to choose the coarse-level variables and how to derive a sufficiently good interpolation from the coarse level to the next finer level (see [4] and [5, Appendix A]). Once such an interpolation is known, the coarse-level equations can be derived by the classical Galerkin method (see below).

For simplicity, in this paper, we always assume that the set of variables at each coarse level is a subset of the next-finer-level set, chosen by a particular selection algorithm. The so-called *"classical" AMG interpolation* uses interpolation weights proportional to the weights of the graph edges. This yields efficient solvers for typical graph problems, i.e., to systems of equations based on the Laplacian of a graph, provided the graph is mostly "local", i.e., connected nodes are generally close to each other in some low-dimensional space in which the graph could be embedded.

A more accurate and general interpolation scheme has been developed more recently in the framework of Bootstrap AMG (BAMG, see [7] Section 17.2, and [9] Section 4). The BAMG interpolation is defined as the interpolation that fits best (in some least-squares sense) a set of relaxed error vectors, each being produced by relaxing the homogeneous system of equations starting from a random approximation. With this improved interpolation, many more general linear systems (not relying at all on having a matrix with dominant main diagonal and not requiring graph locality) can be solved fast by AMG. And, as explained below, this interpolation is particularly useful when one needs to calculate *many* eigenvectors. Brezina et al. [34] also suggest a similar relaxation process to approximate error components that are slow to converge. The slow to converge error vectors are then used to improve the interpolation, and thus, the whole algebraic multigrid process.

In the course of this paper, we present two types of multilevel eigensolvers. The first solver is based on the Full Approximation Scheme (FAS) [9], in which the coarse levels are used to calculate approximations to the "smooth" error

function. This solver can use either the classical AMG interpolation or the more accurate BAMG interpolation. For most eigenproblems emerging in data analysis, the classical AMG solver has already turned out to be very efficient, as demonstrated in Section 4 below on several representative examples of clustering [13], image segmentation [14], and dimensionality reduction [17]. Specifically, we show that the running time of this (nonoptimized Matlab and C) AMG multilevel eigensolver is an order of magnitude faster than the Lanczos algorithm implemented in the ARPACK package [42].

The second type of multilevel solver that we have adapted for graph eigenproblems employs the Exact Interpolation Scheme (EIS, first introduced in [9]). In that scheme, at each fine level of each multilevel cycle, one calculates an interpolation operator that exactly fits the current (after relaxation) approximate solution; here, of course, the BAMG-interpolation must be used. The next coarser level is then used to calculate a *solution* (not a correction, as in FAS) that, upon interpolation to the fine level, yields an improved approximation there.

An obvious disadvantage of the EIS is the need to recalculate the interpolations. This disadvantage is rather mild in *algebraic* multigrid solvers, which should calculate good interpolations for the (generally unstructured) system anyway, and in many cases, should keep improving the interpolations to fit nearly singular components, as suggested, for example, in [7], [32], [33], [34]. More important, the EIS is the scheme of choice for calculating *many* eigenvectors because, based on just several briefly relaxed vectors, it is possible to construct *one* BAMG interpolation that accurately enough fits them all (see Section 3.3). As a result, *one* coarse-level eigenproblem is obtained whose solution gives good approximation to all of those eigenvectors. For many types of applications, one need not even interpolate all of those eigenvectors, obtained at some coarse level, back to the finer level(s). In fact, it is often preferable to use their coarse-level versions, as explained by an example in Section 3.3. Thus, to calculate many eigenvectors, it is often enough to derive one BAMG interpolation, a purely local operation, with the rest of the calculations being very inexpensive since they no longer use the finest level (or even *several* finest levels).

This process for solving many-eigenvector problems is part of a more general process that extends the idea to higher eigenvectors and to coarser grids by using more than one interpolation per level. It may lead to the calculation of N eigenvectors in just $O(NlogN)$ computer operations and $O(NlogN)$ computer storage, as shown for discretized one-dimensional differential operators in [24] (and previously by a different method in [22]).

In the context of our work, we show an application of the EIS solver for computing many eigenvectors of Laplacian eigenproblems corresponding to a graph whose data points comprise a 2D grid and whose edges might have random weights. Then we demonstrate an application in multiclass spectral clustering of a Gaussian mixture model [41] (see Section 4.2). We demonstrate that the EIS solver with BAMG interpolation can have higher efficiency and accuracy than the FAS solver with classical AMG interpolation. Moreover, we demonstrate that a specific low eigenvector can also be computed efficiently and to a high accuracy by using the EIS solver.

2 Spectral Methods for Data Analysis Problems

Let W denote the pairwise affinity matrix of a graph $G(V, W)$ with nodes V representing the data points and edges whose positive weights are the pairwise affinities. The graph Laplacian is defined as $L = D - W$, where D is the diagonal matrix with entries $d_{ii} = \sum_j w_{ij}$. In the spirit of spectral theory [16], a solution for a related data partition problem can be constructed from the solution of the following generalized eigenvalue problem:

$$LU = \lambda DU, \tag{1}$$

or a closely related eigenvalue problem

$$D^{-1/2}LD^{-1/2}Z = (I - D^{-1/2}WD^{-1/2})Z = \lambda Z. \tag{2}$$

The first low eigenvectors of (2) induce an embedding of the data points into a low-dimensional subspace such that their values can then be used to define the partition of the data into coherent groups, generally by using a simple clustering technique (such as k-means [45]) [13], [14]. A similar method is used in dimensionality reduction problems, e.g., [17], where the first eigenvectors are used to embed the data into a lower dimensional subspace.

3 The Eigensolver Algorithms

3.1 The Problem

The eignsolvers suggested in this paper are designed to numerically compute the first d eigenvectors of the generalized eigenvalue problem

$$AU = BU\Lambda, \tag{3}$$

where A and B are $N \times N$ symmetric positive semidefinite matrices and Λ is a diagonal matrix of the corresponding eigenvalues. That is, we are looking for the smallest real numbers $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_d$ and the vectors $U_1, U_2, \ldots, U_d$ so that

$$AU_i = \lambda_i BU_i \quad i = 1, \ldots, d \tag{4}$$

and

$$(U_i, BU_j) = \delta_{ij} \quad i, j = 1, \ldots, d. \tag{5}$$

3.2 The Full Approximation Scheme Solver

The algorithm starts with an initialization step which provides an initial approximation to the first d eigenvectors and their corresponding eigenvalues. Then, the eigenvalues are fixed, and the eigenvector approximation is updated independently for each single eigenvalue. For that matter, we employ a multilevel scheme which utilizes a common interpolation throughout the approximation of *all* the first eigenvectors. Once approximations of the first d eigenvectors are obtained by the multilevel scheme, the subspace X spanned by them is a good approximation to the subspace spanned by the desired eigenvectors. Henceforth, the multilevel procedure is followed by a Ritz projection

procedure which determines a basis for X that is closest to the eigenvectors in some sense, along with an update of the eigenvalues. Generally, a *few* iterations of the multilevel procedure followed by the Ritz projection are performed until a predefined accuracy is achieved. Next, we describe the major part of the algorithm, i.e., the multilevel scheme. For simplicity, we focus on a *two-level* scheme. The multilevel scheme is a quite straightforward generalization of the two-level scheme.

Starting at the fine level on which the original eigenvalue problem is considered for a *single* eigenvalue,

$$(A - \lambda B)u = \tau, \tag{6}$$

with $\tau = 0$, a few *relaxation* sweeps are first performed to obtain a certain approximation for u. Then, a *coarse problem* is constructed on a coarser level, employing only a subset of the original variables. Finally, after approximating the coarse level solution, a *coarse level correction* is transferred to the fine level, on which a few additional relaxation sweeps are performed. These basic ingredients of the multilevel scheme: relaxation, coarse problem construction, and coarse level correction are explained in detail below.

Relaxation. Given a system of equations of the form $(A - \lambda B)u = \tau$, and certain initial approximations for λ and u, a Gauss-Seidel (GS) relaxation sweep is defined as follows: Keeping λ fixed, the variables u_i are scanned in some order, updating each to satisfy the corresponding equation. Namely, denoting $A - \lambda B$ by M and the approximation before starting the sweep by u^{old}, we set

$$u_i^{new} = m_{ii}^{-1}\left(\tau_i - \sum_{j<i} m_{ij}u_j^{new} - \sum_{j>i} m_{ij}u_j^{old}\right). \tag{7}$$

The sequence of relaxation sweeps converges if (and generally only if) the symmetric matrix M is positive definite (PD). When M is only approximately PD, i.e., M contains a relatively small number of negative eigenvalues (as happens on a fine level when A is PD and λ is one of the first few eigenvalues), the GS iterations slowly diverge. In the multilevel algorithms, this divergence is not a problem provided the slowly diverging components are well approximated at the coarser levels, and thus, converge by the coarse-to-fine corrections. This, however, no longer works at some of the coarsest levels since there the divergence of those smooth components is much faster. Avoiding these coarsest levels would, on the other hand, result in slow convergence of the multilevel algorithm. Thus, we choose a more efficient approach by switching to *Kaczmarz relaxation* [10] at some of the coarsest levels. In the Kaczmarz scheme, the equations $i = 1, ..., N$ are scanned, simultaneously updating for the ith equation all of the variables participating in it by setting

$$u^{new} = u^{old} + \frac{\tau_i - m_i u^{old}}{m_i m_i^T} m_i^T, \tag{8}$$

where $m_i = (m_{i1}, \ldots, m_{iN})$. After each such update induced by the ith equation, u^{new} becomes u^{old}. Kaczmarz relaxation can be considered as Gauss-Seidel relaxation applied to $MM^T y = \tau$ with $M^T y = u$.

Coarse problem construction. Here we explain the construction of the coarse level equations. First, the coarse level variables are chosen. Then, the coarse-to-fine interpolation is computed, and finally, the coarse level equations are derived. This construction follows the algebraic multigrid (AMG) coarsening scheme explained in [6] and in [5, Appendix A].

