NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2002; **01**:1–22 *Prepared using* nlaauth.cls *[Version: 2000/03/22 v1.0]*

MULTILEVEL BLOCK FACTORIZATIONS IN GENERALIZED HIERARCHICAL BASES

Edmond Chow* and Panayot S. Vassilevski

Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, L-560, Box 808, Livermore, CA 94551, U.S.A.

SUMMARY

This paper studies the use of a generalized hierarchical basis transformation at each level of a multilevel block factorization. The factorization may be used as a preconditioner to the conjugate gradient method, or the structure it sets up may be used to define a multigrid method. The basis transformation is performed with an averaged piecewise constant interpolant and is applicable to unstructured elliptic problems. The results show greatly improved convergence rate when the transformation is applied for solving sample diffusion and elasticity problems. The cost of the method, however, grows and can get very high with the number of of nonzeros per row.

1. INTRODUCTION

Hierarchical basis (HB) preconditioners are composed of a transformation of the nodal basis coefficient matrix A to a hierarchical basis, a preconditioning of this HB coefficient matrix, and a transformation of this preconditioning back to the nodal basis [37, 3, 26]. For elliptic problems, the preconditioned matrices have condition number $O((\log(h^{-1}))^2)$ in two dimensions and $O(h^{-1})$ in three dimensions, where h is the mesh size. Although these condition numbers are poorer than for multigrid methods, HB preconditioners may be more robust, relying only on local properties of the mesh in their analysis, while giving scalability that is better than for many other preconditioners.

HB preconditioners are typically applied to finite element matrices with adaptive local mesh refinement where the hierarchical basis is clearly defined. Recently, however, methods have been developed to construct "generalized" hierarchical bases for completely unstructured problems (i.e., no nested meshes) so that HB preconditioners may be applied [8, 9, 10, 5]. These techniques sequentially select "fine" grid points as those that are near the center of two or three other grid points; these latter grid points are then labeled as "vertex parents". The vertex parents serve as the grid points on the coarser mesh.

*Correspondence to: Lawrence Livermore National Laboratory, L-560, Box 808, Livermore, CA 94551, E-mail: echow@llnl.gov.

Contract/grant sponsor: This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

This paper proposes a method for unstructured problems that assumes that an approximate hierarchical basis for the finite element space is not available or is difficult to find. Instead, a very simple generalized hierarchical basis is used, based on coarsening and interpolation ideas from algebraic multigrid (AMG) methods. The basis is constructed algebraically. The possibly poorer A-orthogonality of the new basis vectors between different levels translates into a transformed matrix that is not as strongly block diagonal and is less well-conditioned than when a good HB transformation can be found. To compensate for this, we use a block factorization preconditioner instead of the usual block diagonal or block Gauss-Seidel preconditioners at each level of the transformed matrix. In principle, the block factorization preconditioner can be made more accurate if necessary when the generalized HB transformation is poor. The combination of the HB transformation with the approximate block factorization can be viewed as a modified block factorization in the sense that certain vectors that are in the near-nullspace of A remain in the near-nullspace of the approximate Schur complement.

The approximate block factorization sets up a structure very similar to that of multigrid methods. In particular, coarse grid operators are constructed, as well as operators that act as prolongators. The multilevel block factorization that is recursively defined at each level may be used as a smoother to a multigrid process with the above components. This defines a type of W-cycle multigrid. More precisely, the kth coarse level grid is visited $\mathscr{O}(k)$ times (versus $\mathscr{O}(2^k)$ times in a model 2-D or 3-D geometric coarsening for a true W-cycle).

Methods that are related to HB preconditioners include those that use a hierarchical ordering of the grid points, such as the classical two-level methods, e.g., [4, 1], and some incomplete LU factorization techniques, e.g., [30, 13]. Multilevel block factorizations can also be connected to multigrid methods, and indeed, for many problems, a hierarchical basis transformation is not necessary to get multigrid convergence rates, e.g., [24, 25].

The proposed hierarchical basis block factorization (HBBF) is a middle ground between algebraic multigrid methods and multilevel block factorization preconditioners. The former relies on coarse grids and interpolation operators that match the problem being solved. The latter uses general purpose ILU or sparse approximate inverse techniques. HBBF utilizes a simple interpolation technique. When this interpolation is effective, the multilevel block factorization is economical to carry out; when it is less effective, the method relies more on the block factorization to compute an accurate preconditioning. HBBF can also be used to define a multigrid method, which we call BFMG. In section 2, these ideas will be made precise. Section 3 reports numerical results that illustrate the behavior of the multilevel block factorization with and without the generalized HB transformation.

2. HIERARCHICAL BASIS BLOCK FACTORIZATION

2.1. *Hierarchical basis transformation*

For simplicity, we will only discuss the hierarchical basis transformation for two levels; the multilevel case is defined recursively. Consider the symmetric positive definite linear system in the nodal basis, $Ax = b$, and a partitioning of the variables and corresponding equations into two sets, called *fine* and *coarse*. The partitioning induces the block form

$$\begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix} \begin{pmatrix} x_f \\ x_c \end{pmatrix} = \begin{pmatrix} b_f \\ b_c \end{pmatrix} \tag{1}$$

where the subscripts $(.)_f$ and $(.)_c$ indicate the fine and coarse sets, respectively.

A hierarchical basis transformation $\mathscr{J}$ transforms a vector from a hierarchical basis to a nodal basis. We consider hierarchical basis transformations of the form

$$\mathscr{J} = \begin{pmatrix} I & \mathscr{P} \\ 0 & I \end{pmatrix} \tag{2}$$

where the partitioning of $\mathscr{J}$ is the same as the partitioning of A. In classical hierarchical basis methods, $\mathscr{P}$ is a matrix with two nonzero entries of 1/2 in each row, corresponding to the contribution of the coarse grid basis vectors to the fine variable. Our choice of $\mathscr{P}$ will be discussed below.

Given the transformation $\mathscr{J}$, the linear system to be solved in the hierarchical basis is

$$\widehat{A}\widehat{x} = \widehat{b} \tag{3}$$

where $\widehat{A} = \mathscr{J}^T A \mathscr{J}$ and $\widehat{b} = \mathscr{J}^T b$, with the solution in the nodal basis being recovered by $x = \mathscr{J}\widehat{x}$. The matrix $\widehat{A}$ has the block form

$$\begin{pmatrix} \widehat{A}_{ff} & \widehat{A}_{fc} \\ \widehat{A}_{cf} & \widehat{A}_{cc} \end{pmatrix} \tag{4}$$

where

$$\begin{aligned} \widehat{A}_{ff} &= A_{ff} \\ \widehat{A}_{fc} &= A_{ff}\mathscr{P} + A_{fc} \\ \widehat{A}_{cf} &= \mathscr{P}^T A_{ff} + A_{cf} \\ \widehat{A}_{cc} &= \mathscr{P}^T A_{ff}\mathscr{P} + A_{cf}\mathscr{P} + \mathscr{P}^T A_{fc} + A_{cc} \end{aligned}$$

and

$$\begin{aligned} \widehat{b}_f &= b_f \\ \widehat{b}_c &= \mathscr{P}^T b_f + b_c \\ x_f &= \widehat{x}_f + \mathscr{P}\widehat{x}_c \\ x_c &= \widehat{x}_c. \end{aligned}$$

If $\mathscr{P} \approx -A_{ff}^{-1}A_{fc}$, then the off-diagonal blocks $\widehat{A}_{fc}$ and $\widehat{A}_{cf}$ are almost zero, i.e., the matrix is almost block diagonal. Finally, an important property is that the inverse HB transformation is sparse,

$$\mathscr{J}^{-1} = \begin{pmatrix} I & -\mathscr{P} \\ 0 & I \end{pmatrix}$$

and thus the preconditioning in the hierarchical basis is easily transformed back to the nodal basis.

From the implementation point of view, the transformed matrix $\widehat{A}$, which is denser than the nodal basis matrix, does not need to be stored during the solve phase—it is utilized in factored form.

