UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Multilevel Algebraic Elliptic Solvers

Tony F. Chan
Petr Vanek

February 1999

CAM Report 99-9

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

Multilevel Algebraic Elliptic Solvers*

Tony F. Chan** and Petr Vaněk***

Department of Mathematics, University of California at Los Angeles, 405 Hilgard Ave, Los Angeles, CA, 90024

Abstract: We survey some of the recent research in developing multilevel algebraic solvers for elliptic problems. A key concept is the design of a hierarchy of coarse spaces and related interpolation operators which together satisfy certain approximation and stability properties to ensure the rapid convergence of the resulting multigrid algorithms. We will discuss smoothed agglomeration methods, harmonic extension methods, and global energy minimization methods for the construction of these coarse spaces and interpolation operators.

1 Introduction

There has been a recent resurgence of interest in algebraic multilevel elliptic solvers. Part of the motivation for considering the *multilevel* approach is that it is the only general method for deriving scalable solvers for large problems. Another reason is that multilevel methods are often naturally parallelizable. The main motivation for considering *algebraic* solvers is ease of use as often the user need only supply the problem in purely matrix-vector form without referring to (often very complicated) grid and other physical and geometric information.

The interest in algebraic multilevel solvers has been heightened by the recent emergence of *unstructured grids* as a powerful and flexible approach in scientific computing for modelling complex geometries. This popularity has led to increased interest in developing fast, efficiently parallelizable elliptic (and some non-elliptic) solvers on such grids. In particular, there has been a lot of research activities in adapting "optimal" multilevel elliptic solvers for structured grids, such as multigrid and domain decomposition methods, to unstructured grids [4]. The absence of a natural coarse grid hierarchy has led to several approaches to developing multilevel methods, ranging from computational geometric approaches for explicitly constructing appropriate coarse grids and the associated intergrid transfer operators, to more abstract agglomeration approaches of constructing appropriate nested or non-nested coarse spaces, together with the associated interpolation operators, over agglomerated coarse grid elements. The resulting methods are often completely *algebraic*, in the sense of not needing to know about geometric information about the underlying

* This research has been supported by NSF grant ASC-972057, Sandia National Laboratory grant LG-4440 and NASA Ames grant NAG2-1238.

** E-mail address: chan@math.ucla.edu

*** E-mail address: pvanek@math.ucla.edu

grids. Thus, in summary, the recent interest in algebraic multilevel methods are due to two factors: the emergence of unstructured grids and the need for parallelizable, scalable and easy-to-use elliptic solvers for increasingly large problems in scientific computing.

In this paper, we'll give a brief survey of some of the key ideas and approaches in this field and also discuss in some more details several approahes that we (with collaborators) have been developing based on energy minimization and smoothed aggregation. The plan of the paper is as follows, after a brief review of the basic multigrid algorithm in Sec. 2, highlighting the essential role of the interpolation operator, we summarize in Sec. 3 the essential desirable properties that the interpolation operator should satisfy in order to lead to rapidly convergent and efficient multilevel algorithms. In Sec. 4, we discuss algorithms which computes *locally* the interpolation operator and in Sec. 5, we discuss methods which compute all of the interpolation operators in a global, coupled, manner.

2 Variational multigrid algorithm

In this section we briefly recall the standard variational (Ritz-Galerkin) coarsening scheme and the multigrid cycle for solving the linear system

$$A\mathbf{x} = \mathbf{b}$$

with symmetric, positive definite matrix A.

Given an $n \times n$ matrix A, we create prolongators I_{l+1}^l and coarse level matrices A_l following this general scheme:

Algorithm 1 (setup) *Initialize* $l = 1$, *set* $n_1 = n$, $A_1 = A$.
Reapeat
- *create* $n_l \times n_{l+1}$ $n_{l+1} < n_l$ *prolongator* I_{l+1}^l,
- *get the Ritz-Galerkin coarse-level matrix*

$$A_{l+1} = (I_{l+1}^l)^T A_l I_{l+1}^l \quad \textit{and set } l \leftarrow l+1 \tag{1}$$

until A_l *is sufficiently small to be treated by a direct solver.*
Set the number of levels $L = l$.

The key step of the above variational coarsening scheme is the construction of prolongators I_{l+1}^l. There are two main issues involved: a) specifying the nonzero structure of prolongators and, b) finding appropriate values of nonzero entries. In a finite element context, the nonzero structure of prolongators corresponds to the supports of coarse-level "macroelements" (shape function supports.) The values of nonzero entries then determine shape functions on given supports.

The purpose of this paper is to clarify the requirements on prolongators that lead to an efficient multilevel solver and present algorithms that strive to match those requirements.

One iteration $\mathbf{x} \leftarrow MG(\mathbf{x}, \mathbf{b})$ of the multigrid algorithm is as follows:

Algorithm 2 (iteration) *Let* $R_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$, $l = 1, \ldots, L-1$ *be given smoother preconditioners and* $\nu, \gamma > 0$ *be a given smoothing and cycle parameter, respectively and* $A \equiv A_1$ *a stiffness matrix. Set* $MG = MG_1$, *where* $MG_l(\cdot,\cdot)$, $l = 1, \ldots, L-1$ *is defined by:*

Pre-smoothing: *Perform* ν *iterations of* $\mathbf{x}^l \leftarrow (I - R_l A_l)\mathbf{x}^l + R_l \mathbf{b}^l$.
Coarse grid correction:
- *Set* $\mathbf{b}^{l+1} = (I_{l+1}^l)^T(\mathbf{b}^l - A_l \mathbf{x}^l)$,
- *if* $l+1 = L$, *solve* $A_{l+1}\mathbf{x}^{l+1} = \mathbf{b}^{l+1}$ *by a direct method, otherwise set* $\mathbf{x}^{l+1} = \mathbf{0}$ *and perform* γ *iterations of* $\mathbf{x}^{l+1} \leftarrow MG_{l+1}(\mathbf{x}^{l+1}, \mathbf{b}^{l+1})$,
- *correct the solution on level* l *by* $\mathbf{x}^l \leftarrow \mathbf{x}^l + I_{l+1}^l \mathbf{x}^{l+1}$.

Post-smoothing: *Perform* ν *iterations of* $\mathbf{x}^l \leftarrow (I - R_l A_l)\mathbf{x}^l + R_l \mathbf{b}^l$.

3 Construction of prolongators - objectives

Following the considerations in [5], this section specifes requirements on prolongators I_{l+1}^l needed to design an efficient multigrid solver.

Let us start the discussion by considering a simple two-level method. It is well-known that commonly used smoothers eliminate efficiently fine-level errors $\mathbf{e}$ which satisfy

$$\frac{\mathbf{e}^T A \mathbf{e}}{\mathbf{e}^T \mathbf{e}} \approx \varrho(A_l). \tag{2}$$

This behavior can be demonstrated on a case of a simple Richardson-type smoothing procedure

$$\mathbf{x} \leftarrow (I - \varrho(A)^{-1} A) + \varrho(A)^{-1}\mathbf{b} \tag{3}$$

(that is, $R_1 = \varrho(A)^{-1} I$ in our notation.) Setting $\hat{\mathbf{x}} = A^{-1}\mathbf{b}$, the smoothing procedure (3) replaces the error $\mathbf{e} = \mathbf{e}(\mathbf{x}) \equiv \mathbf{x} - \hat{\mathbf{x}}$ by a smoothed error $(I - \varrho(A)^{-1}A)\mathbf{e}$. Let us demonstrate that the smoother (3) is efficient in suppressing errors satisfying (2).