3.2.1 Selection of Coarse Variables

The construction of the set of coarse variables C and its complement denoted by F is guided by the principle that each F variable should be "strongly connected" to C, i.e., each $i \in F$ should satisfy $\sum_{j\in C}|a_{ij}| \geq \alpha \sum_j |a_{ij}|$, where α is a parameter, typically $\alpha = 0.2$. To achieve this objective one starts with an empty set C and sequentially transfers variables from F to C until all of the remaining $i \in F$ satisfy the mentioned relation.

3.2.2 Interpolation

For each variable $i \in F$, a coarse set $N_i \subseteq \{j \in C, a_{ij} < 0\}$ is defined. The size of N_i is called the interpolation caliber. Let $I(j)$ be the coarse set index of the variable whose index at the fine level is j. The classical AMG interpolation matrix P (of size $N \times n$, where $n = |C|$) is defined by

$$P_{iI(j)} = \begin{cases} a_{ij}/\sum_{k\in N_i} a_{ik}, & \text{for } i \in F, j \in N_i \\ 1, & \text{for } i \in C, j = i \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The interpolation defined above, referred to as *direct interpolation*, assigns for each fine point a coarse interpolatory set. However, a different interpolation may be defined such that the value for a fine point i can be interpolated from other fine points sharing a strong connection with i. This type of interpolation, referred to as *indirect interpolation*, is discussed in [5, Appendix A]. Indirect interpolation allows the use of more aggressive coarsening, i.e., selecting a much smaller coarse set in each coarsening step, while maintaining comparable efficiency. Trottenberg et al. [5, Appendix A] discuss the indirect interpolation for the AMG case and compare its performance with the performance of direct interpolation. Indirect interpolation is demonstrated for the EIS solver with BAMG interpolation (Section 4.2).

3.2.3 Derivation of the Coarse Equations

The derivation of the coarse equations follows the Full Approximation Scheme, which originally was developed for nonlinear equations [8]. Consider the fine level problem $Mu = \tau$, with $u = \tilde{u} + Pe^c$, where $\tilde{u}$ denotes the current approximation and e^c is the coarse-level correction. The fine level problem can be written as

$$Mu - M\tilde{u} = \tau - M\tilde{u}. \tag{10}$$

An analogous coarse level problem is

$$M^c u^c - M^c P^T \tilde{u} = P^T(\tau - M\tilde{u}), \tag{11}$$

where M^c can be computed, for example, by Galerkin coarsening $M^c = P^T AP - \lambda P^T BP$ [6]. The sparseness of M^c is maintained by employing an interpolation dilution procedure in which the interpolation caliber is limited to a

certain small number p, by updating N_i to be $N_i^{\text{diluted}} = \{j \in C, a_{ij} < 0 \text{ and } a_{ij} \text{ is among the } p \text{ largest } \{|a_{ik}|\}_{k \in C}\}$. In our experiments $p = 2$ or 4.

Rewriting (11) yields the following coarse level equations:

$$M^c u^c = P^T \tau + M^c P^T \tilde{u} - P^T M \tilde{u}. \tag{12}$$

If we denote the right-hand-side by τ^c, it can be seen that the general form, given by (6), is preserved. τ^c is called the fine-to-coarse defect correction. The starting approximation for solving (12) is $P^T \tilde{u}$.

Coarse level correction. The coarse level equations are solved by recursion, employing relaxation, and still coarser levels. Then, the coarse level correction is transferred to the fine level, i.e.,

$$u^{new} = \tilde{u} + P(u^c - P^T \tilde{u}). \tag{13}$$

Ritz projection. Once the vectors $u_1, \ldots, u_d$ have been approximated by a multilevel cycle, the subspace $X = span\{u_1, \ldots, u_d\}$ is a good approximation to the subspace spanned by the eigenvectors $U_1, \ldots, U_d$ that we seek. Ritz projection is a process which finds $\overline{u}_1, \ldots, \overline{u}_d$ in X and $\overline{\lambda}_1, \ldots, \overline{\lambda}_d$ so that the orthogonal projection of such $A\overline{u}_i - \overline{\lambda}_i B\overline{u}_i$ is zero. This ensures that any generalized eigenvector contained in X will be found by Ritz projection. More generally, in a certain sense, it will determine a basis for X that is closest to the $U_i's$.

To determine the Ritz vectors, we first perform a Gram-Schmidt orthonormalization on $u_1, \ldots, u_d$ (with respect to the matrix B), resulting in vectors $\widetilde{u}_1, \ldots, \widetilde{u}_d$. The matrix $\Phi = [\widetilde{u}_1, \ldots, \widetilde{u}_d]$ is defined. Then, $\Phi\Phi^T$ is an orthogonal projection operator onto X. Any vector in X can thus be written as Φz, where $z \in R^d$. Letting $\overline{u}_i = \Phi z_i$, then Ritz projection attempts to find z_i and $\overline{\lambda}_i$, $i = 1, \ldots, d$, so that

$$\Phi\Phi^T(A\Phi z_i - \overline{\lambda}_i B\Phi z_i) = 0 \tag{14}$$

or, since Φ is full rank, so that

$$\Phi^T A\Phi z_i - \overline{\lambda}_i \Phi^T B\Phi z_i = 0. \tag{15}$$

Hence, the Ritz process requires the solution of a relatively small $(d \times d)$ generalized eigenvalue problem. The obtained vectors $\overline{u}_1, \ldots, \overline{u}_d$ are normalized and used to initiate the next multilevel cycle.

Algorithm initialization. The algorithm is initialized by a simple multilevel cycle. First, the matrices A and B are iteratively coarsened by the Galerkin method. That is, at a fine level where matrices A and B are defined, the next level matrices are constructed as $A^c = P^T AP$ and $B^c = P^T BP$, where P is the interpolation operator that is constructed as in (9). The coarse level should have enough degrees of freedom to approximate the first d eigenvectors of the generalized eigenvalue system involving A^c and B^c. Typically, for 2D problems, the coarse level has at least $4d$ number of degrees of freedom. The d eigenvectors of the system $A^c U^c = \lambda B^c U^c$ are directly computed and then interpolated and relaxed until the finest level is reached. Finally, the d fine eigenvectors are corrected by the Ritz projection.

3.3 The Exact Interpolation Scheme (EIS) Solver

The FAS scheme described in Section 3.2 with classical AMG interpolation is very efficient for the computation of the first *few* eigenvectors of (3). However, in certain tasks, such as multiclass spectral clustering [41], one would like to compute *many* of the low eigenvectors with improved efficiency and accuracy. In such tasks, the classical AMG interpolation designed to interpolate the "smooth" error function is usually not suitable to accurately interpolate higher eigenvectors. For such applications, we suggest the Exact Interpolation Scheme eigensolver [9] combined with the BAMG interpolation. In the EIS solver, the interpolation P is repeatedly fitted to accurately interpolate the current approximations of the eigenvectors. As a result, no defect correction τ is needed in the coarse equations.

Moreover, with the BAMG approach, a single interpolation P which is simultaneously accurate for many low eigenvectors can be inexpensively constructed. As a result, a single eigenproblem is obtained at the coarse level whose eigenvectors, when interpolated by P, yield very good approximations to a multitude of low eigenvectors. The accuracy can further be enhanced by increasing the caliber of P. We will show that even one EIS cycle yields satisfactory accuracy. For simplicity, we will focus on a two-level cycle.

The two-level EIS solver explained below consists of similar steps as the FAS solver described earlier. Namely, it encodes procedures for choosing coarse variables, constructing an interpolation operator, and constructing the coarse equations. Below we explain the main modifications introduced by the two-level EIS algorithm.

Selection of coarse variables. For 2D grid problems which originate from data sets in which the data points lie on a 2D grid (such as the 2D differential Laplacian eigenproblems), the *standard grid coarsening* is used. Specifically, along each direction on the grid, every second point is chosen to be coarse. For unstructured problems, such as those emerging in various data analysis tasks, the coarse variables selection is based on the same AMG procedure described in Section 3.2.

Interpolation. For notation convenience the approximation of the kth eigenvector U_k is denoted by $\tilde{u}^{(k)}$. In order to accurately interpolate many eigenvectors, the interpolation operator P is computed to satisfy best, in the *weighted least squares sense*, $\tilde{u}^{(k)} = P\tilde{u}^{(k)c}$, simultaneously for all current approximate eigenvectors $\tilde{u}^{(k)}$ and their coarse level representations $\tilde{u}^{(k)c}$ $(k = 1, \ldots, d)$. In fact, for the approximation of many eigenvectors, only a small set of K representative low-residual *Test Vectors* (TVs) is enough to produce suitable P. For example, let $x^{(k)}$ and $x^{(k)c}$, $1 \le k \le K$, $K \ll d$, be a set of fine TVs and their coarse representation, respectively. In our implementation $x^{(k)}$ is a set of random vectors smoothed by relaxation of $Ax^{(k)} = 0$ and $x^{(k)c}$ is obtained by *injection*, i.e., $x_j^{(k)c} = x_m^{(k)}$ for each $m \in C$, where j is the coarse level index corresponding to the fine index m. The relaxed $x^{(k)}$ are linear combinations of low eigenvectors since relaxation dumps the components in $x^{(k)}$ that correspond to high eigenvectors. The set of interpolation coefficients $\{P_{ij}\}_j$ for each $i \notin C$ can be determined so that it best satisfies, in the weighted least squares sense, the over determined set of equations:

$$x_i^{(k)} = \sum_{j \in N_i} P_{ij} x_j^{(k)c}, \quad (k = 1, \ldots, K), \tag{16}$$

where N_i is the set of coarse points used to interpolate to i. The weight of the kth equation is chosen to be proportional to $\|Ax^{(k)}\|^2$.

