MULTI-RESOLUTION APPROXIMATE INVERSE PRECONDITIONERS[*]

ROBERT BRIDSON[†] AND WEI-PAI TANG[‡]

Abstract. We introduce a new preconditioner for elliptic PDE's on unstructured meshes. Using a wavelet-inspired basis we compress the inverse of the matrix, allowing an effective sparse approximate inverse by solving the sparsity vs. accuracy conflict. The key issue in this compression is to use *second-generation* wavelets which can be adapted to the unstructured mesh, the true boundary conditions, and even the PDE coefficients. We also show how this gives a new perspective on multiresolution algorithms such as multigrid, interpreting the new preconditioner as a variation on node-nested multigrid. In particular, we hope the new preconditioner will combine the best of both worlds: fast convergence when multilevel methods can succeed, but with robust performance for more difficult problems.

The rest of the paper discusses the core issues for the preconditioner: ordering and construction of a factored approximate inverse in the multiresolution basis, robust interpolation on unstructured meshes, automatic mesh coarsening, and purely algebraic alternatives. Some exploratory numerical experiments suggest the superiority of the new basis over the standard basis for several tough problems, including discontinuous anisotropic coefficients, strong convection, and indefinite reaction problems on unstructured meshes, with scalability like hierarchical basis methods achieved.

Key words. preconditioner, multilevel, multigrid, hierarchical basis, unstructured mesh, elliptic PDE's

AMS subject classifications. 65F10, 65F50

1. Motivation.

Approximate inverses are becoming increasingly popular preconditioners for the iterative solution of large sparse linear systems. The main reason is that they can be efficiently applied (with just matrix-vector products) on high performance hardware; they are also a valuable general-purpose alternative to ILU for tough problems where ILU breaks down from instabilities.

Several algorithms for computing sparse approximations to $\mathbf{A}^{-1}$, or to its inverse triangular factors $\mathbf{L}^{-1}$ and $\mathbf{U}^{-1}$, have been proposed: e.g. [5, 6, 7, 19, 27, 30, 36]. Unfortunately, for linear systems arising from elliptic PDE's, there appears to be an inherent problem in the explicit nature of these preconditioners, a fundamental conflict between accuracy and sparsity. As problem sizes increase, their performance (either in terms of convergence rate at a fixed number of nonzeros per row, or storage required for a fixed convergence rate) quickly decreases.[1]

For a simple heuristic analysis of this problem, ignore boundary conditions. Suppose the elliptic PDE $\mathcal{L}u = f$ on domain Ω is discretized to $\mathbf{Au} = \mathbf{f}$ on points x_1, …, x_n in Ω, where the matrix $\mathbf{A}$ is the discrete form of the elliptic operator $\mathcal{L}$, and $u_i \approx u(x_i)$. (The discretization may use finite elements, finite volumes, or any other reasonable scheme.) The continuous solution may be written with the Green's

[*] This work was supported by the Natural Sciences and Engineering Council of Canada, and Communications and Information Technology Ontario (CITO), funded by the Province of Ontario.

[†]SCCM, Gates 2B, Stanford University, Stanford, CA 94305, USA. email: rbridson@stanford.edu

[‡]Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada. email: wptang@elora.uwaterloo.ca

[1]Of course, switching to an implicit preconditioner isn't guaranteed to solve the scalability problem; for example, standard ILU does not scale any better than no preconditioner, and the question of how best to use matrix orderings, dropping strategies, and numerical modifications to improve the scalability of ILU is still open [3, 4, 9, 14, 32, 33, 34].

function $\mathcal{G}(x,y)$ on $\Omega \times \Omega$ satisfying $\mathcal{L}\mathcal{G}(x,y) = \delta(x-y)$:

$$u(x) = \int_{\Omega} \mathcal{G}(x,y) f(y)\, dy$$

(assuming this exists, as one would expect for an elliptic problem well-posed enough to permit numerical solution). The discrete solution is similarly found with the matrix $\mathbf{A}^{-1}$ satisfying $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$:

$$u_i = \sum_{j=1}^{n} A^{-1}_{ij} f_j$$

Through this analogy, it is clear that $\mathbf{A}^{-1}$ is the discrete approximation to the Green's function.