Denoting $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$ and $\|\mathbf{x}\| = \langle \mathbf{x}, \mathbf{x} \rangle^{1/2}$, elementary manipulations give

$$\|(I - \varrho(A)^{-1}A)\mathbf{e}\|^2 = \|\mathbf{e}\|^2 - 2\varrho(A)^{-1} \langle A\mathbf{e}, \mathbf{e} \rangle + \varrho(A)^{-2}\|A\mathbf{e}\|^2.$$

Using the estimate $\|A\mathbf{e}\|^2 = \langle A\mathbf{e}, A\mathbf{e} \rangle \leq \varrho(A) \langle A\mathbf{e}, \mathbf{e} \rangle$, we get

$$\|(I - \varrho(A)^{-1}A)\mathbf{e}\|^2 \leq \|\mathbf{e}\|^2 - \frac{1}{\varrho(A)} \langle A\mathbf{e}, \mathbf{e} \rangle = \left(1 - \frac{1}{\varrho(A)} \frac{\langle A\mathbf{e}, \mathbf{e} \rangle}{\|\mathbf{e}\|^2}\right) \|\mathbf{e}\|^2.$$

As one can see from the last estimate, the "efficiency" of the smoother is guided by the ratio on the left-hand side of (2); the convergence speeds up as the left-hand side of (2) approaches its maximum $\varrho(A)$.

Two-level methods strive to split the computational work between a fine-level smoothing and a coarse-level correction. To accomplish this goal succesfully, one has to assure that error components not suppressed by the smoother significantly

can be approximated well by coarse-level vectors $I_2^1\mathbf{v}$. In other words, we need a coarse space Range (I_{l+1}^l) that approximates well fine-level e vectors which satisfy

$$\frac{\langle A\mathbf{e},\mathbf{e}\rangle}{\|\mathbf{e}\|^2} << \varrho(A_l).$$

The smaller the ratio $\langle A\mathbf{e},\mathbf{e}\rangle/\|\mathbf{e}\|^2$ is, the better approximation $I_2^1\mathbf{v}$ is needed. This requirement can be seen in a key assumption of the classical variational two-level method theory [2] (when we divide both sides of the inequality by $\|\mathbf{e}\|^2$):

There is a positive constant C_A such that for every fine level vector e *we can find a coarse-level vector* v *such that*

$$\|\mathbf{e} - I_2^1\mathbf{v}\|^2 \leq \frac{C_A}{\varrho(A)}\langle A\mathbf{e},\mathbf{e}\rangle . \tag{4}$$

Assuming the smoother (3) is used, the convergence theory in [2] gives an $A-$norm rate of convergence estimate ($\mathbf{x}^i$ denotes the result after i iterations)

$$\|\mathbf{e}(\mathbf{x}^{i+1})\|_A \leq (1 - 1/C_A)\|\mathbf{e}(\mathbf{x}^i)\|_A$$

In the general multilevel case, the mechanism is essentially the same. For each level $l < L$ and every vector $\mathbf{u}_l$ on the level l, we need coarse approximation $\mathbf{u}_{l+1}$ such that

$$\|\mathbf{u}_l - I_{l+1}^l\mathbf{u}_{l+1}\|^2 \leq \frac{C_A}{\varrho(A_l)}\langle A_l\mathbf{u}_l,\mathbf{u}_l\rangle. \tag{5}$$

There are two natural ways to satisfy (5), and their combination is needed to accomplish it cheaply:
a) minimizing the left-hand side of (5) (i.e. controlling the approximation properties of the range of I_{l+1}^l),
b) maximizing the right-hand side (making (5) easier to satisfy).
To make the right-hand side of (5) large, we will minimzize $\varrho(A_l)$ subject to certain constraints. Note that the easiest (but useless) way of making $\varrho(A_l)$ small is to multiply the prolongator I_l^{l-1} by a small constant α. Then $A_l \equiv (I_l^{l-1})^T A_{l-1} I_l^{l-1}$ becomes $\alpha^2 A_l$, $\varrho(A_l)$ becomes $\alpha^2\varrho(A_l)$ and the right-hand side of (5) remains unchanged for all vectors $\mathbf{u}_l$. Such a degeneration of a minimization process will be prevented by constraints aimed to a).

We sum up the above considerations in a form of list of requirements on I_{l+1}^l that lead to an efficient multigrid solver.

Zero-energy modes (local kernel) preserving property. Let B^1 be the matrix consisting of columns that generate the local kernel of A. That is,

$$(AB^1)_{ij} = 0$$

for all columns j of B^1 and all degrees of freedom i that are not located on elements with Dirichlet boundary conditions.

We require columns of B^1 (the zero energy modes) to be represented on each level. More precisely, we want prolongators I_{l+1}^l such that there are coarse-level reperesentations $B^2, \ldots, B^L$ of B^1 satisfying (aside from Dirichlet boundary conditions, see Rem. 1.)

$$B^1 = I_2^1 B^2 = \ldots = I_2^1 \ldots I_L^{L-1} B^L \text{ or equivalently, } B^l = I_{l+1}^l B^{l+1} \ \forall l < L. \quad (6)$$

Note that the finest level zero-energy modes B^1 must be supplied as input data. To create prolongators I_{l+1}^1 satisfying (6), we first generate simultaneously I_2^1 and B^2 such that $B^1 = I_2^1 B^2$. Then B^2 is used to get I_3^2 and B^3, etc.

The motivation arise from the need to capture the low-energy vectors by coarse levels. On the theoretical front, (6) together with bounded overlaps of basis-function supports allows to prove approximation properties of the form: *For every* $\mathbf{u}_l$ *there is a coarse level vector* $\mathbf{u}_{l+1}$ *such that*

$$\|\mathbf{u}_l - I_{l+1}^l \mathbf{u}_{l+1}\|^2 \leq C_A(l) \langle A_l \mathbf{u}_l, \mathbf{u}_l \rangle \quad (7)$$

using Bramble-Hilbert lemma type arguments, see [6].

Small spectral radii of coarse-level matrices. Assume B^l is available when constructing I_{l+1}^l and its coarse-level representation B^{l+1} has been chosen. We want

$$\varrho(A_{l+1}) = \varrho\left((I_{l+1}^l)^T A_l I_{l+1}^l\right)$$

to be as small as possible; the requirement (6) prevents the "scaling degenearion" mentioned above.

Note that from (7), the inequality (5) holds with $C_A(l) = C(l)\varrho(A_l)$. Our goal is to create the prolongators I_{l+1}^l so that $C(l)\varrho(A_l) \leq C$ with a small constant C independent of the level l. As the level l increases, coarse level approximation properties deteriorate and $C(l)$ in (7) grows. To guarantee a good convergence rate, this "approximation" loss must be compensated by a decrease of the "maximal energy meassure" $\varrho(A_l)$.

Note 1. (Relaxing (6) near Dirichlet boundary conditions). For nodes that are close to Dirichlet boundary conditions, the objective (6) contradicts to the smoothness requirement (small $\varrho(A_l)$.)

As zero-energy modes do not respect Dirichlet boundary conditions in general, (6) causes the function analogues of coarse-level vectors to be nearly discontinuous (hence high-energetical) close to Dirichlet constraints.

For the above reason, we relax (6) as follows:

$$(B^l)_{i,j} = (I_{l+1}^l B^{l+1})_{ij} \quad \text{for all} \quad i,j \text{ such that } (A_l B^l)_{ij} = 0. \quad (8)$$

In other words, we require (6) only for the degrees of freedom where the column of B^l is a valid local kernel.

Sparsity of coarse-level matrices. The coarse-level matrices A_l should be sparse. More precisely, the number of nonzero entries per column of A_l should be bounded by a small constant independent of the level. The reasons for this

requirement are mostly practical; the excessive fill-in of coarse-level matrices increases the amount of computational work needed both in the setup and the iterations.

In a finite element context, nonzero off-diagonal entries of A_l indicate the overlap of basis-function supports. The assumption on bounded overlaps of supports simplify a convergence analysis, but can be avoided ([7].)

Geometry of coarsening and coupling. Prolonagtors I_{l+1}^l must have only bounded number of nonzeroes per column and the nonzero structure of the columns should follow large entries of A_l.

More specifically, two entries $(I_{l+1}^l)_{ij}$, $(I_{l+1}^l)_{kj}$ of the j–th column of the prolongator are allowed to be nonzeroes only if i, k are two degrees of freedom on the level l that are strongly coupled in the matrix A_l^α, where α is a small integer (eg. 1,2 or 3.)