Coarse equations. The interpolation operator P that best satisfies (16) is used to construct the coarse eigenequations:

$$A^c u^{(k)c} = \lambda^{(k)} B^c u^{(k)c}, \quad \left(u^{(k)c}\right)^T B^c u^{(k)c} = 1 \quad (1 \le k \le d), \quad (17)$$

where $A^c = P^T AP$ and $B^c = P^T BP$. Unlike the FAS scheme, no defect (i.e., τ) is transferred to the coarse level and the coarse equations have the same form of the generalized eigenvalue problem. Equation (17) is then solved for $u^{(k)c}$ and $\lambda^{(k)}$. At convergence, orthogonality of eigenvectors at the coarse level implies orthogonality at the fine level since P is constructed to accurately interpolate the coarse representation of the fine eigenvectors. The eigenvectors are interpolated to the finer level by utilizing the designed interpolation, followed by relaxation.

One of the special features of the EIS is that eigenvectors do not need to actually be represented at the finest level. This can not only save most of the computer time and storage, but is actually even preferable for many applications. For example, it facilitates a very efficient expansion of a function f in terms of the eigenvectors, by computing each expansion coefficient as the inner product of f with the corresponding eigenvector u using the relation

$$(f, u) = (f, Pu^c) = (P^T f, u^c) = (f^c, u^c). \quad (18)$$

So, the inner products can be calculated to a high accuracy only at the coarse level!

4 Experiments

In the first part of the experiments, we demonstrate and analyze the performance of the FAS solver with classical AMG interpolation on three types of data analysis problems, in each of which only few eigenvectors are required: dimensionality reduction, image segmentation, and clustering. In the second part of the experiments, the EIS-based algorithm is demonstrated to compute inexpensively many eigenvectors for 2D grid eigenproblems and for the task of multiclass spectral clustering. Additionally, we show that the EIS can be efficiently used for accurate computation of a specific low eigenvector.

4.1 The FAS Solver

4.1.1 Dimensionality Reduction

A data set of $N = 100{,}000$ data points is generated in a 3D space $X = (x, y, z)$. The x and y-coordinates are drawn from a uniform distribution in the interval $[0, 1]$. The z-coordinate is given by $z = sin(\pi x)tan(\pi y)$. This formulates a landscape of two peaks, known as the "twin peaks" landscape. Then a sparse and symmetric affinity matrix W is constructed as follows: The pairwise affinities are computed for each point X_i as in [17], i.e., $w_{ij} = \exp(-\frac{\|X_i - X_j\|^2}{\sigma^2})$, $(\sigma = 1)$, where $\|\cdot\|$ is the euclidean norm and X_j is among the eight nearest neighbors of X_i, determined by the KD-tree algorithm [43], [44]. The symmetry of W is maintained by adding X_i to the nearest neighbor list of each neighbor X_j of X_i. Following this construction, while maintaining symmetry, the maximal number of nonzero entries per row in the affinity matrix W is 16.

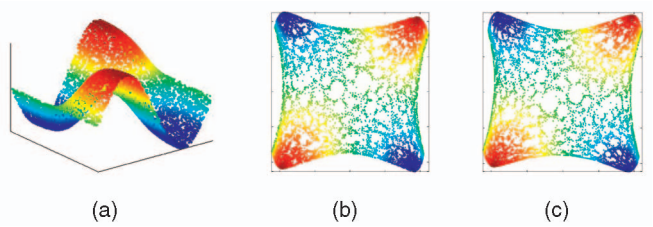


Fig. 1. Dimensionality reduction of the "twin peaks," with "Laplacian eigenmaps." (a) The 3D data (b) embedding into 2D using Lanczos method, (c) embedding into 2D using the multilevel eigensolver.

The Laplacian Eigenmaps algorithm [17] attempts to find a mapping of the data points set X in a given high dimension into a lower dimensional representation Y, where neighboring points in the high dimension are close to each other in the low dimension, more formally, such that $\sum_{ij}(y_i - y_j)^2 w_{ij}$ is minimized under the constraint $Y^T DY = I$, where D is diagonal $d_{ii} = \sum_j w_{ij}$. This optimization problem can be translated to an eigenvalue problem that seeks for the first $d + 1$ eigenvectors of $LU = \lambda DU$, where $L = D - W$ is the graph Laplacian matrix and d is the lower dimension ($d = 2$ in our example). The first eigenvector is constant, and therefore, disregarded. The next d eigenvectors are used to employ the "Laplacian eigenmaps" low-dimensional embedding from the higher dimension to the lower one. Specifically, in our example by using the two first eigenvectors as 2D representation of the original 3D points. The multilevel eigensolver is applied to this problem and the obtained embedding result is compared with the result obtained by using the Lanczos sparse solver from the commercial package ARPACK [42]. The embedding of the points as seen in Fig. 1 preserves the global structure of the data: the color coding that corresponds to the z-coordinate in X is preserved in Y. In particular, the embedding results obtained by both solvers are similar. However, our nonoptimized code is much faster than the Lanczos algorithm. Specifically, for a residual accuracy of 10^{-4}, the running time of our eigensolver, implemented by a nonoptimized code is 25 seconds, while the running time of Lanczos is 244 seconds. We note that the Lanczos running time is hardly affected by changing the accuracy criterion since it usually obtains very high accuracy. We have therefore chosen, here and below, to base comparisons on the level of accuracy sufficient in the applications. More important is the running time as a function of the problem size, which is discussed below (see Fig. 5).

4.1.2 Image Segmentation

A well known spectral method for image segmentation is the normalized-cut method [14]. Given an image, it first uniformly samples for each pixel a certain number of neighbors, e.g., 20, drawn from a surrounding circle with fixed radius of 10 pixels. Then, the pairwise affinity matrix W is constructed according to the spatial proximity and a bank of filter responses as explained in [14]. A related eigenvalue problem seeks for the first eigenvectors of the eigenvalue problem $(I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}})U = \lambda U$. Those eigenvectors induce an embedding of the pixels into a low-dimensional space wherein a simple clustering method is used to determine the

Fig. 2. Image segmentation. (a) The original image, (b) segmentation result obtained by normalized-cuts utilizing Lanczos method, and (c) segmentation result obtained by normalized-cuts utilizing the multilevel eigensolver.

image partition to coherent segments. Alternatively, the eigenvectors can be used to assign pixels to segments, as suggested in [41].

In order to test our algorithm performance, we integrated the FAS eigensolver into the normalized-cuts application [15] and compared its performance with the performance of Lanczos. The results are demonstrated on a 386×323 gray scale image in Fig. 2, where the red curves indicate the segmentation of the image into coherent segments. The segmentation is obtained by using the five first eigenvectors while neglecting the first one. The segmentation results are nearly identical. However, for residual accuracy of 10^{-4} which was used in these results, the running time of our algorithm is 84 seconds, while the running time of Lanczos is 835 seconds. We further elaborate on this problem in the Section 4.1.4 below.

For the particular eigenvalue problem evolving in image segmentation tasks, we demonstrate the quality of the eigenvectors approximation and the running time behavior for computing a larger number of eigenvectors for several more image segmentation problems of different sizes. The matrices are constructed similarly to the above construction. The results are demonstrated for images of sizes 233×180 pixels (41,940 nodes), 441×331 pixels (145,971 nodes), and 1024×768 pixels (786,432 nodes), where, in each problem, 10 eigenvectors were computed. The average number of nearest neighbors of each pixel is 20, 20, and 45, for each image, respectively. In order to obtain informative filter responses in the $1{,}024 \times 768$ image, the neighbors of each pixel were drawn from a circle with an increased radius of 16 pixels (instead of 10 pixels in all previous problems), thus increasing the number of nonzero entries and the nonlocality of the graph. The running times of our solver for a residual accuracy of 10^{-4} are: 74 seconds, 165 seconds, and 980 seconds, respectively. For the Lanczos solver, the running times obtained for the first two problems are 85 seconds, 1,124 seconds, whereas, for the last problem, the Lanczos solver ran for several hours and exhausted the machine memory without producing any results. The segmentation results as well as the plot of the second eigenvector are shown in Fig. 3. The plot of the second eigenvector values on the image domain demonstrates the high agreement between eigenvectors obtained by our method and the eigenvectors obtained by the Lanczos method.

The running time of our method obtained for this problem manifests roughly linear asymptotic dependence on the number of nodes in the graph. Another important result emerging from our experiments is that for matrices of size smaller than a certain number (perhaps in the thousands) and for a small number of computed eigenvectors, it is more efficient to use the Lanczos solver. The reason is the overhead of our multilevel construction and the multilevel cycles which are performed for each eigenvector separately. The significant improvement in running time obtained by our method is manifested for matrices of large size as for example in the 441×331 and the $1{,}024 \times 768$ images. The exact crossing point between the running times of both solvers cannot be determined from the experiments because the Lanczos implementation is optimized, whereas our solver implementation is not.

4.1.3 Clustering

Spectral clustering algorithms propose to map the original data into the k first eigenvectors of the affinity matrix (or a matrix similar to it) and then apply a standard clustering algorithm such as k-means on these new coordinates. In order to test the multilevel eigensolver, the spectral clustering algorithm suggested by [13] was implemented while solving the related generalized eigenvalue problem $(I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}})U = \lambda U$, once by our eigensolver and once by Lanczos algorithm. To that end, we generated synthetic data composed of two 2D concentric rings, with radii 0.25 and 0.5 and widths determined by the normal distribution with standard deviation 0.025. The pairwise affinities are computed for each data point X_i as in [13], $w_{ij} = \exp(-\frac{\|X_i - X_j\|^2}{\sigma^2})$, where $\sigma = 0.07$, and X_j is among the eight nearest neighbors of X_i, determined by the KD-tree algorithm. Following this construction, while maintaining symmetry, the maximal number of nonzero entries per row in the affinity matrix W is 16.