The matrix $\mathscr{P}$ is analogous to the coarse-to-fine prolongation mapping in AMG. Our aim in the remainder of this subsection is to establish some choices of $\mathscr{P}$ for the HBBF preconditioner.

2.1.1. Coarsening In AMG, coarsening refers to the partitioning of the variables into fine and coarse sets. Almost all coarsening algorithms rely on some determination of whether a coupling between two variables is “strong” or “weak.” From there, the algorithms may choose the coarse set to be a set of variables that do not have any strong couplings between them. In graph theory terminology, this is called an *independent set*.

In AMG defined in [29] (motivated mostly for M–matrices), a variable x_i is *strongly coupled* to x_j if

$$-a_{ij} \geq \theta_s \max_{k \neq i} \{-a_{ik}\} \tag{5}$$

where $0 < \theta_s \leq 1$ is called the *strength threshold*. We additionally say that for $\theta_s = 0$, x_i is strongly coupled to x_j if $a_{ij} < 0$. In this paper, we coarsen by using this definition of strong coupling and select an independent set as the coarse set.

Coarsening procedures are also used in some multilevel block factorizations to define the variables that form the next level. Here, an objective of several practitioners is to select the fine set such that A_{ff} is diagonally dominant so that relaxation or solves with this matrix is efficient [14, 33, 19]. These procedures, however, do not try to assure that the coarse set provides good interpolation for the fine grid problem.

2.1.2. Interpolation Once the coarse and fine sets have been chosen, interpolation defines the weights in the matrix $\mathscr{P}$. The definition of strong couplings is also used here to define which coarse variables contribute to which fine variables in the hierarchical basis.

Let the "smooth vector" e denote a vector from the "smooth" part of the spectrum of A, i.e., $Ae \approx 0$. For scalar elliptic PDEs, e can be the vector of all ones. Further let e_f and e_c denote the components of e on the fine and coarse variables, respectively. It is desirable that $\mathscr{P}$ properly interpolates these smooth vectors, i.e.,

$$\mathscr{P} e_c = e_f. \tag{6}$$

For a single vector e, this can always be exactly satisfied by scaling the rows of $\mathscr{P}$. Combined with $Ae \approx 0$, condition (6) leads to

$$\widehat{A}_{cc} e_c = \begin{pmatrix} \mathscr{P} \\ I \end{pmatrix}^T A \begin{pmatrix} \mathscr{P} \\ I \end{pmatrix} e_c = (\mathscr{P}^T A_{ff} \mathscr{P} + A_{cf} \mathscr{P} + \mathscr{P}^T A_{fc} + A_{cc}) e_c \approx 0 \tag{7}$$

which means that the smooth vector e is preserved in the near-nullspace of $\widehat{A}_{cc}$. Thus, as a coarse grid operator, $\widehat{A}_{cc}$ approximates the behavior of A, at least for the vector e.

A simple interpolant that satisfies (6) for $e = (1)$ and that does not depend on matrix values is the *averaged piecewise constant* or *equally weighted* interpolant. Given an ordering of the fine and coarse variables, this interpolant is defined as

$$\mathscr{P}_{ij} = \begin{cases} 1/d_i & \text{the } i\text{th fine variable is strongly coupled to the } j\text{th coarse variable} \\ 0 & \text{otherwise} \end{cases}$$

where d_i is the number of coarse variables that are strongly coupled to variable i.

Note that all strong couplings are used in this interpolant. In some cases, some of these strong couplings are redundant and can be neglected. In [8, 9, 10, 5], only two or three couplings for each fine variable are used in the generalized hierarchical basis transformation. In section 3.6, we experiment with using only a single strong coupling (corresponding to a *piecewise constant* interpolant (denoted by $\mathscr{P}_1$), and with using at most two strong couplings (denoted by $\mathscr{P}_2$), in order to reduce the cost of the HB transformation.

More sophisticated choices for $\mathscr{P}$ can be used. For example, if access to the geometric coordinates of the fine grid points is available, one may use 2 (in 2-D) or 3 (in 3-D) strongly coupled coarse nodes to interpolate linear functions exactly. This will generally change the weights of $\mathscr{P}$ in the above formula.

2.2. *Approximate block factorization*

2.2.1. Approximate block factorization in the nodal basis Given a partitioning of the variables into fine and coarse sets, the approximate block LU factorization of the matrix A in the nodal basis is

$$\begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix} \approx \begin{pmatrix} A_{ff} & 0 \\ A_{cf} & S \end{pmatrix} \begin{pmatrix} I & -P \\ 0 & I \end{pmatrix} \quad (8)$$

where $S \approx A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$ is an approximation to the Schur complement and P is an approximation to $-A_{ff}^{-1}A_{fc}$. To solve approximately with this factorization, solves with A_{ff} and S are required, either of which may be performed exactly or approximately.

The *multilevel* factorization recursively applies this factorization to S [2, 32, 31, 6]. In order for this process to be economical, a sparse approximation to S is typically needed. There are many proposed approximations to S, including several based on multigrid ideas [27, 36, 28, 7, 5]. In this paper, we use approximations based on algebraic techniques. For example, the following approximations are possible.

- $S_1 = A_{cc} - A_{cf}\widetilde{A_{ff}^{-1}}A_{fc}$, where $\widetilde{A_{ff}^{-1}}$ is a sparse approximation to A_{ff}^{-1}.
- $S_2 = A_{cc} + A_{cf}P$ where P is a sparse approximation to $-A_{ff}^{-1}A_{fc}$, which may or may not be the same as the P in (8). This construction of S_2 is not necessarily symmetric.
- $S_3 = \begin{pmatrix} P^T, & I \end{pmatrix} A \begin{pmatrix} P \\ I \end{pmatrix}$, where P is again a sparse approximation to $-A_{ff}^{-1}A_{fc}$. We refer to this as the *Galerkin* form. The matrix S_3 is positive definite if A is positive definite. In addition, S_3 is the exact Schur complement if $P = -A_{ff}^{-1}A_{fc}$.
- Given an ILU factorization partitioned in the same way as (8), i.e.,

  $$\begin{pmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{pmatrix} \approx \begin{pmatrix} L_{ff} & 0 \\ L_{cf} & L_{cc} \end{pmatrix} \begin{pmatrix} U_{ff} & U_{fc} \\ 0 & U_{cc} \end{pmatrix} \quad (9)$$

  we have the approximation $S_4 = A_{cc} - L_{cf}U_{fc}$. This approximation can be created from a *partial* ILU factorization (the factors L_{cc} and U_{cc} are not computed) [31].

Once an approximation S has been computed on each level, the following algorithm may be used to approximately solve (1) using a multilevel approximate block factorization. The algorithm is written in a way to show its similarity to multigrid methods. The algorithm assumes that P in (8) has the form $-\widetilde{A_{ff}^{-1}}A_{fc}$.

1: $b^H = b_c - A_{cf}\widetilde{A_{ff}^{-1}}b_f$
2: Solve $Sx_c = b^H$ recursively, with an exact solve on the final level
3: $x_f = -\widetilde{A_{ff}^{-1}}A_{fc}x_c$
4: $x_f = x_f + \widetilde{A_{ff}^{-1}}b_f$

Algorithm 1: BF, approximate solution of (1) using a multilevel block factorization

In the algorithm, b^H can be interpreted as the restriction of b onto a coarse grid. The restriction and prolongation operators are $\begin{pmatrix} -A_{cf}\widetilde{A_{ff}^{-1}}, & I \end{pmatrix}$ and $\begin{pmatrix} -\widetilde{A_{ff}^{-1}}A_{fc} \\ I \end{pmatrix}$ respectively. The actions of $\widetilde{A_{ff}^{-1}}$ (required in steps 1 and 3) can be viewed as *F-smoothing* [35].