The motivation for this objective comes again from considerations about underlying (continuous) basis function supports. The bounded number of nonzeroes per column of I_{l+1}^l together with the "distance" constraint (small α) reflects the need to control the meassure and diameter of supports.

The need to follow strong connections is important for solving anisotropic problems, where the coarsening should be performed only in the direction that corresponds to a large coefficient (semicoarsening.)

The need to follow strong connections in the coarsening can be avoided if more sophisticated (colored or line) smoothers are used.

4 Local energy minimizatiom methods

The zero-energy modes preservation constraint (14) is global in a sense that one cannot update two columns of the prolongator indepndently without (potentialy) violating (14). To avoid data dependencies that come with such a globality, two coarsening techniques secribed here use a following princile: *If all the columns of the prolongator are updated (independently) via a multiplication by (a same) linear mapping that behaves like the identity for zero energy modes, the updated prolongator satisfies (14) again.* As those methods do not need explicit means to establish a global communication between the columns, we call them local.

4.1 Smoothed aggregation method

The smoothed aggregation method (proposed by Vaněk in [9, 8] and further developed in [5, 10, 6]), builds the prolongator in the form

$$I_{l+1}^l = S_l P_{l+1}^l, \tag{9}$$

where P_{l+1}^l is a very simple *tentative prolongator* satisfying

$$B^l = P_{l+1}^l B^{l+1} \tag{10}$$

and S_l is a Richardson-type *prolongator smoother* derived from matrix A_l, e.g.,

$$S_l = I - \frac{\omega}{\varrho(A_l)} A_l. \tag{11}$$

Note that on the finest level, the zero-energy modes B^1 must be given. To satisfy (10), we build simultaneously P_2^1 and B^2 so that $B^1 = P_2^1 B^2$. Then using B^2, we construct P_3^2 and B^3, etc. For details, see Alg. 3.

Before specifying P_{l+1}^l and S_l, we briefly discuss their purpose and relationship to requirements on I_{l+1}^l listed in Sec. 3.

The requirement (8) is enforced through (10). Assume a prolongator smoother of the form (11) is being used. Then the smoothed prolongator satisfies the relaxed constraint (8). Indeed, from (10) (setting $\alpha = \omega/\varrho(A_l)$ for convenience),

$$I_{l+1}^l B^{l+1} = (I - \alpha A_l) P_{l+1}^l B^{l+1} = (I - \alpha A_l) B^l = B^l - \alpha A_l B^l,$$

and one can see that the matrices $I_{l+1}^l B^{l+1}$ and B^l differ only in entries where $A_l B^l \neq 0$.

The purpose of the prolongator smoother S_l is to minimize $\varrho(A_{l+1})$. As shown at the beginning of Sec. 3, simple pointwise smoothers eliminate efficiently high-energy errors. The prolongator smoother (11) is an error propagation operator of a Richardson-type iteration. By applying it to the range of P_{l+1}^l, we suppress high-energy vectors, which in turn reduces the "maximal energy measure" $\varrho(A_{l+1})$. Rigorous explanation is given in Remark 2.

The geometrical requirements are respected through a choice of aggregates that determine a nonzero structure of P_{l+1}^l, see Alg. 3.

Note 2. (PROLONGATOR SMOOTHING EFFECT) Consider a tentative prolongator P_{l+1}^l such that $\|P_{l+1}^l \mathbf{x}\| = \|\mathbf{x}\|$ for all $\mathbf{x} \in \mathbb{R}^{n_{l+1}}$. Note that Alg. 3 creates P_{l+1}^l satisfying above isometry, see [6]. We will show that if ω in (11) is chosen properly, the spectral radius $\varrho(A_{l+1})$ is at least 9 times smaller then $\varrho(A_l)$.

As S_l is a polynomial in A_l, S_l is symmetric and commutes with A_l. Hence (1) and (9) gives $A_{l+1} = (S_l P_{l+1}^l)^T A_l (S_l P_{l+1}^l) = (P_{l+1}^l)^T S_l^2 A_l P_{l+1}^l$. From this identity we have the estimate

$$\varrho(A_{l+1}) = \max_{\mathbf{x} \in \mathbb{R}^{n_{l+1}}} \frac{\langle A_{l+1}\mathbf{x}, \mathbf{x}\rangle}{\|\mathbf{x}\|^2} = \max_{\mathbf{x} \in \mathbb{R}^{n_{l+1}}} \frac{\langle S_l^2 A_l P_{l+1}^l \mathbf{x}, P_{l+1}^l \mathbf{x}\rangle}{\|P_{l+1}^l \mathbf{x}\|^2} \leq \varrho(S_l^2 A_l).$$

The spectral mapping theorem applied to the last estimate yields (σ denotes the spectrum)

$$\varrho(S_l^2 A_l) = \max_{t \in \sigma(A_l)} \left(1 - \frac{\omega}{\varrho(A_l)} t\right)^2 t \leq \max_{t \in [0, \varrho(A_l)]} \left(1 - \frac{\omega}{\varrho(A_l)} t\right)^2 t = \varrho(A_l) \cdot \max_{\tau \in [0,1]} (1 - \omega\tau)^2 \tau.$$

It is easy to verify that for $\omega = 4/3$, the expression on the right-hand side of the last estimate equals to $\frac{1}{9}\varrho(A_l)$.

Based on Rem. 2, we use a prolongator smoother

$$S_l = I - \frac{4}{3}\bar{\varrho}(A_l)^{-1}A_l, \tag{12}$$

where $\bar{\varrho}(A_l)$ is an available upper bound of $\varrho(A_l)$.

It remains to specify the tentative prolongator P_{l+1}^l.

Assume B^l is available and denote the number of its columns by r. Our goal is to create the tentative prolongator P_{l+1}^l and the coarse-level representation B^{l+1} of B^l satisfying (10).

Our construction is based on the supernodes aggregation concept. On each level, degrees of freedom are organized in small disjoint clusters called supernodes. On the finest level, these clusters have to be specified, e.g., as the sets of degrees of freedom associated with the finite element vertices, the coarse level supernodes are then created by our aggregation algorithm. The prolongator P_{l+1}^l is constructed from a given system of aggregates $\{\mathcal{A}_i^l\}_{i=1}^{N_l}$ that forms a disjoint covering of level l supernodes. A simple greedy algorithm for generating aggregates based on the structure of the matrix A_l is given in [5]. The property (10) is enforced aggregate by aggregate; columns of P_{l+1}^l associated with the aggregate $\mathcal{A}_i^l$ are formed by orthonormalized restrictions of the columns of B^l onto the aggregate $\mathcal{A}_i^l$. For each aggregate, such a construction gives rise to r degrees of freedom on the coarse level forming a coarse level supernode.

The detailed algorithm follows. For ease of presentation, we assume that the fine level supernodes are numbered by consecutive numbers within each aggregate. This assumption can be easily avoided by renumbering.

Algorithm 3 *For the given system of aggregates $\{\mathcal{A}_i^l\}_{i=1}^{N_l}$ and the $n_l \times r$ matrix B^l satisfying $P_l^1 B^l = B^1$, we create a prolongator P_{l+1}^l, a matrix B^{l+1} satisfying (10) and supernodes on level $l+1$ as follows:*

1. *Let d_i denote the number of degrees of freedom associated with aggregate $\mathcal{A}_i^l$. Partition the $n_l \times r$ matrix B^l into blocks B_i^l of size $d_i \times r$, $i = 1, \ldots, N_l$, each corresponding to the set of degrees of freedom on an aggregate $\mathcal{A}_i^l$ (see Fig. 1).*
2. *Decompose $B_i^l = Q_i^l R_i^l$, where Q_i^l is an $d_i \times r$ orthogonal matrix, and R_i^l is an $r \times r$ upper triangular matrix.*
3. *Using the blocks Q_i^l, $i = 1, \ldots, N_l$, create the prolongator P_{l+1}^l as shown by Fig. 1.*
4. *Create B^{l+1} consisting of the blocks R_i^l, $i = 1, \ldots, N_l$, (see Fig. 1.)*
5. *For each aggregate $\mathcal{A}_i^l$, the coarsening gives rise to r degrees of freedom on the coarse level (the $i-$th block column of P_{l+1}^l). These degrees of freedom define the $i-$th coarse level supernode.*

The abstract convergence bounds for smoothed aggregation methods are given in [6]. Under weak assumptions on the aggregates it is shown that each iteration reduces the $A-$norm of the error at least by a factor $1 - C/L^3$, where L is the number of levels and C is a constant independent of L. When applied to a linear system obtained by discretizing a second order elliptic problem, C is independent of the meshsize.