Utilizing the first three eigenvectors, the clustering results obtained by the eigensolver and by Lanczos are identical, as exemplified for 250,000 data points in Fig. 4. The color coding shows the assignment of points to two distinct clusters induced by using the computed eigenvectors within the algorithm of [13]. The running times for 10^{-4} residual accuracy are 33 seconds and 383 seconds, for our eigensolver and Lanczos, respectively.

4.1.4 Algorithm Performance

The performance of the FAS solver is evaluated by measuring the *average convergence factor per work unit*, or in other words, the trade-off between the accuracy and the algorithm runtime. Since one does not generally know the exact

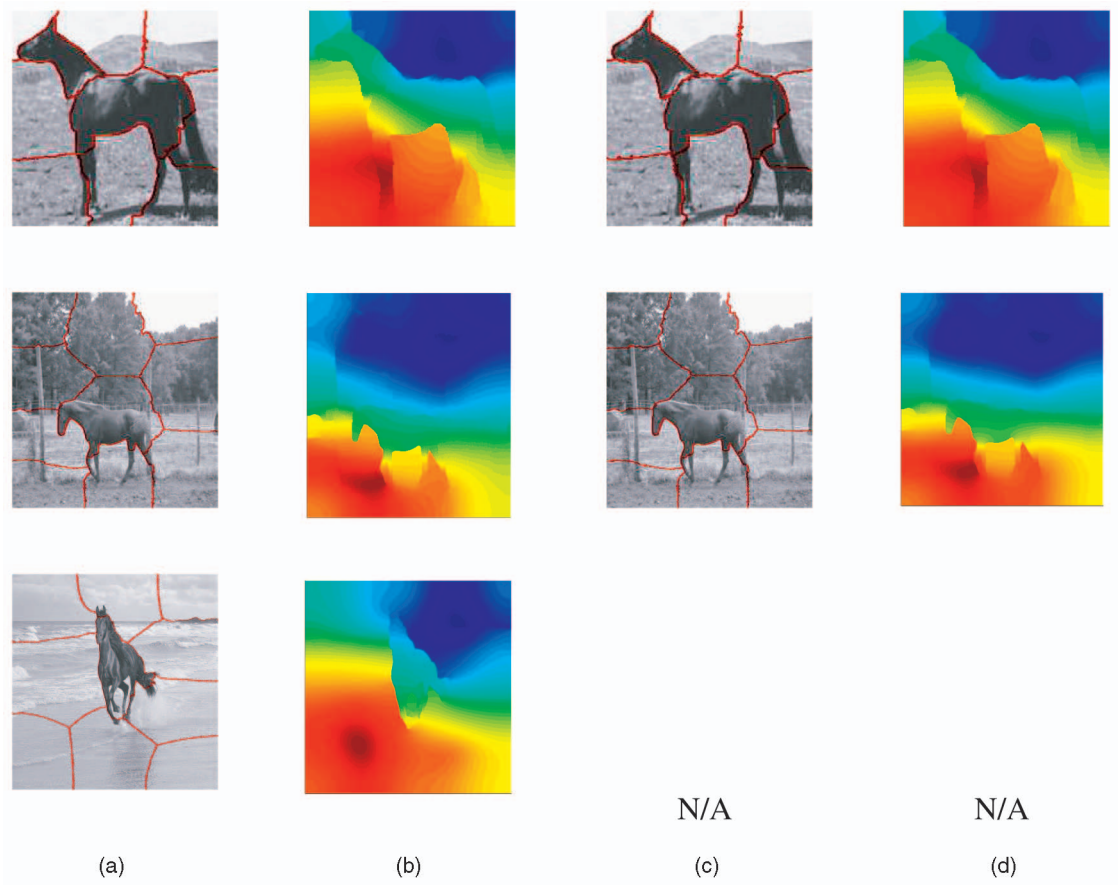


Fig. 3. Image segmentation. (a), (b) Segmentation and second eigenvector plot, obtained by normalized cuts utilizing the multilevel eigensolver. (c), (d) Segmentation and second eigenvector plot, obtained by normalized cuts utilizing the Lanczos method.

solution of the eigenvalue problem, the accuracy is measured in terms of the residual. Define r_i^k to be the residual obtained by the approximation u_i^k to the ith eigenvector, after employing k cycles, i.e., $r_i^k = \|Au_i^k - \lambda_i^k Bu_i^k\|$. The convergence factor at a certain cycle k for the ith eigenvector measures the decrease of the residual between two successive cycles, i.e., $\mu_i^k = r_i^k / r_i^{k-1}$. Averaging this convergence factor over m cycles yields the average convergence factor $\bar{\mu}_i$ (in our experiments, $\bar{\mu}_i$ is computed with $m = 5$). The work per cycle, w, is measured by $w = \sum_s (v^{[s]} + 1)\gamma^{[s]}$, where $v^{[s]}$ is the number of relaxation sweeps performed at level s and $\gamma^{[s]}$ is the total number of nonzero entries of the matrix at level s divided by the total number of nonzero entries of the fine level matrix $M = A - \lambda B$. The work amount equivalent to one relaxation sweep is added at each level to account for constructing the coarse equations. The *average convergence factor per work unit* is defined by $\rho_i = \bar{\mu}_i^{1/w}$. Table 1 presents ρ for different problems solved by the FAS eigensolver. The high value attained for the image segmentation problem is due to the higher density of the problem and the special graph structure: On average, 20 neighbors are sampled from a radius of 10 pixels and their affinities are based on filter responses, thus suggesting a larger number of geometrically distant connections between pixels. In this sense, the locality of the graph is, to some extent, violated (so, ideally, we should have used here the BAMG interpolation). The number of nonzero entries at each level is 2,757,069, 3,409,468, 839,238, 200,344, 56,214, 21,297, and 7,177, producing a cycle of 8.3 work units. This manifests itself in the relatively higher running time reported above for this problem.

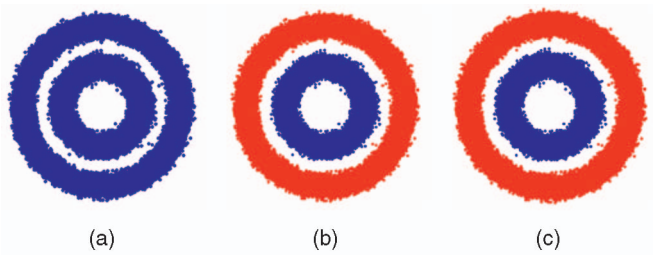


Fig. 4. Clustering. (a) The data set, (b) clustering result obtained by spectral clustering [13] utilizing Lanczos method, and (c) clustering result obtained by spectral clustering utilizing the multilevel solver.

Additionally, to verify that our eigensolver performance is optimized with respect to the multigrid textbook efficiency, we have compared its performance with the optimal theoretical multigrid convergence factor per work unit μ^o, as defined (slightly differently than $\bar{\mu}$) and computed in [8] for problems discretized on grids. The comparison is done for the matrix obtained by discretizing the 2D Laplace differential equation on a uniform grid. For the second eigenvector, our solver obtained $\mu^o = 0.542$ for the 10K problem and $\mu^o = 0.592$ for the 100K problem, which is in good agreement with the asymptotic value $\mu^o = 0.595$ reported in [8, Table 1].

TABLE 1
Average Convergence Factor per Work Unit for Different Problems

problem	Lap Eq.	Lap Eq.	Dim. Red	Dim. Red	Segment	Cluster
size	10K	100K	50K	100K	120K	250K
non-zeros per row	4	4	16	16	20	16
# rlx	3	3	2	2	2	2
ρ_2	0.66	0.62	0.79	0.77	0.85	0.71
ρ_3	0.68	0.68	0.81	0.83	0.85	0.71
ρ_4	0.68	0.71	-	-	0.85	-
ρ_5	0.72	0.71	-	-	0.85	-

4.1.5 Comparison to Lanczos Method

The running time of our FAS solver is compared with the running time of the Lanczos solver [42] (ARPACK software package implemented in Fortran). We indicate that our eigensolvers code is compiled of C and Matlab procedures that are not optimized to achieve minimal running time. The convergence of the Lanczos algorithm follows the Kaniel-Page Theorem [3]: The ith eigenvector convergence depends on the separation between the ith and $(i+1)$th eigenvalues relative to the separation between the $(i+1)$th and the largest eigenvalue. For large Laplacian matrices, the small relative eigenvalue separation causes a significant slowdown in the convergence of the Lanczos algorithm. The comparison of the running time as a function of the problem size is demonstrated in Fig. 5 for the computation accuracy of 10^{-4} for the second eigenvector in each of the three representative eigenvalue problems described above. The results clearly indicate that our FAS solver running time is linear in the number of data points, whereas the Lanczos running time is super-linear. Similar results are obtained for the third eigenvector.

4.2 The EIS Solver

2D grid Laplacian problems. To demonstrate the accuracy obtained by the EIS solver, we apply it for computing the eigenvectors of (3) for the Laplacians of two graphs:

1. *Constant Laplacian*: The nodes of the graph comprise a 125×125 grid with edges only between nearest neighbors, all having the same constant weight; hence the matrix A corresponds to the discretization of the 2D differential Laplacian.
2. *Random Laplacian*: The same, except that the weights are random numbers, each drawn from a uniform distribution on the interval $[0, 1]$.

The matrix B in both cases is the identity.