Unlike multigrid methods, approximate block factorization preconditionings do not give scalable convergence rates because, in general, S is not a suitable coarse grid operator: smooth vectors of A may not be preserved in the near-nullspace of S (see section 2.1.2). However, if S is constructed as $A_{cc} - A_{cf}\widetilde{A_{ff}^{-1}}A_{fc}$, then the row-sum condition

$$\widetilde{A_{ff}^{-1}}A_{ff}e_f \approx e_f \tag{10}$$

on $\widetilde{A_{ff}^{-1}}$ leads to $Se_c \approx 0$ being satisfied (since $A_{ff}e_f + A_{fc}v_c \approx 0$). The row-sum condition on the approximate inverse is not easy to satisfy, however.

2.2.2. Approximate block factorization in a hierarchical basis In HBBF, the approximate block factorization is performed in the hierarchical basis. Given the matrix $\widehat{A}$ in the block form (4), its approximate block LU factorization is

$$\begin{pmatrix} \widehat{A}_{ff} & \widehat{A}_{fc} \\ \widehat{A}_{cf} & \widehat{A}_{cc} \end{pmatrix} \approx \begin{pmatrix} \widehat{A}_{ff} & 0 \\ \widehat{A}_{cf} & \widehat{S} \end{pmatrix} \begin{pmatrix} I & -\widehat{P} \\ 0 & I \end{pmatrix} \tag{11}$$

where $\widehat{S} \approx \widehat{A}_{cc} - \widehat{A}_{cf}\widehat{A}_{ff}^{-1}\widehat{A}_{fc}$ is an approximation to the Schur complement and $\widehat{P}$ is an approximation to $-\widehat{A}_{ff}^{-1}\widehat{A}_{fc}$. We note that the *exact* Schur complement of the transformed matrix is equal to the exact Schur complement of A in the nodal basis, i.e.,

$$\widehat{A}_{cc} - \widehat{A}_{cf}\widehat{A}_{ff}^{-1}\widehat{A}_{fc} = A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}. \tag{12}$$

In this paper, for SPD problems, we focus on Schur complement approximations in Galerkin form. We thus define the following three approximations to the Schur complement.

Definition 2.1. *$\widehat{A}_{cc} \equiv \mathscr{P}^T A_{ff}\mathscr{P} + A_{cf}\mathscr{P} + \mathscr{P}^T A_{fc} + A_{cc}$, where $\mathscr{P}$ was defined in section 2.1.2.*

This leads to a method similar to the hierarchical basis multigrid method, HBMG.

Definition 2.2. *$S \equiv P^T A_{ff}P + A_{cf}P + P^T A_{fc} + A_{cc}$, where P is some approximation to $-A_{ff}^{-1}A_{fc}$.*

This approximate Schur complement is defined for approximate block factorizations in the nodal basis. If P has the form $-\widetilde{A_{ff}^{-1}}A_{fc}$, then

$$S = A_{cc} + A_{cf}\widetilde{A_{ff}^{-1}}A_{ff}\widetilde{A_{ff}^{-1}}A_{fc} - 2A_{cf}\widetilde{A_{ff}^{-1}}A_{fc}. \tag{13}$$

Definition 2.3. *$\widehat{S} \equiv \widehat{P}^T \widehat{A}_{ff}\widehat{P} + \widehat{A}_{cf}\widehat{P} + \widehat{P}^T \widehat{A}_{fc} + \widehat{A}_{cc}$, where $\widehat{P}$ is some approximation to $-A_{ff}^{-1}\widehat{A}_{fc}$.*

This approximate Schur complement is defined for approximate block factorizations in a hierarchical basis. If $\widehat{P}$ has the form $-\widetilde{A_{ff}^{-1}}\widehat{A}_{fc}$, then

$$\widehat{S} = \widehat{A}_{cc} + \widehat{A}_{cf}\widetilde{A_{ff}^{-1}}A_{ff}\widetilde{A_{ff}^{-1}}\widehat{A}_{fc} - 2\widehat{A}_{cf}\widetilde{A_{ff}^{-1}}\widehat{A}_{fc}. \tag{14}$$

If the generalized hierarchical basis transformation is good, then the terms $\widehat{A}_{cf}$ and $\widehat{A}_{fc}$ in (14) will be small and $\widehat{A}_{cc} \approx \widehat{S}$ will be a good coarse grid operator. The approximation $\widehat{S}$ is generally an improvement over $\widehat{A}_{cc}$, especially when the transformation is poor.

Further, if the terms $\widehat{A}_{cf}$ and $\widehat{A}_{fc}$ are smaller in some sense than the terms A_{cf} and A_{fc}, then $\widehat{S}$ depends less on the accuracy of $\widetilde{A_{ff}^{-1}}$ than S does. However, if $\widetilde{A_{ff}^{-1}}$ is very accurate, then the approximations $\widehat{S}$ and S have similar quality; they all approximate well the exact Schur complement.

Proposition 2.1. *Assume that* $\widehat{P}$ *has the form* $-\widetilde{A_{ff}^{-1}}\widehat{A}_{fc}$ *and for some vector* $e = \begin{bmatrix} e_f \\ e_c \end{bmatrix}$ *let the following properties hold:*

- $Ae \approx 0$ *and in particular,* $A_{ff}e_f + A_{fc}e_c \approx 0$*, that is, e is in the near-nullspace of A. Such vectors are commonly referred to as smooth vectors in AMG.*
- *the generalized HB transformation preserves e, that is,* $e_f = \mathscr{P}e_c$*.*

Then, the approximate Schur complement $\widehat{S}$ *contains* e_c *in its near-nullspace, that is,* $\widehat{S}e_c \approx 0$*.*

Proof. This property is seen from the identity

$$\widehat{S} = \begin{pmatrix} \widehat{P} \\ I \end{pmatrix}^T \widehat{A} \begin{pmatrix} \widehat{P} \\ I \end{pmatrix},$$

and the fact that

$$\widehat{P}e_c = -\widetilde{A_{ff}^{-1}}\widehat{A}_{fc}e_c \approx 0.$$

The latter holds since $\widehat{A}_{fc}e_c = (A_{ff}\mathscr{P} + A_{fc})e_c = A_{ff}e_f + A_{fc}e_c \approx 0$. Hence

$$\begin{aligned} \widehat{S}e_c &\approx \begin{pmatrix} \widehat{P} \\ I \end{pmatrix}^T \widehat{A} \begin{bmatrix} 0 \\ e_c \end{bmatrix} \\ &\approx \begin{pmatrix} \widehat{P} \\ I \end{pmatrix}^T \begin{bmatrix} 0 \\ \widehat{A}_{cc}e_c \end{bmatrix} \\ &\approx \widehat{A}_{cc}e_c \\ &\approx \begin{bmatrix} \mathscr{P} \\ I \end{bmatrix}^T A \begin{bmatrix} \mathscr{P} \\ I \end{bmatrix} e_c \\ &= \begin{bmatrix} \mathscr{P} \\ I \end{bmatrix}^T Ae \\ &\approx 0. \end{aligned}$$

□

Given an approximation $\widehat{S}$ to the Schur complement, the following algorithm may be used to approximately solve (3) using a multilevel approximate block factorization in a hierarchical basis. Note that the transformed matrix is not stored. The algorithm for the solution in the nodal basis is recovered if $\mathscr{P} = 0$. The algorithm assumes that $\widehat{P}$ has the form $-\widetilde{A_{ff}^{-1}}\widehat{A}_{fc}$.

1: $x_f = \widetilde{A_{ff}^{-1}}b_f$
2: Solve $\widehat{S}x_c = \{b_c + \mathscr{P}^T(b_f - A_{ff}x_f) - A_{cf}x_f\}$ recursively, with an exact solve on the final level
3: $x_f = x_f - \widetilde{A_{ff}^{-1}}\{A_{ff}\mathscr{P}x_c + A_{fc}x_c\} + \mathscr{P}x_c$

Algorithm 2: HBBF, approximate solution of (3) using a multilevel block factorization in a generalized hierarchical basis

Remark 2.1. *It is clear that step (3) of Algorithm 2 can be rewritten as,*

$$x_f = x_f + \left[(I - \widetilde{A_{ff}^{-1}}A_{ff})\mathscr{P} - \widetilde{A_{ff}^{-1}}A_{fc}\right]x_c.$$

Hence, the expression

$$\widetilde{P} \equiv (I - \widetilde{A_{ff}^{-1}} A_{ff}) \mathscr{P} - \widetilde{A_{ff}^{-1}} A_{fc} = \mathscr{P} + \widehat{P}, \tag{15}$$

can be viewed as a modified interpolation matrix. Note that, in the setting of Proposition 2.1, the modified interpolation matrix satisfies $\widetilde{P} e_c = e_f$. Finally, it is also clear that a better quality $\widetilde{A_{ff}^{-1}}$ implies less importance of the HB transformation matrix $\mathscr{P}$ (the weight $(I - \widetilde{A_{ff}^{-1}} A_{ff})$ is small in that case).