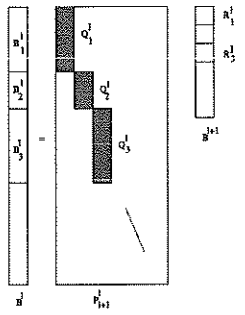


Fig. 1. The construction of a tentative prolongator P_{l+1}^l and B^{l+1}

4.2 Macroelement based coarsening technique

The method outlined here creates – by algebraic means – coarse levels that are very close to finite element spaces, while overcoming their geometrical limitations. For more details, see [3].

We restrict our brief explanation to a case of coarsening the finest level finite element stifness matrix A. As the method reproduces essential properties of finite element spaces on the next level, the further coarsening is analogous. The main steps follow:

1. Using a nodal adjacency information contained in the stiffness matrix, the elements are clustered into small nonoverlapping groups called coarse-level *macroelements*. Then the coarse points (macroelement vertices) are specified, see Fig.2.

2. The basis functions (columns of I_2^1) satisfying (14) are first defined only on the interfaces (boundaries of macroelements). Efficient ways of constructing the interface basis are given in [3].

3. The basis is then extended inside the element *harmonically*, i.e. by setting macroelement interior degrees of freedom so that the energy (evaluated over the macroelement) is minimal. This operation is performed by solving Dirichlet problems corresponding to the macroelemnts.

The harmonic extension enforces the smoothness of the coarse level basis. Further, since the harmonic extension returns the vector with minimal energy, the harmonic extension of a zero-energy mode restricted to the interface is the zero-energy mode. Hence (14) is satisfied on all macroelements where B^1 is a valid local kernel.

5 Global energy minimization method

This section describes a method of constructing prolongators by minimizing the trace of the coarse-level matrix. In a finite element context, diagonal entries of coarse-level matrices correspond to energies of coarse-space basis functions and the trace minimization can be interpreted a minimization of their sum.

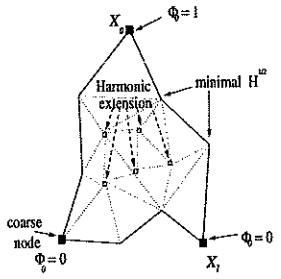


Fig. 2. The macroelement

Using the requirements on prolongators formulated in [5], the energy minimization method has been developed independently in [12] and [11]. This section follows the slightly more general variant proposed in [11].

Based on the objectives given in Sec. 3, the most straightforward way of constructing the prolongator I_{l+1}^l is the following:

Assume B^l has been cretated during the construction of $I_2^1, \ldots I_l^{l-1}$.

1. Define the nonzero structure of I_{l+1}^l using the geometrical objectives in Sec. 3.
2. Choose the coarse-level representation B^{l+1} of B^l.
3. Find I_{l+1}^l of a given nonzero structure minimizing $\varrho((I_{l+1}^l)^T A_l I_{l+1}^l)$ subject to (8).

Direct minimization of a functional by an iterative method usually requires the calculation of its value and steepest descent direction in every iteration. Such calculations are very expensive for a spectral radius objective function.

On the other hand, it is very cheap to evaluate a trace of the matrix (sum of eigenvalues=sum of diagonal entries.) As the contribution of large eigenvalues prevails in their sum, the trace minimization is aimed mainly to the higher part of the spectrum. Further, as demonstrated at the beginning of Sec. 3, simple iterative methods tend to eliminate "energetically reach" error components within a few iterations. For the above heuristical reasons, one can expect to get a good prolongator by minimizing the objective function $J(\cdot) : I_{l+1}^l \mapsto \mathrm{tr}((I_{l+1}^l)^T A_l I_{l+1}^l)$.

Let Z be a 0/1 matrix defining a nonzero structure of I_{l+1}^l and the symbol $*$ denotes the matrix multiplication entry by entry ($Z * I_{l+1}^l = I_{l+1}^l$ means that I_{l+1}^l has nonzeroes only in positions i, j marked by a flag $Z_{ij} = 1$.)

Using this notation, the minimization problem can be formulated as follows:

$$\text{find } I_{l+1}^l \text{ minimizing } J(\cdot) : I_{l+1}^l \mapsto \mathrm{tr}((I_{l+1}^l)^T A_l I_{l+1}^l) \tag{13}$$

subject to

$$I_{l+1}^l B^{l+1} = B_{mod}^l \text{ and } Z * I_{l+1}^l = I_{l+1}^l \tag{14}$$

Here, B^l_{mod} is a (properly) modified matrix B^l such that

$$(B^l - B^l_{mod})_{ij} \neq 0 \text{ only if } (AB^{l+1})_{ij} \neq 0.$$

Note that if the prolongator satisfies the above constraints, it will also satisfy (8).

Following the considerations in Remark 1, we modify the zero-energy modes B^{l+1} to eliminate "discontinuities" near Dirichlet boundary conditions. The natural way consists in replacing all columns $\mathbf{b}$ of B^l by modified columns $\mathbf{b}'$

$$\text{minimizing } \langle A_l\mathbf{b}', \mathbf{b}'\rangle \text{ subject to } \mathbf{b}_i = \mathbf{b}'_i \text{ for all } i : (A_l\mathbf{b})_i = 0.$$

The above procedure typically requires to solve a well-conditioned system of a modest size. Few iterations of a simple pointwise iterative method ussually gives a satisfactory approximation.

In what follows we omit the subscript in B^l_{mod}; the original matrix B^l is not needed anymore.

To solve (13), we use a projected gradient type iterative method that requires an initial guess I^l_{l+1} and B^{l+1} satisfying (14). Those data can be obtained by the aggregation Algorithm 3 or by smoothed aggregations.

Once the initial guess I^l_{l+1} satisfying (14) is available, we search for correction C such that $I^l_{l+1} - C$ solves the minimization problem (13) subject to (14) (with $I^l_{l+1} - C$ in the place of I^l_{l+1}.)

Obviously, *admissible corrections* form a subspace of the space $\mathbf{M}^{n_l \times n_{l+1}}$ of all $n_l \times n_{l+1}$ matrices,

$$\mathbf{U} = \{U \in \mathbf{M}^{n_l \times n_{l+1}} : \ UB^{l+1} = 0 \text{ and } Z * U = U\} \tag{15}$$

and their columns U_{i*}, $U \in \mathbf{U}$, form linear spaces

$$\mathbf{U}_i = \{U \in \mathbf{M}^{1 \times n_{l+1}} : \ UB^{l+1} = 0 \text{ and } Z_{i*} * U = U\}. \tag{16}$$

Here, Z_{i*} denotes the $i-$th row of Z.

First, we calculate the gradient of $J(I^l_{l+1})$ [1] in the space $\mathbf{Z} \equiv \{U \in \mathbf{M}^{n_l \times n_{l+1}} : Z * U = U\}$ by

$$G = Z * (A_l I^l_{l+1}),$$

see [11], Lemma 4.1. In general, G is not an admissible correction as GB^{l+1} is not guaranteed to be zero. We construct its (best) admissible approximation

$$G' \in \mathbf{U} \text{ such that tr } \left((G - G')(G - G')^T\right) \text{ is minimal.}$$

In other words, we project $G \in \mathbf{Z}$ onto $\mathbf{U}$ using a projection orthogonal in Hilbert space equipped with the Frobenius norm $\|\cdot\|_{fr} : U \mapsto \text{tr}(UU^T)^{1/2}$. Note that the Frobenius norm is related to Euclidean norms of rows U_{i*} by

$$\|U\|^2_{fr} \equiv \text{tr}(UU^T) = \sum_{j=1}^{n_l} \|U_{i*}\|^2. \tag{17}$$

[1] That is $G \in \mathbf{Z}$ maximizing $\frac{d}{dt}\text{tr}((I^l_{l+1} + tG)^T A_l (I^l_{l+1} + tG))$ at $t = 0$.