For constructing the coarse equations and interpolating the coarse level solution, two interpolation calibers are used: caliber of size 2 and caliber of size 4. For caliber of size 2, the interpolation operator is the classical bilinear stencil and its adjoint is used for coarsening the equations (i.e., restriction). The caliber-2 stencil is illustrated in Fig. 6a. For the caliber-4 case, the restriction of the current approximation and the construction of the coarse equations by Galerkin coarsening are done with the stencil illustrated in Fig. 6b, whereas the interpolation of the coarse level solution is actually performed in two steps with the stencil illustrated in Fig. 6c, namely, the coarse to fine interpolation is first performed for two types of fine points (red and green). Second, the value at the third type of points (pink) is determined by an LSE-based interpolation from *fine* grid points. The two-step interpolation of the solution yields higher accuracy than using the stencil illustrated in Fig. 6b. For fine boundary points, the caliber-4 stencils (Fig. 6b and Fig. 6c) are modified so that each fine point is interpolated noncentrally from four points (e.g., three internal points and one boundary point) in each relevant direction.

As explained in Section 3 above, a great potential advantage of the EIS solver is that one can calculate many

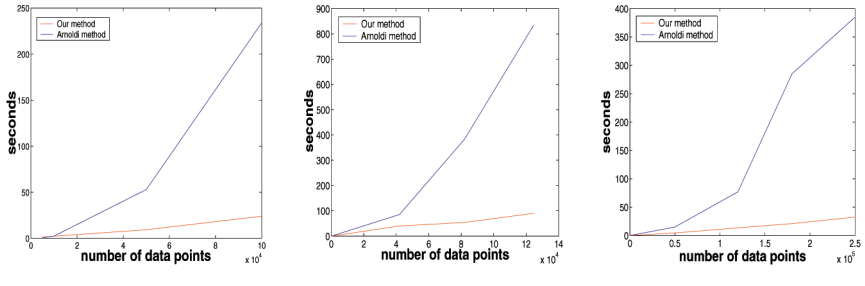


Fig. 5. Performance comparison between Lanczos eigensolver and the multilevel eigensolver (to 10^{-4} residual accuracy): number of data points versus running time (in seconds) (a) dimensionality reduction, (b) image segmentation, and (c) clustering.

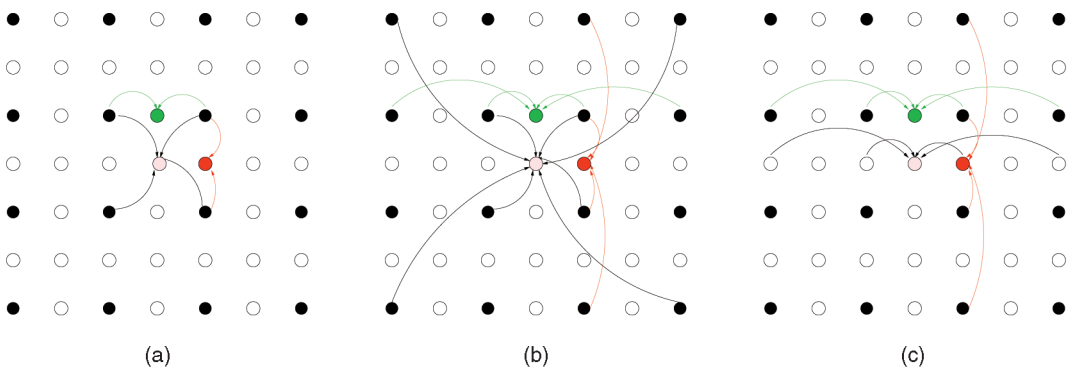


Fig. 6. Restriction and interpolation stencils for 2D grid domain. (a) Caliber-2 interpolation stencil. (b) Caliber-4 restriction stencil. (c) Caliber-4 interpolation stencil. Full black circles correspond to coarse nodes. Empty circles correspond to fine nodes. The stencils are sketched for three typical grid nodes (green, red, and pink).

eigenvectors accurately without actually computing each of them at the finest level. Here, however, to demonstrate the obtained accuracy, we do interpolate the coarse EIS vectors to the fine level and compare them, together with their corresponding eigenvalues, to the exact fine-level eigenvectors and eigenvalues. We have measured three types of errors: the *relative eigenvalue error*, the *interpolation error*, and the *eigenvector error*. The relative eigenvalue error is measured by:

$$d_{val}^{(k)} = \frac{\left|\lambda^{(k)} - \lambda_{Lanc}^{(k)}\right|}{\lambda_{Lanc}^{(k)}}, \tag{19}$$

where $\lambda_{Lanc}^{(k)}$ is the kth smallest eigenvalue computed by Lanczos and $\lambda^{(k)}$ is the kth smallest eigenvalue computed for our results by using the Rayleigh Quotient

$$\lambda^{(k)} = \frac{\tilde{u}^{(k)T} A \tilde{u}^{(k)}}{\tilde{u}^{(k)T} \tilde{u}^{(k)}}, \tag{20}$$

where $\tilde{u}^{(k)}$ is the relaxed kth eigenvector approximation obtained by interpolation of the corresponding coarse level eigenvector. Note that we have reordered the eigenvectors so that $\lambda^{(k)}$ are indeed in ascending order, similar to $\lambda_{Lanc}^{(k)}$, instead of the modified order that sometimes arises at the coarse level between very close eigenvalues. The interpolation error is given by

$$d_{intrp}^{(k)} = \left\| v^{(k)} - P v^{(k)c} \right\|, \quad k = 1, \ldots, 100, \tag{21}$$

where $v^{(k)}$ is the kth eigenvector of (3) computed by Lanczos and $v^{(k)c}$ is the injection of $v^{(k)}$ to the coarse grid (as explained in Section 3.3). The eigenvector errors are computed by

$$d_{lc}^{(k)} = \left\| \tilde{u}^{(k)} - \tilde{v}^{(k)} \right\|, \quad k = 3, \ldots, 98, \tag{22}$$

where

$$\tilde{v}^{(k)} = \sum_{k-2 \le i \le k+2} \alpha_i v^{(i)},$$

with the coefficients α_i that minimize

$$\left\| \tilde{u}^{(k)} - \sum_{k-2 \le i \le k+2} \alpha_i v^{(i)} \right\|.$$

The linear combination $\tilde{v}^{(k)}$ of Lanczos eigenvectors is used to reduce error components that result from subspaces containing several eigenvectors with the same eigenvalues, or from eigenvectors with very close eigenvalues whose order changes at the coarse level.

In Fig. 7, we show the errors obtained when using the two-step interpolation (i.e., indirect interpolation) over using the direct interpolation (with the stencil of Fig. 6c). As seen, the use of interpolation from nearby fine points reduces the interpolation error. Figs. 8 and 9 report the errors for 100 eigenvectors when using caliber of size 2 interpolation versus using caliber of size 4 for interpolation, for the two types of graph Laplacians, respectively. We have used the minimal number of TVs and relaxation sweeps to obtain highest accuracy. Thus, for the caliber of size 2, we used 8 TVs relaxed by 10 sweeps, and for the interpolation with caliber size 4 we used 16 TVs with 30 relaxation sweeps for the constant Laplacian, and 12 TVs with 10 relaxation sweeps for the random Laplacian. Note that the exceptionally large values seen in the diagrams for some eigenvector errors are due to eigenvectors whose ordinal number k, despite the above reordering, is still different from their original ordinal number at the fine level (due to slight changes in eigenvalues), and the difference is more than the two ordinal places taken into account in the definition of $\tilde{v}^{(k)}$.

The purpose of the random Laplacian experiment is to show that the EIS method for calculating many eigenvectors is not restricted to smooth solutions. This is clearly demonstrated in the caliber-2 experiment. However, unlike the constant Laplacian case, in the random case the higher caliber experiment (caliber-4) does not exhibit any substantial improvement over the lower caliber experiment. The reason is obvious: The pair of four collinear points chosen here for the caliber-4 interpolation does yield a higher (fourth) order interpolation in the smooth case; any other, not collinear choice of four points would not so raise the interpolation order and would not yield substantial improvements; without collinearity a larger set of interpolation points is needed to raise the order. In the random Laplacian case, collinearity has no particular relevance, so a much larger set of interpolation points must be used to achieve the equivalent of a higher order interpolation. Indeed, the full BAMG scheme can include a procedure to choose such higher order-yielding sets, based on the size of

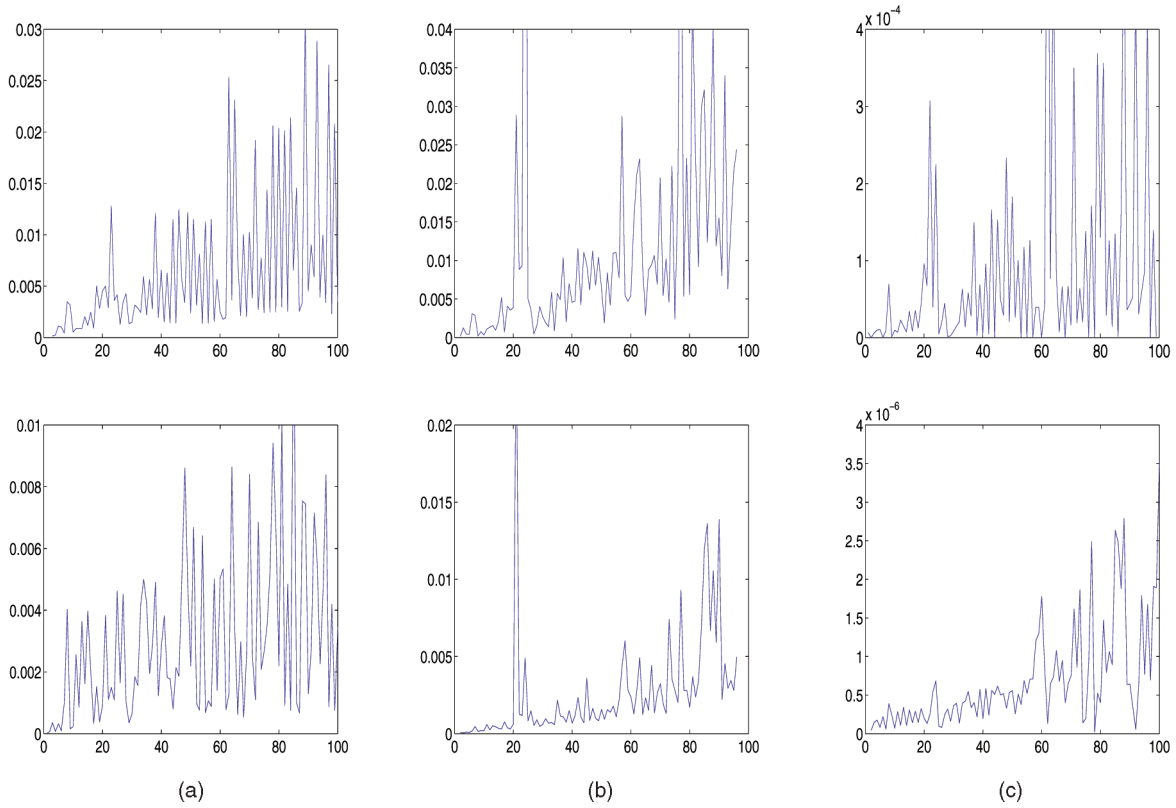


Fig. 7. Errors comparison for direct interpolation and two-step interpolation for the 2D constant Laplacian with caliber-4 interpolation. Upper row: direct interpolation. Bottom row: two-step interpolation. (a), (b), and (c): interpolation, eigenvector, and relative eigenvalue errors, respectively.