2.2.3. Approximating $A_{ff}^{-1} \widehat{A}_{fc}$ The efficiency of HBBF depends critically on how $\widehat{P} \approx -A_{ff}^{-1} \widehat{A}_{fc}$ is computed. This choice may be related to the method chosen to solve with A_{ff}. The following are some of the options. Similar comments apply to the approximation $P \approx -A_{ff}^{-1} A_{fc}$.

Incomplete factorization with sparse approximate solves. It is popular to use an incomplete factorization $L_{ff} U_{ff} \approx A_{ff}$ to solve approximately with A_{ff}. It is too costly, however, to use these solves to form $\widehat{P}$ since the matrix $(L_{ff} U_{ff})^{-1} \widehat{A}_{fc}$, is typically dense, and these solves do not take advantage of the sparseness of $\widehat{A}_{fc}$. However, it is possible to solve approximately with the incomplete factorization such that the result is sparse [11].

We use the "level 0" strategy described in [11], where the sparsity pattern of the approximate $(L_{ff} U_{ff})^{-1} \widehat{A}_{fc}$ is restricted to the pattern of $\widehat{A}_{fc}$. For $L_{ff} U_{ff} x = b$ where b is sparse, this strategy only uses the nonzeros in rows and columns of L_{ff} and U_{ff} corresponding to nonzeros in b; the other nonzeros are neglected.

Sparse approximate inverses. A variety of techniques are available for approximating a symmetric positive definite A_{ff}^{-1} by $G^T G$, where G is sparse and approximates the inverse of the lower triangular Cholesky factor, L, of A_{ff} [12, 23, 34]. We restrict the pattern of G to the pattern of the lower triangular part of A_{ff} and perform the minimization

$$\min_G \|I - GL\|_F^2.$$

The matrix L does not need to be known, and the minimization is easily performed in parallel if necessary. The product $G^T G \widehat{A}_{fc}$ is sparse and is efficient to compute.

The matrix $\widehat{P} = -G^T G \widehat{A}_{fc}$ may still contain too many nonzeros for HBBF to be efficient. For this reason, small nonzeros in $\widehat{P}$ may be dropped. Since $\widehat{P}$ is usually constructed column-by-column, we drop an entry $\widehat{P}_{ij}$ if it satisfies

$$|\widehat{P}_{ij}| \leq \theta_p \max_k |\widehat{P}_{kj}| \tag{16}$$

where θ_p is a truncation threshold.

For nonsymmetric A_{ff}, nonsymmetric factorizations are available, as well as nonfactorized forms of the sparse approximate inverse [18, 20, 17]. We mention in passing that for nonfactorized sparse approximate inverses, it is possible to find $M = \widetilde{A_{ff}^{-1}}$ such that the row-sum condition (10) is satisfied. A matrix M satisfying this condition can be found by adding a constraint to the usual Frobenius norm minimization, i.e.,

$$\min_M \|I - M A_{ff}\|_F, \quad M A_{ff} e_f = e_f. \tag{17}$$

However, whether or not the constraint is well-defined depends on the sparsity pattern of M; see [22].

Frobenius norm minimization for $\widehat{P}$. The matrix $\widehat{P}$ may be defined by performing the minimization

$$\min_{\widehat{P}} \|A_{ff}\widehat{P} + \widehat{A}_{fc}\|_F$$

which is discussed in [16]. We choose the sparsity pattern of $\widehat{P}$ as the sparsity pattern of $\widehat{A}_{fc}$. The above minimization may be very costly when columns of $\widehat{A}_{fc}$ contain many nonzeros. We thus drop small entries in $\widehat{A}_{fc}$ prior to the minimization. The dropping is performed using the parameter θ_p in the same way entries in $\widehat{P}$ are dropped via (16).

2.3. A multigrid method based on the approximate block factorization

As mentioned, the approximate block factorization sets up a structure very similar to that of multigrid methods. Once the approximate block factorization is constructed, the following multigrid method can be defined. The method uses the $\widehat{S}$ matrices as coarse grid operators at each level, and the $\widetilde{P} = \mathscr{P} + \widehat{P}$ matrices (see (15)) in the prolongation and restriction operators.

```
Relax Ax = b using HBBF defined at the current level, with x = 0 initially
Construct the residual r = b − Ax
Restrict the residual using r^H = (P̃^T, I) r
Solve Ŝ e^H = r^H recursively, with an exact solve on the final level
Prolong the error using e = (P̃^T, I)^T e^H
Correct the approximate solution x = x + e
Relax Ax = b using HBBF defined at the current level
```

Algorithm 3: BFMG, a multigrid method based on approximate block factorization for solving $Ax = b$

3. NUMERICAL INVESTIGATIONS

The main goal of this section is to numerically compare the multilevel block factorization preconditioner with and without the generalized hierarchical basis transformation (the HBBF and BF preconditioners, respectively). We also test BFMG as a solver and as a preconditioner. We primarily use 2-D isotropic and anisotropic test problems with various mesh sizes, but include results on some difficult 3-D elasticity problems as well. We initially compare the convergence rate and scalability of the preconditioners with respect to some of the major options available, such as for Schur complement approximation, and then examine timings and storage requirements for the more competitive options.

The 2-D test problems are finite element discretizations of

$$\begin{aligned} au_{xx} + bu_{yy} &= f \quad \text{in} \quad \Omega = (0,1)^2 \\ u &= 0 \quad \text{on} \quad \partial\Omega \end{aligned}$$

where the right-hand side f was chosen randomly. For the anisotropic problems, the PDE coefficients were $a = 1$ and $b = 1000$. Linear triangular elements were used. The matrices were generated by a code by Stan Tomov (Texas A&M University).

Table I shows the number of equations n and the number of nonzeros nnz in the test matrices. The same grids were used for both the isotropic and anisotropic problems.

Problem	n	nnz
UNI2/ANI2	231	1491
UNI3/ANI3	861	5781
UNI4/ANI4	3321	22761
UNI5/ANI5	13041	90321
UNI6/ANI6	51681	359841
UNI7/ANI7	205761	1436481

Table I. Isotropic (UNI) and anisotropic (ANI) test matrices, showing number of equations n, and number of nonzeros nnz.

The storage required by the preconditioners is expressed in terms of grid and operator complexities. These terms are common in the AMG literature, e.g., [15]. *Grid complexity* is the total number of grid points, on all grids, divided by the number of grid points on the finest grid. *Operator complexity* is the total number of nonzero entries, in all coarse and fine grid matrices, divided by the number of nonzero entries in the fine grid matrix.

BF and HBBF were accelerated by the conjugate gradient method. A zero initial guess was used, and the iterations were stopped when the preconditioned residual norm was decreased by 12 orders of magnitude. The experiments were run on a Linux 1.5 GHz Intel Xeon computer with 256 kbytes of cache memory and 512 Mbytes of main memory.

3.1. Convergence rate and scalability

In this section we investigate the convergence rate and scalability of BF and HBBF with respect to the Schur complement approximation, the use of pre- and post-smoothing at each level, "modified" (row-sum preserving) approximations for A_{ff}, and the accuracy of the A_{ff} solve.

Tables II and III show iteration counts for BF and HBBF when the number of levels was fixed at 4. For larger problems, the size of the coarsest grid is larger, which influences computation time, but these tables allow a comparison of convergence rate when the number of levels is fixed.