The "orthogonality" of the columns in (17) together with the constraint (16) allows to implement the projection operator row by row. Indeed, one can easily see that to get

$$G' \in \mathbf{U} \text{ minimizing } \mathrm{tr}\left((G-G')(G-G')^T\right) = \sum_{i=1}^{n_l} \|G_{i*} - G'_{i*}\|^2, \tag{18}$$

it is sufficient to find (independently) its rows

$$G'_{i*} \in \mathbf{U}_i \text{ minimizing } \|G_{i*} - G'_{i*}\|, \quad i = 1, \ldots, n_l. \tag{19}$$

To do so, we further reformulate the row constraints (16) using the fact that $G \in \mathbf{Z}$,

$$(R^i)^T G_{i*}^T = 0, \quad \text{where } R^i_{k*} = \begin{cases} B^{l+1}_{k*} \text{(copy the line)} & \text{if } Z_{ik} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

The Euclidean orthogonal projection of a column vector $\mathbf{x} \in \mathbb{R}^n$ onto the nullspace of a matrix R^T, $R \in \mathbf{M}^{n \times m}$, is given by the well-known formula

$$\mathbf{x} \mapsto [I - R(R^T R)^+ (R)^T]\mathbf{x}, \tag{21}$$

where $+$ denotes a pseudoinverse.

Using the equivalence of the minimization problems (18) and (19), the row constraints in the form (20) and the formula (21), the projection of a gradient can be performed as follows:

Algorithm 4 (Projection of a gradient) *For given matrix $G \in \mathbf{Z}$, construct $G' = PG \in \mathbf{U}$ minimizing (18) as follows:*
For each row G_{i*} of G, $i = 1, \ldots, n_l$, do
- $R = R_i$, where R_i is given by (20),
- $\mathbf{y} = [I - R((R)^T R)^+ (R)^T](G_{i*})^T$,
- create the $i-th$ row $G'_{*i} = \mathbf{y}^T$
end for.

Now we are ready to write down the minimization algorithm:

Algorithm 5 (Energy minimization - iteration) *Given a matrix A_l, initial guess I^l_{l+1}, (post-processed) zero energy modes B^l, their coarse-level representation B^{l+1} and the nonzero structure Z satisfying (14), find I^l_{l+1} minimizing $J(.)$ subject to (14) as follows:*
Repeat
- Compute the gradient of $J(I^l_{l+1})$ in the space $\mathbf{Z}$ by $G = Z * (A_l I^l_{l+1})$,
- Get the projected gradient $G' = PG \in \mathbf{U}$ using Alg. 4.
- update $I^l_{l+1} \leftarrow I^l_{l+1} - \alpha G'$, where $\alpha \in (0, 2/\varrho(A_l))$ is a given parameter.
until convergence.

Note that $(R_i)^T R_i$ is an $r \times r$ matrix, where r is the number of columns of B^1 ($r = 1$ for scalar elliptic problems, 3 for planar elasticity, 6 for solids, plates and shells.) Hence its pseudoinverse is not expensive.

6 Numerical experiments

The experiments we run on 4 and 15 R10000 processors of a 16-processor SGI Origin/2000. The results of the experiments are presented in Tab. 6. The first line in the table corresponds to the smoothed aggregation technique with a smoother of degree 1. The lines 2–6 demonstrate the effect of the additional energy minimization. In those experiments, the smoothed prolongator has been used as an initial guess and its nonzero structure has been used for defining Z in (14).

In all the experiments below, the method is used as a preconditioner for the conjugate gradient method, and the stopping criterion used was

$$\|\mathbf{r}^i\|_B \leq (10^{-5}/\sqrt{\mathbf{cond}(B,A)} \cdot \|\mathbf{r}^0\|_B$$

where B is the preconditioner, $\mathbf{r}^i$ denotes the residual after i steps of the iteration, and $\mathrm{cond}(B,A)$ is a condition number estimate computed at run time.

solid, (see Fig. 3) 75,174 dofs, 4 CPUs.					solid, 407,277 dofs, 15 CPUs.			
min st	setup/iter/tot time [s]	num it.	cond est	tim/tim sm aggr	setup/iter/tot time [s]	num it.	cond est	tim/tim sm aggr
1	8.1/5.3/13.4	8	2.58	1.00	12.7/36.3/49.1	13	6.16	1.00
2	8.6/5.3/13.9	8	2.55	1.03	15.4/34.0/49.4	12	5.04	1.01
3	8.8/5.4/14.2	8	2.50	1.06	18.1/32.0/50.1	11	4.45	1.02
4	8.8/4.7/13.7	7	2.20	1.02	20.2/31.1/51.4	11	4.23	1.04
5	9.0/4.7/13.8	7	2.21	1.03	22.4/28.4/50.8	10	4.00	1.03
6	9.5/4.7/14.3	7	2.17	1.06	24.9/28.5/53.5	10	3.90	1.09

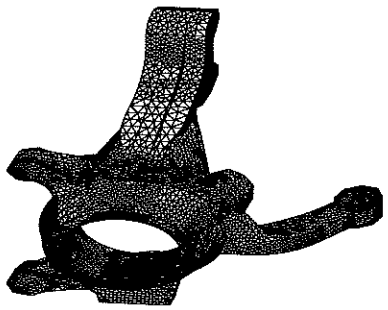


Fig. 3. The mesh of an automobile steering knuckle. Courtesy of Charbel Farhat, University of Colorado at Boulder.)

References

1. J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.
2. A. BRANDT, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.
3. T. F. CHAN, J, XU, AND L. ZIKATANOV, *An agglomeration multigrid for unstructured meshes.*, In: Domain Decomposition Methods 10, (Proceedings of the tenth international conference on domain decomposition methods) Mandel, Farhat, Cai Eds., AMS 1998.
4. T. F. CHAN, S. GO, AND L. ZIKATANOV, *Lecture Notes on Multilevel Methods for Elliptic Problems on Unstructured Grids*, UCLA CAM Report 97-11, March 1997. Lectures notes for the lecture series "Computational Fluid Dynamics", von Karman Inst., Belgium, March 3-7, 1997. An abridged version has been published as CAM Report 97-36, August, 1997 and appeared in "Computational Fluid Dynamics Review 1997", Hafez and Oshima (eds.), Wiley.
5. P. VANĚK, J. MANDEL, AND M. BREZINA, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.
6. P. VANĚK, M. BREZINA, AND J. MANDEL, *Convergence of Algebraic Multigrid Based on Smoothed Aggregation*, Submitted to Num. Math.
7. P. VANĚK, A. JANKA, AND H. GUILLARD, *Convergence of Petrov-Galerkin Smoothed Aggregation Method*, To appear.
8. P. VANĚK, *Fast multigrid solver.* Applications of Mathematics, to appear.
9. *Acceleration of convergence of a two-level algorithm by smoothing trarnsfer operator*, Applications of Mathematics, 37 (1992), pp. 265–274.
10. P. VANĚK, M. BREZINA, AND R. TEZAUR, *Two-Grid Method for Linear Elasticity on Unstructured Meshes*, To appear in SIAM J. Sci. Comp.
11. J. MANDEL, M. BREZINA, AND P. VANĚK, *Energy Optimization of Algebraic Multigrid Bases*, To appear in Computing
12. W. L. WAN, T. F. CHAN, AND B. SMITH, *An energy-minimizing interpolation for robust multigrid methods*, UCLA CAM Report 98-6, Department of Mathematics, UCLA, February 1998.

4.2 Single and Multiple Vector Iterations

contributed by Ming Gu

The single and multiple vector iteration methods for the Hermitian eigenproblem in Section 4.3 can also be used to solve the non-Hermitian eigenproblem, in many cases with similar convergence properties. Although these methods are in general not as competitive as the other methods to be covered in later sections, they are a good choice for those who only want to found a few extreme eigenvalues with a very simple method. One exception seems to be the Rayleigh Quotient Iteration, which is converges much more slowly than its Hermitian counterpart.