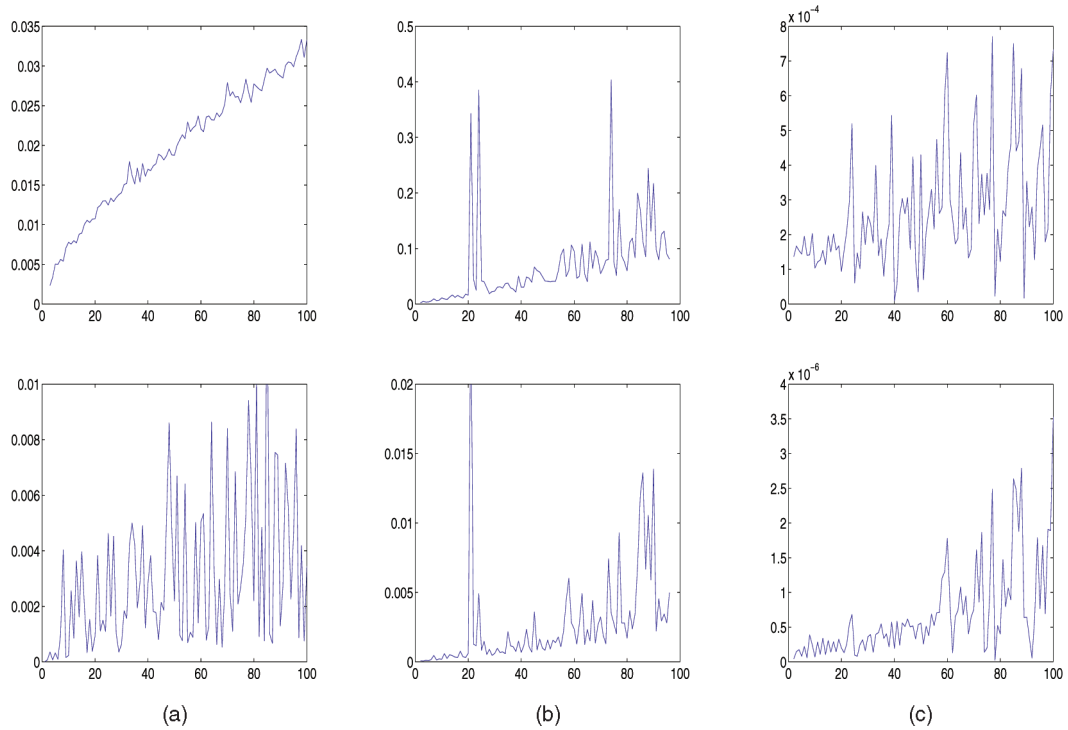


Fig. 8. Errors for computing the 100 eigenvectors of 2D constant Laplacian. Upper row: caliber of size 2 interpolation. Bottom row: caliber of size 4 interpolation. (a), (b), and (c): interpolation, eigenvector, and relative eigenvalue errors, respectively.

the least squares fit, but such a procedure has not been constructed here.

Multiclass spectral clustering with many eigenvectors. Next, we investigate the fast EIS calculation of many eigenvectors for a model data-analysis problem. We show in this context that the BAMG interpolation yields significantly more accurate approximations than the classical AMG interpolation. It is also shown that, even for aggressive

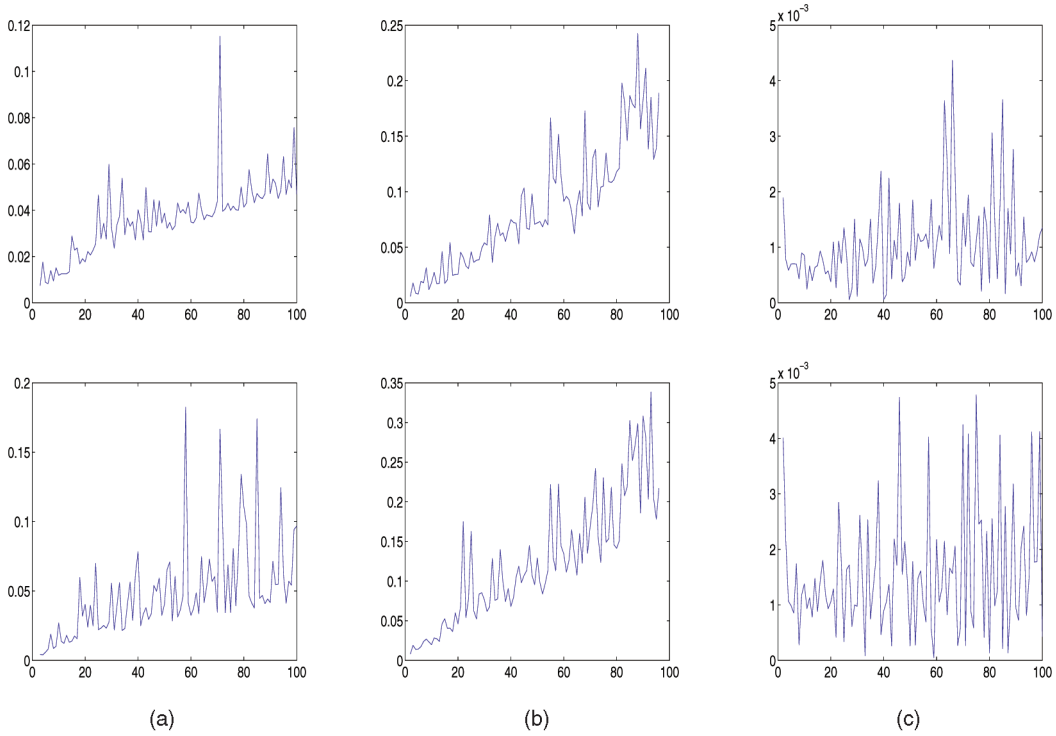


Fig. 9. Errors of computing the 100 eigenvectors of 2D random Laplacian. Upper row: caliber of size 2 interpolation. Bottom row: caliber of size 4 interpolation. (a), (b), and (c): interpolation, eigenvector, and relative eigenvalue errors, respectively.

coarsening, i.e., using significantly less variables at the coarse level, the BAMG interpolation still yields accurate results. For these purposes, we use a data set of 2D Gaussian mixture model. The data set contains 50,000 data points drawn from a set of 100 Gaussian distributions $N_i(\mu_i, \sigma_i^2 I)$, $i = 1, \ldots, 100$, with centers $\mu_i = (i_1, i_2)$, where $i_1, i_2 = 1, 2, \ldots, 10$ and $\sigma_i = 0.2$. A weighted graph is constructed with the pairwise similarity weights $w_{ij} = \exp(-\frac{\|X_i - X_j\|^2}{\sigma^2})$, where $\sigma = 0.1$ and X_j is among the 30 nearest neighbors of X_i as computed by the KD-tree algorithm. Since the number of clusters is a priori known, we have computed a hundred eigenvectors of $AU = \lambda U$, where $A = (I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}})$ and D is the diagonal matrix $d_{ii} = \sum_j w_{ij}$. The assignment of points to clusters is based on the multiclass spectral clustering algorithm suggested in [41].

To demonstrate how accurately many eigenvectors can be calculated by just one EIS coarsening (and hence, potentially avoid computing them at the fine level), we use a single cycle composed of 2-levels. We compare the performance of our two interpolation schemes: the BAMG interpolation and the classical AMG interpolation. In both cases, the coarse variable choice is done according to the AMG procedure for coarse variable choice described in Section 3.2. For the BAMG cycle, we construct the interpolation matrix P (as explained in Section 3.3), and use it to construct the coarse eigenproblem. After the coarse problem is solved, P is used to interpolate the coarse eigenvectors back to the fine level, followed by 10 relaxation sweeps of the interpolated eigenvectors and the different error types are measured. In the AMG case, we use the classical interpolation matrix (constructed as explained in Section 3.2) instead of fitting an interpolation as in BAMG. Then the same steps are done as in the BAMG cycle. The interpolation caliber in each case is 4. For the BAMG interpolation, the least squares construction employs 6 TVs, each relaxed by 30 sweeps.

Additionally, for each solver choice we conducted two experiments, by generating two coarse sets: one by using a coarsening factor of $\alpha = 0.2$, yielding a coarse set of size $|C| = 12{,}908$, and the other with a smaller coarsening factor $\alpha = 0.1$, yielding $|C| = 7{,}722$.