The truncation threshold θ_p was 0.01, in order to reduce the cost of the preconditioning. In the base case, the matrix A_{ff} was approximated with a level 0 incomplete Cholesky factorization. Modified and level 1 factorizations were used in other cases. Sparse approximate solves with the incomplete factorizations were not used here. The smoother, if used, was symmetric Gauss-Seidel. The Schur complement approximations described in section 2.2.2 are compared.

The following observations may be made:

- In all cases, HBBF($\widehat{S}$) shows better convergence rate and scalability compared to the other preconditioners.
- Adding a pre- and post-smoothing at each level improves the performance of all the preconditioners, especially HBBF($\widehat{A}_{cc}$). Without smoothing, HBBF($\widehat{A}_{cc}$) is similar to a simple coarse grid correction.
- Using modified approximations for A_{ff} improves BF, as verified in the tables. However, the next subsection shows that it is too costly to approximate $A_{ff}^{-1}A_{fc}$ using incomplete factorization techniques, and thus modified approximations may not be readily used in general.
- Increasing the accuracy of the A_{ff} solve using IC(1) reduces the difference between the results for HBBF(S) and HBBF($\widehat{S}$), as expected.

In the remainder of this paper, BF refers to BF(S) and HBBF refers to HBBF($\widehat{S}$).

IC(0), no smoothing

	UNI2	UNI3	UNI4	UNI5
BF(S)	12	19	31	55
HBBF($\widehat{A}_{cc}$)	28	33	41	43
HBBF(S)	11	13	19	32
HBBF($\widehat{S}$)	11	13	14	15

IC(0), one smoothing step

	UNI2	UNI3	UNI4	UNI5
BF(S)	8	12	18	32
HBBF($\widehat{A}_{cc}$)	10	12	15	16
HBBF(S)	6	9	15	26
HBBF($\widehat{S}$)	5	7	7	8

Modified IC(0), no smoothing

	UNI2	UNI3	UNI4	UNI5
BF(S)	12	16	18	21
HBBF($\widehat{A}_{cc}$)	12	16	19	21
HBBF(S)	12	16	19	22
HBBF($\widehat{S}$)	12	16	18	19

IC(1), no smoothing

	UNI2	UNI3	UNI4	UNI5
BF(S)	8	11	15	24
HBBF($\widehat{A}_{cc}$)	26	31	38	40
HBBF(S)	7	8	10	13
HBBF($\widehat{S}$)	7	8	9	10

Table II. Iteration counts for the *isotropic* problems UNI2–UNI5 using BF and HBBF preconditioners with 4 levels.

IC(0), no smoothing

	ANI2	ANI3	ANI4	ANI5
BF(S)	16	22	28	44
HBBF($\widehat{A}_{cc}$)	35	53	65	80
HBBF(S)	14	18	18	27
HBBF($\widehat{S}$)	13	17	16	18

IC(0), one smoothing step

	ANI2	ANI3	ANI4	ANI5
BF(S)	10	14	17	25
HBBF($\widehat{A}_{cc}$)	16	24	32	40
HBBF(S)	8	11	14	22
HBBF($\widehat{S}$)	8	9	9	10

Modified IC(0), no smoothing

	ANI2	ANI3	ANI4	ANI5
BF(S)	16	22	23	28
HBBF($\widehat{A}_{cc}$)	36	60	72	93
HBBF(S)	14	21	21	27
HBBF($\widehat{S}$)	14	21	22	27

IC(1), no smoothing

	ANI2	ANI3	ANI4	ANI5
BF(S)	9	14	13	19
HBBF($\widehat{A}_{cc}$)	33	50	61	74
HBBF(S)	8	12	11	13
HBBF($\widehat{S}$)	8	12	11	12

Table III. Iteration counts for the *anisotropic* problems ANI2–ANI5 using BF and HBBF preconditioners with 4 levels.

3.2. *Comparing approximations of $A_{ff}^{-1}\widehat{A}_{fc}$*

The techniques for approximating $A_{ff}^{-1}\widehat{A}_{fc}$ described in section 2.2.3 are compared in Table IV. Results are shown for HBBF using the UNI6 test problem. The strength threshold $\theta_s = 0$ was used in these tests, and the recursion to the next level was stopped when the coarse grid matrix contained fewer than 100 equations. The table shows that incomplete factorization techniques for approximating $A_{ff}^{-1}\widehat{A}_{fc}$ lead to high setup timings. The most efficient method is to use a factorized sparse approximate inverse in making this approximation.

3.3. *Timings for UNI7 and ANI7*

This section reports detailed timings for the large UNI7 and ANI7 test problems using a variety of values for the thresholds θ_s and θ_p. Unfortunately for these methods, there is not a simple way to choose these thresholds that will give the lowest total computation time. A sparse approximate inverse

Incomplete factorization for A_{ff}

θ_p	levels	iterations	time (s) setup	solve	total	complexity grid	operator
0.003	4	17	68.91	1.24	70.15	1.31	4.20
0.010	4	18	65.44	1.13	66.57	1.31	3.55
0.030	4	21	63.21	1.19	64.40	1.32	2.97
0.100	5	45	60.92	2.17	63.09	1.33	2.35
0.300	5	103	60.22	4.22	64.44	1.35	1.84

Incomplete factorization for A_{ff} with sparse approximate solves

	levels	iterations	time (s) setup	solve	total	complexity grid	operator
	5	72	9.84	3.32	13.16	1.33	2.23

Sparse approximate inverse for A_{ff}^{-1}

θ_p	levels	iterations	time (s) setup	solve	total	complexity grid	operator
0.003	4	28	3.50	1.65	5.15	1.32	3.21
0.010	4	27	3.05	1.56	4.61	1.32	3.06
0.030	4	28	2.40	1.54	3.94	1.32	2.78
0.100	5	45	1.70	2.23	3.93	1.33	2.34
0.300	5	108	1.13	4.68	5.81	1.35	1.82

Frobenius norm minimization for $\widehat{P}$

θ_p	levels	iterations	time (s) setup	solve	total	complexity grid	operator
0.003	5	97	4.64	4.46	9.10	1.35	2.22
0.010	5	98	3.76	4.50	8.26	1.35	2.19
0.030	5	98	2.90	4.45	7.35	1.35	2.16
0.100	5	100	2.23	4.52	6.75	1.35	2.11
0.300	5	108	1.62	4.78	6.40	1.35	2.02

Table IV. HBBF with UNI6 problem: Comparison of techniques for approximating $A_{ff}^{-1}\widehat{A}_{fc}$.

was used in the approximation for $A_{ff}^{-1}\widehat{A}_{fc}$. Further, no smoothing was added to the BF and HBBF algorithms. Results of additional experiments (not shown here) revealed that the addition of smoothing usually increased the overall timings for these problems.

Four tables are shown: Tables V and VI show results using BF and HBBF for the isotropic problem UNI7, and Tables VII and VIII show these results for the anisotropic problem ANI7. For a wide range of parameter values, the results clearly show that HBBF has lower iteration counts and lower total timings than BF for these problems.