4.2.1 Power Method

The power method can be used to solve the non-Hermitian eigenproblem without any apparent change.

ALGORITHM 4.1 : **The Power Method for the computation of $\lambda_{max}(A)$**

Input: *A device to compute Ax for arbitrary x, a starting vector v, and a tolerance ϵ.*
Output: *An approximate eigenpair (θ, z) of the largest eigenvalue λ_{max} and its corresponding eigenvector and satisfies $\|Az - \theta z\| \leq \epsilon$.*

(1) *compute* $z := v/||v||_2$
for $k = 2, 3, 4, ...$
(2) *Compute* $y := Az$ *and* $\theta := z^* y$.
(3) *Stop if* $\|y - \theta z\|_2 \leq \epsilon$
(4) *Else* $z := y/\|y\|_2$.

Under conditions similar to those in the Hermitian case, the power method converges to $\lambda_{max}(A)$, the largest eigenvalue in magnitude. The convergence rate also depends on the ratio $|\lambda_2/\lambda_{max}|$, where λ_2 is the second largest eigenvalue of A in magnitude. For detailed discussions on the power method, see Demmel [3, Ch. 4], Golub and Van Loan [5], and Parlett [7].

4.2.2 Inverse Iteration

Inverse iteration can also be used to solve the non-Hermitian eigenproblem without any apparent change.

ALGORITHM 4.2 : **The Inverse Iteration for the computing an eigenvalue closest to σ**

Input: *A device to compute $(A - \sigma)^{-1} x$ for arbitrary x, a starting vector v, and a tolerance ϵ.*
Output: *the approximate eigenpair (θ, z) of the eigenvalue closest to σ and its corresponding eigenvector and satisfies $\|Az - \theta z\| \leq \epsilon$.*

(1) *compute* $z := v/||v||_2$
for $k = 2, ...$
(2) *Compute* $y := (A - \sigma)^{-1} z$, $z := y/\|y\|_2$, $w := Az$, *and* $\theta := z^* w$
(3) *Stop if* $\|w - \theta z\|_2 \leq \epsilon$

ALGORITHM 4.4 : **Subspace Iteration with Projection and Deflation**

Input: *A device to compute $A\,X$ for arbitrary matrix X, a starting matrix V, an initial iteration parameter iter, and a tolerance ϵ.*

Output: *nev dominant eigenvalues and their corresponding schur vectors.*

(1) *QR-factorize $Z\,R := V$*
while $j \leq nev$ **do**
(2) *Compute $\widehat{Y} := [q_1, q_2, \ldots, q_{j-1}, A^{iter}\,Z]$.*
(3) *Orthonormalize the column vectors of $\widehat{Y}$ (starting at column j) into Z.*
(4) *Compute $\Theta := Z^*\,A\,Z$ and the Schur form $\Theta = Q\,\Lambda\,Q^*$ with nearly equimodular eigenvalues grouped in diagonal blocks in Λ.*
(5) **for** *diagonal blocks $k = 1, 2, \ldots$ in Λ* **do**
(6) *Let Q_k and Λ_k be the corresponding columns in Q and Λ, respectively.*
(7) **if** $\|A\,(Z\,Q_k) - (Z\,Q_k)\,\Lambda_k\| \leq \epsilon$ **then**
(8) *Append $Z\,Q_k$ to $[q_1, q_2, \ldots, q_{j-1}]$, set $j = j + \#$ of columns in $Z\,Q_k$.*
(9) **else** *break.*
(10) *Choose a new iteration parameter iter.*

We now describe some implementation details.

1. The initial starting matrix V should be constructed to be dominant in eigenvector directions of interest to accelerate convergence. In case no such information is known *a priori*, a random matrix is as good a choice as any other.

2. The convergence criterion in the above algorithm checks for convergence only for groups of eigenvalues that have nearly the same modulus. The diagonal blocks in step (5) are ordered from top to bottom, with block 1 at the top of Λ. In step (9), convergence testing is stopped as soon as the first block of eigenvalues in Λ fail to converge.

3. The iteration parameter *iter* should be chosen to minimize orthonormalization cost while maintaining a reasonable amount of numerical accuracy.

If eigenvalues near a shift σ are desired, and a factorization of $A - \sigma\,I$ can be easily obtained, then one can apply the above algorithm to $(A - \sigma\,I)^{-1}$. The eigenvalues near σ will converge fast.

One can also use polynomial acceleration to speed up the computation by replacing the power A^{iter} by a polynomial $T_m[(A - \sigma I)/\rho]$ in which T_m is the Chebyshev polynomial of the first kind of degree m.

Much of the material in this section is drawn from Bai and Stewart [1], Demmel [3], Golub and Van Loan [5], and Saad [8]. For further discussion on subspace iteration, The reader is refered to Stewart [9] and Wilkinson [13].

4.2.5 Software Availability

There are several pieces of software available for subspace iteration. EB12 is a routine by Duff and Scott on subspace iteration [4]. It is part of the Harwell Subroutine Library. Given a real unsymmetric matrix, this routine computes the r eigenvalues that have the most positive real part, the most negative real part, or the largest modulus, with the option of returning the associated eigenvectors. It can also be used to compute other eigenvalues.

4.3 Single and Multiple Vector Iterations

contributed by Ming Gu

In this section, we discuss the power method and its various variations. These methods are very simple to implement and often find a few extreme eigenvalues quickly. They also serve as a motivation for the more sophisticated methods to be discussed later.

4.3.1 Power Method

The *power method* is the simplest iterative method. Under some mild assumptions it finds the largest eigenvalue of a given matrix A in absolute value, and a corresponding eigenvector.

ALGORITHM 4.5 : **The Power Method for the computation of $\lambda_{\max}(A)$**

Input: *a device to compute Ax for arbitrary x, a starting vector v and a tolerance ϵ.*
Output: *the approximate eigenpair (θ, z) of the largest eigenvalue λ_{max} and its corresponding eigenvector which satisfies $\|Az - \theta z\| \leq \epsilon$.*

(1) *compute $z := v/\|v\|_2$ for the starting vector v*
for $k = 2, \ldots$
(2) *Compute $y := Az$ and $\theta := z^T y$.*
(3) *Stop if $\|y - \theta z\|_2 \leq \epsilon$*
(4) *Else $z := y/\|y\|_2$.*

Let x_1 be the eigenvector corresponding to $\lambda_1 = \lambda_{\max}(A)$. The *angle* θ between x_1 and v is defined by the relation

$$\cos\theta = \frac{v^T x_1}{\|v\|_2 \, \|x_1\|_2}.$$

If the starting vector v and the eigenvectorx_1 are perpendicular to each other, then $\cos\theta = 0$. In this case the power method does not converge.

On the other hand, if $\cos\theta \neq 0$, the power method generates a sequence of vectors that become increasingly parallel to x_1. This condition on θ is true with very high probability if v is chosen at random. The convergence rate of the power method depends on $|\lambda_2/\lambda_{\max}|$, where λ_2 is the second largest eigenvalue of A in magnitude. This ratio is generally smaller than 1, allowing adequate convergence. But there are cases this ratio can be very close to 1, causing very slow converence. For detailed discussions on the power method, see Demmel [3, Ch. 4], Golub and Van Loan [5], and Parlett [7].