To evaluate the performance of the two interpolations in terms of accuracy, the average errors d_{intrp}, d_{lc}, and d_{val} for $1 \leq k \leq 100$ are reported in Table 2, along with the size and work parameters of the 2-level cycle. An order of magnitude improvement is observed for the interpolation error, and one to two orders of magnitude improvement in the eigenvector error and the relative eigenvalue error, when using the BAMG instead of the classical AMG interpolation.

The clustering results for both interpolation types are demonstrated in Fig. 10, where the color coding indicates the assignment of points to 100 distinct clusters induced by using the computed eigenvectors. The results demonstrate that numerically the BAMG interpolation-based eigensolution is more accurate than the eigensolution obtained with the AMG classical interpolation. Roughly, the clustering obtained by both schemes is similar. However, a closer examination reveals misclassifications in which the clustering obtained with the classical AMG interpolation manifests visible intercluster intrusions, a sample of which is zoomed into in Fig. 10c. As seen in Fig. 10d, such intrusions do not exist for the clustering induced by the BAMG interpolation-based eigenvectors.

Fast and accurate computation of a single low eigenvector. The EIS solver is demonstrated below to accurately

TABLE 2
Accuracy Comparison for Computing 100 Eigenvectors by a Single Two-Level Cycle Employing AMG Classical Interpolation and BAMG Interpolation

Exprmnt	# data pts	# coarse nodes	Non-zeros A	Non-zeros A^c	avg d_{lc}	avg d_{intrp}	avg d_{val}
$\alpha = 0.2$, classical	50000	12908	1781660	593226	~ 0.2	~ 0.038	~ 0.01
$\alpha = 0.2$, BAMG	50000	12908	1781660	593226	~ 0.006	~ 0.005	$\sim 1.4 \times 10^{-4}$
$\alpha = 0.1$, classical	50000	7722	1781660	287286	~ 0.32	~ 0.047	~ 0.016
$\alpha = 0.1$, BAMG	50000	7722	1781660	287286	~ 0.03	~ 0.01	$\sim 2.5 \times 10^{-4}$

compute a given eigenvector by increasing its weight in the least squares optimization of (16). For this purpose, a multicycle 2-level EIS solver involving BAMG interpolation is performed. As described in Section 3.3, we start by fitting an interpolation matrix to random TVs to obtain a first approximation to the eigenvectors, then the interpolation is refitted to this current approximation of the eigenvectors, and the process is repeated in subsequent cycles. The initial weight of the ith eigenvector is 10^{-i} for $1 \leq i \leq 10$. Then, to obtain an accurate approximation for the second eigenvector, its weight is increased by multiplying it by 100 on each subsequent cycle while keeping the other weights unchanged. In each EIS cycle, the interpolation matrix is recomputed to fit the new eigenvector approximations. The convergence factor for the second eigenvector is computed and compared with the convergence factor of a FAS two-level solver with classical AMG interpolation. The comparison is performed for the eigenproblem involving a $15{,}625 \times 15{,}625$ 2D constant Laplacian discussed above and for the GMM clustering eigenproblem. In both examples, the initial BAMG interpolation was based on six random test vectors relaxed by six relaxation sweeps, and an interpolation caliber of size 2. As seen in Fig. 11, the EIS solver has reached machine precision accuracy much faster than the FAS solver with classical AMG interpolation.

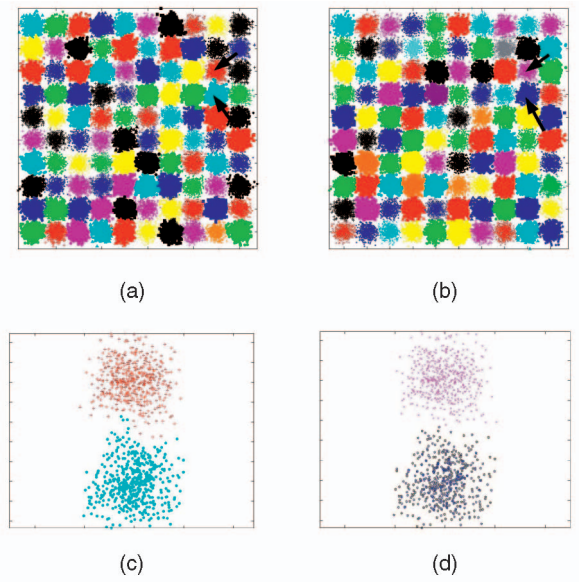


Fig. 10. Spectral clustering results of a Gaussian Mixture Model (GMM) containing 50,000 data points organized in 100 Gaussians. The points are clustered by using 100 eigenvectors of (2). (a) Clustering obtained with classical AMG interpolation. (b) Clustering obtained with BAMG interpolation. Each detected cluster is color coded; however, some neighboring clusters may have a similar color. Arrows identify a region being zoomed-in. (c) A zoomed-in sample of misclassification in which the clustering obtained with the classical AMG interpolation manifests visible intercluster intrusions. (d) The coherent clustering obtained for the same region with BAMG interpolation.

5 Cost versus Performance of Different Multigrid Solvers

The most basic multigrid solver is the Geometric Multigrid (GMG), which is the cheapest in terms of work: The problem is defined on a grid for which the intergrid operators need not be computed at all, the coarse node selection is based on standard coarsening, and the coarse equations need not be derived and are as simple (non-Galerkin) as the fine-grid equations. One step more general, and correspondingly more expensive, are the GMG solvers which use Galerkin coarsening, allowing the treatment of problems with disordered coefficients, still on structured grids. The next class of multigrid eigensolvers is based on the algebraic multigrid (AMG) technique [6], which is designed for unstructured problems. Such a solver, employing classical AMG interpolation with Galerkin coarsening is described in Section 3.2 for graph-based eigenproblems. This solver involves additional work for the construction of the interpolation operator. The next type of solver involves the BAMG interpolation that is still more expensive but is useful for several purposes:

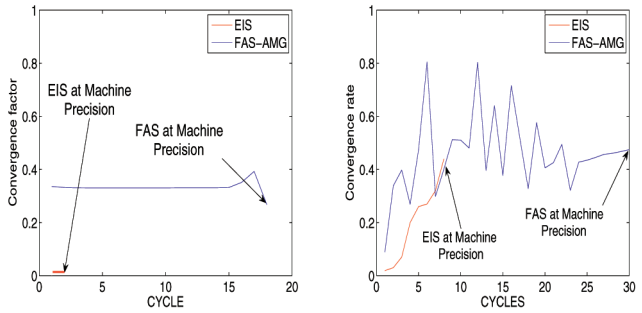


Fig. 11. Convergence factor comparison between the FAS two-level solver and the EIS two-level solver. (a) 2D constant Laplacian example. (b) The GMM example. Arrows indicate the stage at which the related solver obtained machine precision accuracy of the equation residual.

1. to obtain higher accuracy that is needed, e.g., for high order differential equations and for performing *upscaling* (once for all coarsening) in linear as well as in nonlinear problems (see [46]),
2. to obtain higher accuracy for graph problems that are not local (for which classical AMG interpolation fails),
3. to solve systems which are not diagonally dominant, where classical AMG often requires parameter tuning to be effective.

As discussed in our context, the EIS solver with BAMG interpolation has the additional benefits of simultaneously and inexpensively coarsening/upscaling *many* eigenvectors, and producing a coarse eigenproblem that has the standard (linear) form of eigenvalue equations (without the extra affine τ term). The beneficial features of the EIS solver require the extra work of reconstructing the interpolation in every cycle to fit the current approximation of the solution. As shown in the example above, for computing a single eigenvector, a very fast convergence can be obtained but note that, in this case, the extra work may not always worth it.

6 Conclusion

Motivated by the high complexity of spectral data analysis, efficient multilevel solvers for computing the eigenvectors of data driven matrices have been presented. The FAS solver with classical AMG interpolation is designed for solving large eigenvalue problem for the first few eigenvectors. Even the nonoptimized FAS solver has an order of magnitude improvement in running time when compared with the commercial Lanczos solver on various data analysis tasks. Also presented is an EIS solver that has the advantage of very efficiently computing many eigenvectors, as required, for example, in multiclass spectral clustering. The EIS is also demonstrated to compute with high accuracy and efficiency a specific low eigenvector.

We emphasize that the EIS solver with BAMG interpolation requires a certain amount of work in relaxing the random test vectors. If just a few low eigenvectors are desired with a relatively low residual accuracy, then the same work can be invested through a classical AMG interpolation-based solver (as our FAS) to obtain a solution with the same accuracy. Therefore, the choice between the two solvers should be based on the number of eigenvectors required and the desired accuracy of the computation.

Acknowledgments

The authors would like to thank the associate editor Kevin Murphy and the anonymous reviewers for their helpful comments. They also thank Boaz Nadler, Ronen Basri, Ilya Safro, Ofer Rahat, Gilad Barkan, and Leonid Karlinsky for useful discussions.

References

[1] D. Kushnir, M. Galun, and A. Brandt, "Fast Multiscale Clustering and Manifold Identification," *Pattern Recognition,* vol. 39, no. 10, pp. 1876-1891, Oct. 2006.

[2] I.S. Dhillon, Y. Guan, and B. Kulis, "Weighted Graph Cuts without Eigenvectors: A Multilevel Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 11, pp. 1944-1957, Nov. 2007.

[3] G.H. Golub and C.F. Van Loan, *Matrix Computations.* John Hopkins Univ. Press, 1989.

[4] A. Brandt, S. McCormick, and J. Ruge, "Algebraic Multigrid (AMG) for Automatic Multigrid Solution with Application to Geodetic Computations," report, Inst. for Computational Studies, 1982.

[5] U. Trottenberg, A. Schuller, and C. Oosterlee, *Multigrid.* Acdemic Press, 2001.

[6] A. Brandt, "Algebraic Multigrid Theory: The Symmetric Case," *Applied Math. and Computation,* vol. 19, no. 23, pp. 23-56, July 1986.