For a rough comparison, Table IX reports timings of the same problems solved using an AMG code called BoomerAMG [21], which is based on algorithms in [29]. BoomerAMG was used as a solver, rather than as a preconditioner. For the problem UNI7, BoomerAMG is faster than HBBF (accelerated by CG), but the fastest timing for HBBF is comparable. For the problem ANI7, the best timings for HBBF are better than the timing for BoomerAMG.

				time (s)			complexity	
θ_s	θ_p	levels	iterations	setup	solve	total	grid	operator
0.00	0.01	5	409	4.72	55.66	60.38	1.33	2.32
	0.03	6	410	3.96	54.07	58.03	1.34	2.11
	0.10	6	419	3.14	52.68	55.82	1.35	1.85
	0.30	7	492	2.36	57.91	60.27	1.39	1.57
0.25	0.01	8	323	9.28	52.04	61.32	1.51	3.87
	0.03	9	327	6.27	48.24	54.51	1.51	3.17
	0.10	9	337	4.05	45.96	50.01	1.51	2.48
	0.30	10	395	2.90	50.54	53.44	1.52	1.97
0.50	0.01	10	238	25.70	50.78	76.48	1.73	8.13
	0.03	10	241	13.70	45.33	59.03	1.73	6.16
	0.10	11	261	5.80	39.93	45.73	1.74	3.66
	0.30	12	305	4.03	43.48	47.51	1.75	2.86
0.75	0.01	12	164	45.52	43.73	89.25	1.92	13.58
	0.03	12	165	20.08	36.10	56.18	1.92	9.61
	0.10	13	198	6.68	33.34	40.02	1.93	4.96
	0.30	13	271	4.44	40.69	45.13	1.93	3.64
0.95	0.01	14	95	58.51	32.81	91.32	2.00	16.09
	0.03	14	104	22.72	24.73	47.45	2.00	11.08
	0.10	14	155	7.51	27.94	35.45	2.00	5.77
	0.30	14	255	4.54	40.31	44.85	2.00	3.83

Table V. Results for the isotropic UNI7 with BF preconditioning.

				time (s)			complexity	
θ_s	θ_p	levels	iterations	setup	solve	total	grid	operator
0.00	0.01	5	31	13.43	7.81	21.24	1.33	3.16
	0.03	5	33	10.56	7.94	18.50	1.33	2.87
	0.10	5	80	7.39	17.18	24.57	1.33	2.40
	0.30	6	212	4.98	40.67	45.65	1.36	1.87
0.25	0.01	7	22	23.67	6.58	30.25	1.51	5.35
	0.03	7	40	15.33	10.45	25.78	1.51	4.38
	0.10	8	94	9.40	21.32	30.72	1.51	3.29
	0.30	9	181	6.22	36.29	42.51	1.51	2.42
0.50	0.01	10	23	54.47	9.03	63.50	1.72	10.72
	0.03	10	44	28.03	14.41	42.44	1.72	8.03
	0.10	10	96	14.31	25.79	40.10	1.72	5.30
	0.30	11	183	9.34	41.42	50.76	1.73	3.63
0.75	0.01	12	25	85.13	22.10	107.23	1.92	16.86
	0.03	12	48	37.48	17.90	55.38	1.92	12.45
	0.10	13	111	14.69	30.35	45.04	1.93	6.76
	0.30	13	188	9.68	45.66	55.34	1.93	4.78
0.95	0.01	14	27	119.53	21.98	141.51	2.00	17.73
	0.03	14	53	31.25	19.63	50.88	2.00	12.72
	0.10	14	122	12.44	32.81	45.25	2.00	6.68
	0.30	14	212	8.18	49.30	57.48	2.00	4.51

Table VI. Results for the isotropic UNI7 with HBBF preconditioning.

				time (s)			complexity	
θ_s	θ_p	levels	iterations	setup	solve	total	grid	operator
0.00	0.01	7	975	7.53	158.37	165.90	1.62	3.62
	0.03	8	978	6.15	152.64	158.79	1.63	3.24
	0.10	8	911	4.74	134.55	139.29	1.68	2.85
	0.30	9	894	3.15	121.73	124.88	1.76	2.35
0.25	0.01	10	321	15.81	62.65	78.46	1.86	6.47
	0.03	10	327	10.19	58.36	68.55	1.86	5.30
	0.10	10	355	6.28	56.82	63.10	1.86	3.97
	0.30	11	455	3.49	63.94	67.43	1.87	2.62
0.50	0.01	11	209	23.29	45.33	68.62	1.91	8.05
	0.03	11	219	13.70	41.96	55.66	1.91	6.38
	0.10	11	249	7.53	41.77	49.30	1.91	4.55
	0.30	12	371	3.79	53.58	57.37	1.91	2.80
0.75	0.01	12	219	37.89	56.49	94.38	1.97	11.98
	0.03	12	229	19.27	50.69	69.96	1.97	8.90
	0.10	13	265	8.19	48.00	56.19	1.97	5.57
	0.30	13	386	3.73	58.42	62.15	1.98	3.07
0.95	0.01	14	243	56.17	77.46	133.63	2.02	15.79
	0.03	14	249	24.17	61.32	85.49	2.02	11.05
	0.10	14	278	9.37	53.59	62.96	2.02	6.42
	0.30	14	476	3.75	73.94	77.69	2.02	3.16

Table VII. Results for the anisotropic ANI7 with BF preconditioning.

				time (s)			complexity	
θ_s	θ_p	levels	iterations	setup	solve	total	grid	operator
0.00	0.01	5	387	31.07	123.17	154.24	1.59	5.26
	0.03	6	385	20.77	111.05	131.82	1.60	4.54
	0.10	6	385	13.27	98.95	112.22	1.62	3.74
	0.30	7	655	8.84	151.25	160.09	1.68	3.17
0.25	0.01	9	33	41.97	12.22	54.19	1.86	9.49
	0.03	9	35	25.48	11.19	36.67	1.86	7.61
	0.10	9	72	15.34	19.60	34.94	1.86	5.71
	0.30	11	366	10.09	87.29	97.38	1.87	4.37
0.50	0.01	10	22	54.90	8.79	63.69	1.90	11.57
	0.03	10	24	33.44	8.32	41.76	1.90	9.32
	0.10	11	65	18.49	18.47	36.96	1.91	6.76
	0.30	11	197	11.65	48.84	60.49	1.91	5.02
0.75	0.01	12	22	121.21	15.58	136.79	1.97	16.86
	0.03	12	26	44.87	10.41	55.28	1.97	12.97
	0.10	12	67	22.16	21.32	43.48	1.97	8.90
	0.30	13	184	11.56	47.58	59.14	1.97	5.72
0.95	0.01	14	22	127.40	27.52	154.92	2.02	20.33
	0.03	14	30	47.18	15.43	62.61	2.02	15.08
	0.10	14	83	21.95	27.96	49.91	2.02	9.64
	0.30	14	250	10.98	66.11	77.09	2.02	5.67

Table VIII. Results for the anisotropic ANI7 with HBBF preconditioning.

problem	levels	iterations	time (s) setup	solve	total	complexity grid	operator
UNI7	11	21	4.55	9.53	14.08	1.87	2.94
ANI7	12	139	3.83	57.59	61.42	1.91	3.04

Table IX. AMG results using a V(1,1) cycle with CF Gauss-Seidel relaxation, and strength threshold 0.25.

3.4. Algorithmic scalability

For problems in 2-D, the iteration counts for hierarchical basis methods scale with the square of the number of levels. The following results verify this theory by showing iteration counts for increasing problem sizes. Again, the recursions were stopped when the the size of the coarse grid problem was less than 100 equations. Table X tabulates the results and Figure 1 plots the iteration counts as a function of the square of the number of levels.

UNI2–UNI7, $\theta_s = 0.$, $\theta_p = 0.03$

	levels	iterations	time (s) setup	solve	total	complexity grid	operator
UNI2	2	16	0.01	0.00	0.01	1.17	1.60
UNI3	3	19	0.02	0.01	0.03	1.24	2.09
UNI4	3	23	0.10	0.06	0.16	1.28	2.42
UNI5	4	25	0.52	0.29	0.81	1.31	2.65
UNI6	4	28	2.39	1.53	3.92	1.32	2.78
UNI7	5	33	10.72	7.93	18.65	1.33	2.87

ANI2–ANI7, $\theta_s = 0.25$, $\theta_p = 0.03$

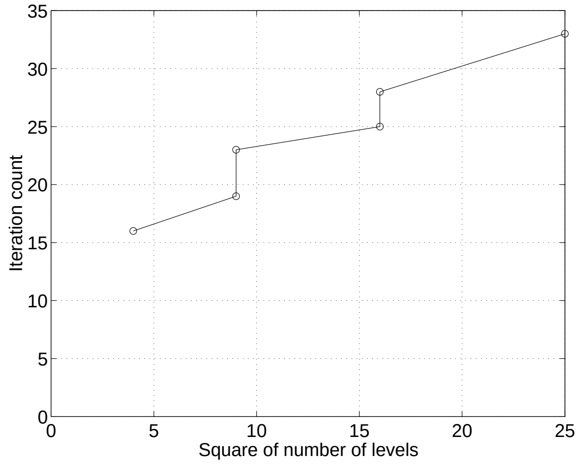
	levels	iterations	time (s) setup	solve	total	complexity grid	operator
ANI2	2	11	0.00	0.01	0.01	1.33	2.08
ANI3	4	14	0.03	0.01	0.04	1.64	4.20
ANI4	5	19	0.19	0.06	0.25	1.75	5.56
ANI5	7	23	1.14	0.36	1.50	1.81	6.66
ANI6	8	28	5.49	2.07	7.56	1.85	7.26
ANI7	9	35	25.83	11.22	37.05	1.86	7.61

Table X. HBBF results for increasing problem sizes.