4.3.2 Inverse Iteration

The drawbacks of the power method can be partially overcome by applying the power method to $(A - \sigma I)^{-1}$ instead of A, where σ is called a *shift*. This will let the method converge to the eigenvalue closest to σ, rather than just $\lambda_{\max}$. This method is called *inverse iteration*, or the *inverse power method.*

ALGORITHM 4.7 : **The RQI for the computation of an eigenvalue**

Input: *a device to compute* $(A - \sigma I)^{-1}\, x$ *for arbitrary* x *and* σ*, a starting vector* v*, and a tolerance* ϵ*.*
Output: *an approximate eigenpair* (θ, z) *that satisfies* $\|Az - \theta z\| \leq \epsilon$*.*

(1) *compute* $z := v/||v||_2$ *and* $\theta := \rho(z)$ *for the starting vector* v
for $k = 2, ...$
(2) *Compute* $y := (A - \theta I)^{-1}\, z$,
$z := y/\|y\|_2$,
$w := A\, z$,
$\theta := z^T\, w$
(3) *Stop if* $\|w - \theta\, z\|_2 \leq \epsilon$

Inverse iteration can be thought of as a way to improve an approximate eigenvalue or eigenvector; but the RQI is a way to find an eigenvalue/eigenvector starting from scratch. Inverse iteration converges linearly; but RQI usually converges cubically. However, it is not obvious as to how to choose the starting vector v to make RQI converge to any particular eigenvalue/eigenvector pair. For example, the RQI can converge to an eigenvalue which is not the closest to $\rho(v)$ and to an eigenvector which is not closest to v. Furthermore, there is the tiny but nasty possibility that it may not converge to an eigenvalue/eigenvector pair. RQI is more expensive than inverse iteration, requiring a factorization of $A - \theta I$ at every iteration. Hence RQI is practical only if such factorizations can be cheaply obtained at every iteration. See Parlett [7] for more details.

4.3.4 Orthogonal Iteration

Another important improvement over the power method permits us to compute a $p > 1$ dimensional invariant subspace, rather than one eigenvector at a time. It is called *Orthogonal Iteration* (and sometimes *Subspace Iteration* or *Simultaneous Iteration*).

ALGORITHM 4.8 : **Simple Subspace Iteration**

Input: *a device to compute* $A\,X$ *for arbitrary matrix* X*, a* $n \times p$*starting matrix* V*, and a tolerance* ϵ*.*
Output: *an approximate invariant subspace spanned by* Z*.*

(1) *QR-factorize* $Z\, R := V$ *for the starting matrix* V
for $k = 2, ...$
(2) *Compute* $Y := A\, Z$ *and* $H := Z^T\, Y$.
(3) *Stop if* $\|Y - Z\, H\|_2 \leq \epsilon$
(4) *Else QR-factorize* $Z\, R := Y$.

This algorithm is a straightforward generalization of the power method. Step (4) is a normalization process that is similar to the normalization used in the power method. There are other normalizations that can also be used. An important observation is that the subspace spanned by the columns in the Z matrix at the k-th iterate is the same as that spanned by columns of $A^k V$.

Locking Because of the different rates of convergence of each of the approximate eigenvalues computed by the subspace iteration, it is a common practice to extract them one at a time and perform a form of deflation. Thus, as soon as the first eigenvector has converged there is no need to continue to multiply it by A in the subsequent iterations. Indeed we can freeze this vector and work only with the vectors $z_2, \ldots, z_m$. However, we will still need to perform the subsequent orthogonalizations with respect to the frozen vector z_1 whenever such orthogonalizations are needed. The term used for this strategy is *locking* . It was introduced by Jennings and Stewart [10].

The following algorithm describes a practical subspace iteration with deflation (locking) for computing the nev dominant eigenvalues.

ALGORITHM 4.10 : **Subspace Iteration with Projection and Deflation**

Input: *a device to compute AX for arbitrary matrix X, a starting matrix V, an initial iteration parameter iter, and a tolerance ϵ.*

Output: *nev dominant eigenvalues and their corresponding eigenvectors in Z.*

(1) *QR-factorize $Z\,R := V$ for the starting V*
while $j \leq$ nev **do**
(2) *Compute $\widehat{Y} := [Z_j,\ A^{iter} Z_{p-j}]$.*
(3) *Orthonormalize $\widehat{Y} = ZR$, (j first columns will be invariant)*
(4) *Compute $H := Z_{p-j}^T\, A\, Z_{p-j}$ and the eigendecomposition $H = Q\,\Theta\,Q^T$*
(5) *Test the diagonal entries in Θ for eigenvalue convergence.*
(6) *Compute $Z := [Z_j\ Z_{p-j}Q]$.*
(7) *Set $j := j + i_{conv}$, where i_{conv} is the number of newly converged eigenvalues..*
(8) *Choose a new iteration parameter iter.*

Preconditioning Preconditioning is especially important for subspace iteration, since the unpreconditioned iteration may be unacceptably slow in some cases. Although we will cover preconditioning in more detail in Chapter VIII, we would like to mention here the main ideas used to precondition the subspace iteration.

- Shift-and-invert. This consists of working with the matrix $(A - \sigma I)^{-1}$ instead of A. The eigenvalues near σ will converge fast.
- Polynomial acceleration. The standard method used is to replace the power A^{iter} in the usual subspace iteration algorithm by a polynomial $T_m[(A - \sigma I)/\rho]$ in which T_m is the Chebyshev polynomial of the first kind of degree m. It gives faster convergence to eigenvalues outside the interval $[\sigma - \rho, \sigma + \rho]$ but no convergence to eigenvalues inside this interval.

With either type of preconditioning subspace iteration may be a reasonably efficient method that has the advantage of being easy to code and understand. Some of the methods to be seen in the rest of this Chapter are often preferred however, because they give convergence using fewer matrix vector multiplies for each eigenvalue.

For more discussion on subspace iteration, reader is referred to the review by Lehoucq and Scott [6] as well as recent books by Chatelin [2] and Saad [8].

4.3.7 Software Availability

There are several pieces of software available for subspace iteration. EA12 is a routine by Duff and Scott on symmetric subspace iteration [4]. It is part of the Harwell Subroutine Library. Given a real

Bibliography

[1] Z. Bai and G.W. Stewart. Algorithm 776. SRRIT — A FORTRAN subroutine to calculate the dominant invariant subspaces of a nonsymmetric matrix. *ACM Trans. Math. Softw.*, 23:494–513, 1998.

[2] F. Chatelin. *Eigenvalues of matrices.* John Wiley and Sons, 1993.

[3] J. Demmel. *Applied Numerical Linear Algebra.* SIAM, Philadelphia, PA, 1997.

[4] I.S. Duff and J.A. Scott. Computing selected eigenvalues of large sparse unsymmetric matrices using subspace iteration. *ACM Trans. Math. Softw.*, 19:137–159, 1993.

[5] G. Golub and C. Van Loan. *Matrix computations.* The Johns Hopkins University Press, 3rd edition, 1996.

[6] R.B. Lehoucq and J.A. Scott. An evaluation of software for computing eigenvalues of sparse nonsymmet ric matrices. Technical Report MCS-P547-1195, Argonne National Laboratory, Argonne, IL, USA, 1995. Submitted to ACM trans. Math. Softw.

[7] B.N. Parlett. *The symmetric eigenvalue problem.* Prentice Hall series in computational mathematics. Prentice-Hall, 1980.

[8] Y. Saad. *Numerical methods for large eigenvalue problems.* Halsted Press, Div. of John Wiley & Sons, Inc., New York, 1992.

[9] G.W. Stewart. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices. *Numer. Math.*, 25:123–136, 1976.

[10] W.J. Stewart and A. Jennings. Algorithm 570: LOPSI a simultaneous iteration method for real matrices. *ACM Trans. Math. Softw.*, 7:230–232, 1981.

[11] W.J. Stewart and A. Jennings. A simultaneous iteration algorithm for real matrices. *ACM Trans. Math. Softw.*, 7:184–198, 1981.

[12] J. H. Wilkinson and C. Reinsch. *Handbook for Automatic Computation. Vol. II Linear Algebra.* Springer, New York, 1971.

[13] J.H. Wilkinson. *The algebraic eigenvalue problem.* Clarendon Press, Oxford, UK, 1965.