[7] A. Brandt, "Multiscale Scientific Computation: Review 2001," *Multiscale and Multiresolution Methods: Theory and Applications,* Springer-Verlag, 2002.

[8] A. Brandt, "Multilevel Adaptive Solutions to Boundary Value Problems," *Math. of Computation,* vol. 31, no. 138, pp. 333-390, Apr. 1977.

[9] A. Brandt and D. Ron, "Multigrid Solvers and Multiscale Optimization Strategies," *Multilevel Optimization and VLSICAD,* Kluwer, 2003.

[10] S. Kaczmarz, "Angenäherte Auflösung von Systemen Linearer Gleichungen," *Bull. Acad. Polon. Sci. et Lett. A,* pp. 355-357, 1937.

[11] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral Grouping Using the Nystrom Method," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 2, pp. 214-225, Feb. 2004.

[12] T. Cour, F. Benezit, and J. Shim, "Spectral Segmentation with Multiscale Graph Decomposition," *Proc. IEEE Int'l. Conf. Computer Vision and Pattern Recognition,* 2005.

[13] Y. Ng Andrew, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Advances in Neural Information Processing Systems,* vol. 14, pp. 849-856, MIT Press, Dec. 2001.

[14] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.

[15] Code obtained from the URL http://www.cis.upenn.edu/jshi/software, 2008.

[16] F. Chung, "Spectral Graph Theory," *Am. Math. Soc., 92,* 1997.

[17] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation,* vol. 15, no. 6, pp. 1373-1396, June 2003.

[18] C. Chennubhotla and A.D. Jepson, "Hierarchical Eigensolver for Transition Matrices in Spectral Methods," *Advances in Neural Information Processing Systems,* vol. 17, p. 273-280, MIT Press, Dec. 2005.

[19] A. Brandt, S. McCormick, and J. Ruge, "Multigrid Methods for Differential Eigenproblems," *SIAM J. Scientific and Statistical Computing,* vol. 4, no. 2, pp. 244-260, June 1983.

[20] S. McCormick, "A Mesh Refinement Method for $Ax = \lambda Ax$," *Math. of Computation,* vol. 36, no. 154, pp. 485-498, Apr. 1981.

[21] S. Costiner and S. Taasan, "Adaptive Multigrid Techniques for Large-Scale Eigenvalue Problems: Solutions of the Schrödinger Problem in Two and Three Dimensions," *Physical Rev. E,* vol. 51, no. 4, pp. 3704-3717, Apr. 1995.

[22] O. Livne and A. Brandt, "O(NlogN) Multilevel Calculation of N Eigenfunctions," *Multiscale Computational Methods in Chemistry and Physics,* IOS Press, 2001.

[23] I. Livshits, "An Algebraic Multigrid Wave-Ray Algorithm to Solve Eigenvalue Problems for The Helmholtz Operator," *Numerical Linear Algebra with Applications,* vol. 11, nos. 2-3, pp. 229-239, Mar. 2004.

[24] I. Livshits, "One-Dimensional Algorithm for Finding Eigenbasis of the Schrödinger Operator," *SIAM J. Scientific Computing,* vol. 30, no. 1, pp. 416-440, Nov. 2007.

[25] Y.A. Erlangga, C.W. Oosterlee, and C. Vuik, "A Novel Multigrid Based Preconditioner for Heterogeneous Helmholtz Problems," *SIAM J. Scientific Computing,* vol. 27, No. 4, pp. 1471-1492, 2006.

[26] H.C. Elman, O.G. Ernst, and D.P. Oleary, "A Multigrid Method Enhanced by Krylov Subspace Iteration for Discrete Helmholtz Equations," *SIAM J. Scientific Computing,* vol. 23, no. 4, pp. 1291-1315, 2001.

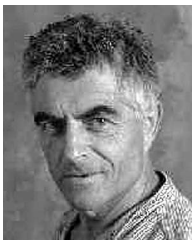
[27] A. Borzi and G. Borzi, "Algebraic Multigrid Methods for Solving Generalized Eigenvalue Problems," *Int'l J. Numerical Methods in Eng.,* vol. 65, no. 8, pp. 1186-1196, Sept. 2005.

[28] U. Hetmaniuk, "A Rayleigh Quotient Minimization Algorithm Based on Algebraic Multigrid," *Numerical Linear Algebra with Applications,* vol. 14, pp. 563-580, 2007.

[29] J. Mandel and S. McCormick, "A Multilevel Variational Method for $Au = \lambda Bu$ on Composite Grids," *J. Computational Physics,* vol. 80, pp. 442-452, 1989.
[30] M. Brezina, T. Manfeuffel, S. McCormick, J. Ruge, G. Sanders, and P. Vassilevski, "Smoothed Aggregation-Based Eigensolvers," *Numerical Linear Algebra with Applications,* vol. 15, pp. 249-269, 2008.
[31] P. Vanek, J. Mandel, and M. Brezina, "Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems," *Computing,* vol. 56, pp. 179-196, 1996.
[32] M. Brezina, R. Falgout, S. MacLachlan, T. Manfeuffel, S. McCormick, and J. Ruge, "Adaptive Smoothed Aggregation (αSA)," *SIAM J. Scientific Computing,* vol. 25, pp. 1896-1920, 2004.
[33] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, "Adaptive Smoothed Aggregation (αSA) Multigrid," *SIAM Rev.,* vol. 47, no.2, pp. 317-346, 2005.
[34] M. Brezina, R. Falgout, S. MacLachlan, T. Manfeuffel, S. McCormick, and J. Ruge, "Adaptive Algebraic Multigrid," *SIAM J. Scientific Computing,* Vol. 27, no. 4, pp. 1261-1286, 2006.
[35] H. de Sterck, T. Manteuffel, S. McCormick, Q. Nguyen, and J. Ruge, "Multilevel Adaptive Aggregation for Markov Chains, with Application to Web Ranking," *SIAM J. Scientific Computing,* vol. 30, pp. 2235-2262, 2008.
[36] Y. Takahashi, "A Lumping Method for Numerical Calculations of Stationary Distributions of Markov Chains," Research Report B-18, Dept. of Information Sciences, Tokyo 23 Inst. of Technology, 1975.
[37] G. Horton and S.T. Leutenegger, "A Multi-Level Solution Algorithm for Steady-State Markov Chains," *Proc. ACM SIGMETRICS,* pp. 191-200, 1994.
[38] S.T. Leutenegger and G. Horton, "On the Utility of the Multi-Level Algorithm for the Solution of Nearly Completely Decomposable Markov Chains," *Numerical Solution of Markov Chains,* W. Stewart, ed., pp. 425-443, Kluwer Publishers, 1995.
[39] U.R. Krieger, "Numerical Solution of Large Finite Markov Chains by Algebraic Multigrid Techniques," *Numerical Solution of Markov Chains,* W. Stewart, ed., pp. 403-424, Kluwer Publishers, 1995.
[40] C. Isensee and G. Horton, "A Multi-Level Method for the Steady State Solution of Markov Chains," *Simulation und Visualisierung,* SCS European Publishing House, 2004.
[41] S. Yu and J. Shi, "Multiclass Spectral Clustering," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* 2003.
[42] R.B. Lehoucq, D.C. Sorensen, and C. Yang, *ARPACK User Guide,* SIAM Publications, 1998.
[43] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 1997.
[44] Code obtained from the URL http://www.autonlab.org/autonweb/2.html, 2009.
[45] J.A. Hartigan and M.A. Wong, "A k-Means Clustering Algorithm," *Applied Statistics,* vol. 28, no. 1, pp. 100-108, 1979.
[46] A. Brandt, "Principles of Systematic Upscaling," *Bridging the Scales in Science and Eng.,* J. Fish, ed., Oxford Univ. Press, 2008.

Dan Kushnir received the BSc degree in computer science from the Hebrew University, Jerusalem, in 2001, the MSc degree in computer science and applied mathematics in 2005, and the PhD degree in applied mathematics from the Weizmann Institute of Science in 2009. He currently holds the J.W. Gibbs Assistant Professor position in the Mathematics Department at Yale University. His current research interests include numerical analysis, harmonic analysis, multigrid methods, and pattern recognition.

Meirav Galun received the BSc degree in mathematics and computer science from Tel-Aviv University in 1985, where she graduated summa cum laude. She received the MSc and PhD degrees in applied mathematics from the Weizmann Institute of Science in 1992 and 1998, respectively, where she accepted an Excellency Award. She is currently a staff scientist associated with the Computer Vision Group in the Applied Mathematics and Computer Science Department at the Weizmann Institute of Science. Her research interests include computer vision, clustering and data analysis, multiscale methods, and applications to biomedical data.



Achi Brandt is a full professor at the Weizmann Institute of Science, where he served in the past as head of the Department of Pure Mathematics (1973-1975), head of the Department of Applied Mathematics (1978-1982), and director of the Gauss Center for Scientific Computation (1992-2003). He is a recipient of the Landau Prize in Mathematics (1978) and the Rothschild Prize in Mathematics (1990). He is a pioneer in research on multiscale computation introducing fundamental computational approaches to diverse fields in mathematics, science, and engineering, including algebraic and partial differential equations, fluid dynamics, integrodifferential equations, integral transforms, quantum mechanics, statistical mechanics, elementary particles molecular and macromolecular dynamic, bioinformatics, global optimizations, graph problems, medical imaging, and multiscale algorithms for various vision tasks, such as surface reconstruction, edge and fiber detection and completion, image segmentation, and recognition processes. He has published more than 150 scientific papers and organized many international workshops. In 2005, he received the highest international prize in scientific computation, the SIAM/ACM Prize in Computational Science and Engineering, cited for pioneering modern multilevel methods, from multigrid solvers for partial differential equations to multiscale techniques for statistical physics and for influencing almost every aspect of contemporary computational science and engineering.