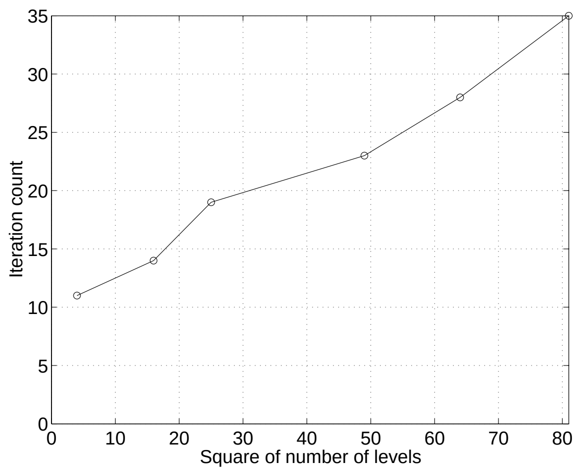
3.5. Multigrid based on the approximate block factorization

Section 2.3 described BFMG, a multigrid method defined using an approximate block factorization. The smoother for BFMG is the block factorization HBBF recursively defined at each level (HBBF smoothing). The smoother may be accelerated by the conjugate gradient method (CG-HBBF smoothing) if necessary. Further, BFMG itself may be used as a preconditioner to the conjugate gradient method (CG-BFMG). Table XI shows iteration counts and timings for BFMG for the UNI7 and ANI7 test problems. The best timings are achieved when BFMG is used as a preconditioner. It is interesting that when CG-HBBF smoothing is used, the total time *decreases* when more smoothing steps are used (up to a limit).

The results show that BFMG timings are somewhat worse than the timings when HBBF is



(a) Isotropic problem, UNI7



(b) Anisotropic problem, ANI7

Figure 1. Plot of iteration count vs. square of number of levels. The plots suggest the linear relationship predicted from theory.

simply used as a preconditioner for the CG method. However, the number of iterations required for convergence can be much lower. Overall, BFMG and CG-BFMG tend to be more scalable in terms of convergence rate, and therefore should be preferred for large problems.

UNI7, $\theta_s = 0.$, $\theta_p = 0.03$

	BFMG HBBF smoothing		BFMG CG-HBBF smoothing		CG-BFMG HBBF smoothing	
smoothing steps	iterations	total time (s)	iterations	total time (s)	iterations	total time (s)
1	40	44.83	36	67.81	12	22.21
2	24	48.36	15	45.53	9	26.81
3	18	52.19	10	41.29	7	29.25
4	14	52.96	6	33.25	6	31.95
5	12	55.62	5	34.48	6	37.04

ANI7, $\theta_s = 0.25$, $\theta_p = 0.1$

	BFMG HBBF smoothing		BFMG CG-HBBF smoothing		CG-BFMG HBBF smoothing	
smoothing steps	iterations	total time (s)	iterations	total time (s)	iterations	total time (s)
1	100	154.50	82	233.81	27	55.66
2	64	183.14	23	105.36	20	71.30
3	49	204.09	15	77.52	17	85.54
4	39	214.31	10	79.51	16	102.62
5	33	224.12	8	76.53	14	110.88

Table XI. Results related to BFMG for the UNI7 and ANI7 test matrices.

3.6. *Elasticity problems*

We conclude this section with some tests to illustrate how the BF, HBBF, and BFMG preconditioners may perform on 3-D finite element elasticity problems. The physical problem is three concentric spherical shells; two steel shells surround a third shell composed of lucite. An octant of these shells is discretized using linear hexahedral elements with one-point integration and hourglass damping. Figure 2 illustrates the gridding of this problem using a very small number of elements. Two test matrices, as listed in Table XII were used. Typical rows in these matrices contain 81 nonzeros per row. We note that for these problems, the CG convergence criterion is the reduction of the residual norm by 8 orders of magnitude.

Problem	*n*	*nnz*
SPH3103	16881	1230831
SPH6206	124839	9586413

Table XII. Two elasticity test problems, showing number of equations *n*, and number of nonzeros *nnz*.

For problems such as these that are derived from systems of PDEs, we consider all couplings between variables of unlike type to be weak. This corresponds to the "unknown" approach described in [29]. Also, in the following tests, we used an incomplete Cholesky factorization to approximately solve with A_{ff}. Using a sparse approximate inverse gave poorer results, but a sparse approximate inverse was still

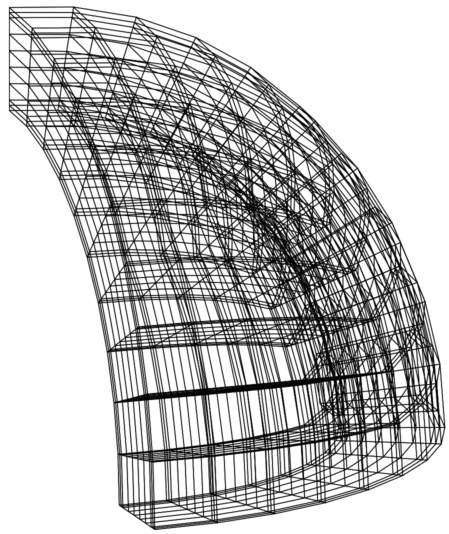


Figure 2. Gridding of an octant of three concentric spherical shells; this is a small example for illustration purposes.

used in the construction of P and $\widehat{P}$.

For matrices with many nonzeros per row, the HB transformed matrices may be very dense and costly to use. This cost can be reduced with large values of the truncation threshold θ_p. In addition, we can use the sparser interpolants, $\mathscr{P}_1$ and $\mathscr{P}_2$, described in section 2.1.2. For the problem SPH3103, Table XIII compares BF and HBBF preconditionings, the latter using the sparser interpolants. Values of θ_s of 0, 0.25, 0.5, 0.75, and 0.95 were tested; the table shows the results using θ_s of 0.25, which were the best for all the preconditioners. Table XIV shows corresponding results for SPH6206.

The results show that the total solution timings for solves with the BF and HBBF preconditioners are comparable. However, as expected, the iteration counts for HBBF are lower. For these matrices coming from discretized elasticity problems, we further expect the results to improve if vectors in the near-nullspace of A (so-called rigid body modes) are preserved in the interpolation. In our setting, this can be ensured if $\mathscr{P}$ interpolates linear functions, that is, a fine degree of freedom or node is interpolated from 2 or 3 strongly coupled coarse nodes in 2-D or 3-D, respectively.

Finally, Table XV shows iteration counts and timings when BFMG is used as a preconditioner. One or two steps of HBBF is used as the smoother for BFMG. The $\mathscr{P}_2$ interpolant was used. Like the results shown earlier, the total time to solution is higher, although the iteration counts are much lower. For the test problem SPH6206, the results were obtained on a slightly slower (1 GHz EV6.8 Alpha) computer with more memory.