				(200		)	

¢ ´ ¯ † / p ¢ ‰ fl Ø / A ¯ m [g p \ R
″

A b v ‚ ~ ¢ • B ¢ ´ ¯ ¯ V « \ O [
h A b v • Ø – ˘ ‟ ¯ « • ' A X ¯ S ~ ¤
g ¢ ¢ ‰ fl • B f
R æ
h A b v ‚ ¯ • B – Œ ¢ E V F A * ¯ ¨
¯ ″ A [‰ T [r X E T | [g ¯ • B * f [^ N G X
g † , F
@ @

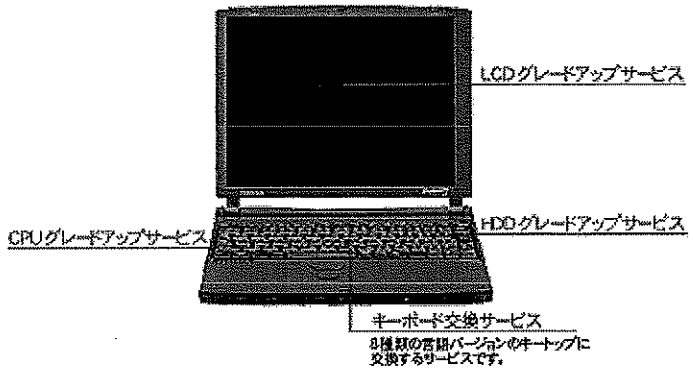
m n [h E F A n
¢ e ‰ ″ ‚ @
E f ″ C ^ [
l b g A
† , > ' A
¯

^ … ¤ d b ª A †
m F > ‡ ¢ B

m I W i
X n
J [O T [r X
¤ ¿ ¯
F J [O I
J [″
¢ • B
@ @
1 [~ O T [r X
¯

[N v g ~ ¨
‰ fl p \ R I
¶ ″ ¤ D t H g
† w Ł ¢ ‰ fl • B
@ @
L N ^ [v g
l C [i [u U [
X L N ^ p \ R
v g • B
@ @

m j [A T [r X n
¤ ¿ ¯
] „ ~ V i ˘ • • B



ピックアップサービス

お客様のご希望の日時にお引き取りにうかがいます。
送付の荷造りも不要です。

¤ ¢ „

¯

Tel.03-3457-8148

t ^

> j

Fax.03-5444-9450

t ^

http://www2.toshiba.co.jp/pc/upgrade/

@ @ @

… ¿

¯

¡
§
@
c ˘ F
œ E ¯ I œ > j

¯

¡
§
@
c ˘ F
œ E ¯ I œ > j

¯

¡
§
@
c ˘ F
œ E ¯ I œ > j

*

m [g

¯ ˝ A m [g p \ R s Œ ¯ ¢ E V F A * A¢ S ¢

E m [g p \ R

¯ • B

*: f [^ N G X g † , F


@

@


@


@


@

100080
(010) 6254 5820

6 Numerical experiments

The experiments we run on 4 and 15 R10000 processors of a 16-processor SGI Origin/2000. The results of the experiments are presented in Tab. 6. The first line in the table corresponds to the smoothed aggregation technique with a smoother of degree 1. The lines 2–6 demonstrate the effect of the additional energy minimization. In those experiments, the smoothed prolongator has been used as an initial guess and its nonzero strauctures has been for defining Z in (14).

In all the experiments below, the method is used as a preconditioner for the conjugate gradient method, and the stopping criterion used was

$$\|\mathbf{r}^i\|_B \leq (10^{-5}/\sqrt{\mathbf{cond}(B,A)} \cdot \|\mathbf{r}^0\|_B$$

where B is the preconditioner, $\mathbf{r}^i$ denotes the residual after i steps of the iteration, and $\mathrm{cond}(B,A)$ is a condition number estimate computed at run time.

solid, (see Fig. 3) 75,174 dofs, 4 CPUs.					solid, 407,277 dofs, 15 CPUs.			
min st	setup/iter/tot time [s]	num it.	cond est	tim/tim sm aggr	setup/iter/tot time [s]	num it.	cond est	tim/tim sm aggr
1	8.1/5.3/13.4	8	2.58	1.00	12.7/36.3/49.1	13	6.16	1.00
2	8.6/5.3/13.9	8	2.55	1.03	15.4/34.0/49.4	12	5.04	1.01
3	8.8/5.4/14.2	8	2.50	1.06	18.1/32.0/50.1	11	4.45	1.02
4	8.8/4.7/13.7	7	2.20	1.02	20.2/31.1/51.4	11	4.23	1.04
5	9.0/4.7/13.8	7	2.21	1.03	22.4/28.4/50.8	10	4.00	1.03
6	9.5/4.7/14.3	7	2.17	1.06	24.9/28.5/53.5	10	3.90	1.09

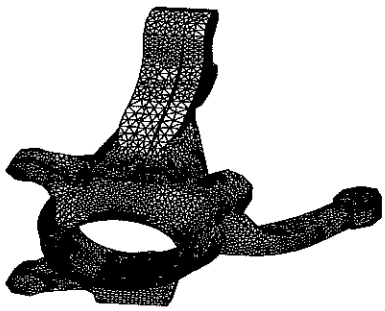


Fig. 3. The mesh of an automobile steering knuckle. Courtesy of Charbel Farhat, University of Colorado at Boulder.)

References

1. J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.
2. A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.
3. T. F. Chan, J, Xu, and L. Zikatanov, *An agglomeration multigrid for unstructured meshes.*, In: Domain Decomposition Methods 10, (Proceedings of the tenth international conference on domain decomposition methods) Mandel, Farhat, Cai Eds., AMS 1998.
4. T. F. Chan, S. Go, and L. Zikatanov, *Lecture Notes on Multilevel Methods for Elliptic Problems on Unstructured Grids*, UCLA CAM Report 97-11, March 1997. Lectures notes for the lecture series "Computational Fluid Dynamics", von Karman Inst., Belgium, March 3-7, 1997. An abridged version has been published as CAM Report 97-36, August, 1997 and appeared in "Computational Fluid Dynamics Review 1997", Hafez and Oshima (eds.), Wiley.
5. P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.
6. P. Vaněk, M. Brezina, and J. Mandel, *Convergence of Algebraic Multigrid Based on Smoothed Aggregation*, Submitted to Num. Math.
7. P. Vaněk, A. Janka, and H. Guillard, *Convergence of Petrov-Galerkin Smoothed Aggregation Method*, To appear.
8. P. Vaněk, *Fast multigrid solver.* Applications of Mathematics, to appear.
9. *Acceleration of convergence of a two-level algorithm by smoothing trarnsfer operator*, Applications of Mathematics, 37 (1992), pp. 265–274.
10. P. Vaněk, M. Brezina, and R. Tezaur, *Two-Grid Method for Linear Elasticity on Unstructured Meshes*, To appear in SIAM J. Sci. Comp.
11. J. Mandel, M. Brezina, and P. Vaněk, *Energy Optimization of Algebraic Multigrid Bases*, To appear in Computing
12. W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, UCLA CAM Report 98-6, Department of Mathematics, UCLA, February 1998.

```
// File backward.cpp
// Recursive string print.
#include <iostream.h>

void backward( char *str )   // str must be null-terminated.
{
   if ( *str )
   {
      backward( str+1 );
      cout << *str ;
   }
}

int main()
{
  char s[82];
  cout << "Enter a string: ";
  cin.getline( s, 81 );
  cout << "Your string reversed is: \n";
  backward(s);
  cout << "\n";
  return 0;
}
```

SIAM Journal on Matrix Analysis and Applications

Volume 17, Number 3, July 1996
pp. 465-469

Any Nonincreasing Convergence Curve is Possible for GMRES

Anne Greenbaum, Vlastimil Ptak, Zdenuek Strakous

Abstract. Given a nonincreasing positive sequence $f(0) \geq f(1) \geq \cdots \geq f(n-1) > 0$, it is shown that there exists an n by n matrix A and a vector r^0 with $\|r^0\| = f(0)$ such that $f(k) = \|r^k\|$, $k = 1, \ldots, n-1$, where r^k is the residual at step k of the GMRES algorithm applied to the linear system $Ax = b$, with initial residual $r^0 = b - Ax^0$. Moreover, the matrix A can be chosen to have any desired eigenvalues.

Key words. GMRES, Krylov subspace, Krylov residual space

AMS Subject Classifications. 65F10, 65F15