4. CONCLUDING REMARKS

This paper has shown that a transformation to a generalized hierarchical basis can lead to improved convergence rates for multilevel block factorization preconditioners. The transformation is simple, but

	θ_p	levels	iterations	time (s)			complexity	
				setup	solve	total	grid	operator
BF	0.3	6	296	20.76	14.08	34.84	1.66	2.60
	0.5	6	485	3.98	17.47	21.45	1.67	1.86
	0.7	6	584	2.55	19.36	21.91	1.67	1.71
	0.9	6	548	1.90	17.01	18.91	1.64	1.53
HBBF	0.3	6	197	4.60	9.82	14.42	1.65	1.90
$\mathscr{P}_1$ interpolant	0.5	6	265	2.77	11.51	14.28	1.66	1.66
	0.7	7	258	2.78	11.09	13.87	1.67	1.63
	0.9	7	311	2.56	13.15	15.71	1.69	1.65
HBBF	0.3	6	135	21.14	9.90	31.04	1.64	2.84
$\mathscr{P}_2$ interpolant	0.5	6	145	8.52	8.26	16.78	1.65	2.15
	0.7	6	190	5.63	9.62	15.25	1.65	1.93
	0.9	6	211	4.83	10.33	15.16	1.65	1.87

Table XIII. Sample results for SPH3103 with BF and HBBF preconditioning. The parameter θ_s was 0.25.

	θ_p	levels	iterations	time (s)			complexity	
				setup	solve	total	grid	operator
BF	0.5	9	781	60.39	269.27	329.66	1.79	2.23
	0.7	8	903	25.72	259.25	284.97	1.74	1.78
	0.9	8	975	16.84	254.29	271.13	1.74	1.58
HBBF	0.5	8	671	33.16	273.23	306.39	1.75	1.76
$\mathscr{P}_1$ interpolant	0.7	8	902	28.54	357.88	386.42	1.74	1.70
	0.9	8	1338	26.90	524.44	551.34	1.74	1.69
HBBF	0.5	8	434	128.74	278.85	407.59	1.74	2.63
$\mathscr{P}_2$ interpolant	0.7	8	501	80.48	265.30	345.78	1.75	2.24
	0.9	8	541	69.58	271.44	341.02	1.75	2.14

Table XIV. Sample results for SPH6206 with BF and HBBF preconditioning. The parameter θ_s was 0.25.

SPH3103

smoothing steps	iterations	total time (s)
1	91	24.73
2	72	34.42

SPH6206

smoothing steps	iterations	total time (s)
1	186	452.70
2	146	653.24

Table XV. Results for CG preconditioned with BFMG using one or two steps of HBBF as the smoother. The parameters θ_s and θ_p were 0.25 and 0.9, respectively.

increases the cost of constructing the preconditioner. The overall time required to solve unstructured isotropic and anisotropic diffusion problems however, is generally reduced.

For matrices with many nonzeros per row, however, the cost of approximate block factorization preconditioners may be very high. This cost is particularly due to the Galerkin approximation for the Schur complement. In these cases, depending upon the size of the problem, BF, HBBF, and BFMG may not be competitive with other, albeit less-scalable, preconditioners.

REFERENCES

1. O. Axelsson and I. Gustafsson. Preconditioning and two-level multigrid methods of arbitrary degree of approximation. *Mathematics of Computation*, 40:219–242, 1983.
2. O. Axelsson and P. S. Vassilevski. Algebraic multilevel preconditioning methods, I. *Numerische Mathematik*, 56:157–177, 1989.
3. R. E. Bank, T. Dupont, and H. Yserentant. The hierarhical basis multigrid method. *Numerische Mathematik*, 52:427–458, 1988.
4. R. E. Bank and T. F. Dupont. Analysis of a two-level scheme for solving finite element equations. Technical Report CNA-159, Center for Numerical Analysis, University of Texas at Austin, 1980.
5. R. E. Bank and R. K. Smith. The incomplete factorization multigraph algorithm. *SIAM Journal on Scientific Computing*, 20:1349–1364, 1999.
6. R. E. Bank and R. K. Smith. An algebraic multilevel multigraph algorithm. *SIAM Journal on Scientific Computing*, to appear.
7. R. E. Bank and C. Wagner. Multilevel ILU decomposition. *Numerische Mathematik*, 82:543–576, 1999.
8. R. E. Bank and J. Xu. The hierarchical basis multigrid method and incomplete LU decomposition. In *Domain Decomposition Methods in Scientific and Engineering Computing: Proceedings of the Seventh International Conference on Domain Decomposition*, volume 180 of *Contemporary Mathematics*, pages 163–173, Providence, Rhode Island, 1994. American Mathematical Society.
9. R. E. Bank and J. Xu. A hierarchical basis multigrid method for unstructured grids. In *Fast Solvers for Flow Problems. Proceedings of the Tenth GAMM-Seminar Kiel*, volume 49 of *Notes on Numerical Mathematics*, pages 1–13. Vieweg-Verlag, Braunschweig, 1995.
10. R. E. Bank and J. Xu. An algorithm for coarsening unstructured meshes. *Numerische Mathematik*, 73:1–36, 1996.
11. T. J. Barth, T. F. Chan, and W.-P. Tang. A parallel non-overlapping domain-decomposition algorithm for compressible flow on triangulated domains. *AMS Contemporary Mathematics Series*, 218:23–41, 1998.
12. M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17:1135–1149, 1996.
13. E. F. F. Botta, A. van der Ploeg, and F. W. Wubs. Nested grids ILU-decomposition (NGILU). *Journal of Computational and Applied Mathematics*, 66:515–526, 1996.
14. E. F. F. Botta and F. W. Wubs. Matrix Renumbering ILU: An effective algebraic multilevel ILU preconditioner for sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 20:1007–1026, 1999.
15. W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM Books, Philadelphia, 2000. Second edition.
16. E. Chow and Y. Saad. Approximate inverse techniques for block-partitioned matrices. *SIAM Journal on Scientific Computing*, 18:1657–1675, 1997.
17. E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, 19:995–1023, 1998.
18. J. D. F. Cosgrove, J. C. Díaz, and A. Griewank. Approximate inverse preconditioning for sparse linear systems. *International Journal of Computer Mathematics*, 44:91–110, 1992.
19. L. Grosz. Preconditioning by incomplete block elimination. *Numerical Linear Algebra with Applications*, 7:527–541, 2000.
20. M. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18:838–853, 1997.
21. V. E. Henson and U. M. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, to appear.
22. T. Huckle. Sparse approximate inverses and applications. In *International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Industrial Applications*, Minneapolis, MN, June 10–12, 1999.
23. L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings I. Theory. *SIAM Journal on Matrix Analysis and Applications*, 14:45–58, 1993.
24. Y. Notay. Using approximate inverses in algebraic multilevel methods. *Numerische Mathematik*, 80:397–417, 1998.

25. Y. Notay. Optimal order preconditioning of finite difference matrices. *SIAM Journal on Scientific Computing*, 21:1991–2007, 2000.
26. M. E. G. Ong. Hierarchical basis preconditioners in three dimensions. *SIAM Journal on Scientific Computing*, 18:479–498, 1997.
27. A. Reusken. A multigrid method based on incomplete Gaussian elimination. *Numerical Linear Algebra with Applications*, 3:369–390, 1996.
28. A. Reusken. On the approximate cyclic reduction preconditioner. *SIAM Journal on Scientific Computing*, 21:565–590, 1999.
29. J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.
30. Y. Saad. ILUM: A multi-elimination ILU preconditioner for general sparse matrices. *SIAM Journal on Scientific Computing*, 17:830–847, 1996.
31. Y. Saad and B. Suchomel. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. Technical Report UMSI 99/107, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1999.
32. Y. Saad and J. Zhang. BILUM: block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing*, 20:2103–2121, 1999.
33. Y. Saad and J. Zhang. Diagonal threshold techniques in robust multi-level ILU preconditioners for general sparse linear systems. *Numerical Linear Algebra with Applications*, 6:257–280, 1999.
34. Yousef Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Co., Boston, MA, 1996.
35. K. Stüben. Algebraic multigrid (AMG): An introduction with applications. Technical Report 53, GMD, St. Augustin, 1999.
36. C. Wagner, W. Kinzelbach, and G. Wittum. Schur-complement multigrid: a robust method for groundwater flow and transport problems. *Numerische Mathematik*, 75:523–545, 1997.
37. H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.