Unfortunately, it is well known that though the Green's function decays away from its diagonal singularity, the decay may be slow especially for convective or indefinite problems (in fact, for problems with Neumann boundaries, $\mathcal{G}$ might not even decay to zero at all). The decay of the true Green's function is independent of the mesh size in the discretization, and so as the mesh is refined the number of large nonzeros in $\mathbf{A}^{-1}$ also increases, roughly like $O(n^{1+1/d})$ where n is the number of mesh nodes and d is the dimensionality of the problem.[2] This means an approximate inverse preconditioner cannot scale effectively with the problem size.[3]

To be more precise, one cannot scale in the standard basis, where u_i approximates $u(x_i)$. In [11, 18], the realization that the Green's function is smooth away from the diagonal suggested wavelets as alternate bases: they can compress smooth functions into high quality sparse approximations, handle non-smooth points (e.g. at the diagonal singularity, or arising from discontinuous coefficients), and provide fast and parallelizable transforms to and from the standard basis. The finer the mesh, the better the smoothness of $\mathcal{G}(x,y)$ can be exploited for compression, so the preconditioner may scale much more effectively.

The original paper [18] considered only classical wavelets, treating the discrete Green's function as a two-dimensional, regularly sampled, periodic image. For problems of dimensions greater than one (so the Green's function is of dimension greater than two) on irregular meshes with non-periodic boundaries, this leads to significant problems. In particular, these "first generation" wavelets are constructed on regularly sampled one-dimensional periodic domains, and so cannot hope to perform well on data coming from more complicated situations. This motivated the use of second generation wavelets[35] in [11] that naturally match such domains, while retaining the attractive properties of compression, tolerance of singularities, and fast transforms. The present article describes the multiresolution approximate inverse preconditioner of [11] and more recent developments.

Other authors have proved that discretizing elliptic PDE's with a wavelet basis in the finite element method (and using point or block diagonal preconditioners)

[2]There are n columns in $\mathbf{A}^{-1}$, with column i containing a discrete approximation to $G(\cdot, x_i)$. There is a d-dimensional subregion $\Omega_L \subset \Omega$ s.t. $G(x, x_i)$ is large for $x \in \Omega_L$. There are $O(n^{1/d})$ mesh nodes contained in Ω_L, so column i of $\mathbf{A}^{-1}$ has roughly $O(n^{1/d})$ large entries, for a total of $O(n^{1+1/d})$ in $\mathbf{A}^{-1}$.

[3]Of course, this reasoning only applies directly to fully explicit approximate inverses: preconditioners in factored form perhaps may be more effective, since their product may be dense, though at the time of writing this has yet to be demonstrated.

FIG. 2.1. *The multiresolution approximate inverse method*

The initialization phase

- Determine multiresolution bases α and β
- Use a standard operator-based approximate inverse algorithm to find $\widetilde{\mathbf{Q}} \approx (\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})^{-1}$

The application phase (called by a Krylov solver)

- Apply preconditioner to x by $y = \mathbf{M}_\alpha^{-1}(\widetilde{\mathbf{Q}}(\mathbf{M}_\beta^{-T}x))$

can give optimal scalability, primarily working with classical wavelets and generalizations to handle boundaries and even dyadically-refined unstructured meshes (e.g. [15, 16, 21, 22, 23, 24]) but also e.g. approximate wavelets derived from an existing hierarchical basis [37, 38]. The work in [21] actually worked with a full approximate inverse rather than just diagonal preconditioners, and found significant improvements by taking into account interactions between different levels. The present paper differs by combining second-generation wavelets with a more sophisticated approximate inverse, with an emphasis on heuristically finding good multilevel algorithm components (regrettably without yet theoretical results to confirm the promising first few numerical experiments).

2. Outline of the Method.

Since the Green's function is defined on the product space $\Omega \times \Omega$, it is natural to look for a wavelet basis that is a tensor product $\alpha \otimes \beta$ of wavelet bases α and β on Ω. In the discrete case, this means representing $\mathbf{A}^{-1}$ as

$$A_{ij}^{-1} = \sum_{k=1}^{n}\sum_{l=1}^{n} Q_{kl} a_i^k b_j^l$$

where the separable basis functions are the product of elements $a^k \in \alpha$ and $b^l \in \beta$, and the coefficients are stored in matrix $\mathbf{Q}$. Equivalently,

$$\mathbf{A}^{-1} = \mathbf{M}_\alpha^{-1}\mathbf{Q}\mathbf{M}_\beta^{-T}$$

where $\mathbf{M}_\alpha^{-1}$ and $\mathbf{M}_\beta^{-1}$ have the basis functions of α and β respectively as their columns. Applying these operators to the standard basis vectors shows that $\mathbf{M}_\alpha$ and $\mathbf{M}_\beta$ are the transforms from the wavelet bases to the standard basis.

The transformed $\mathbf{A}^{-1}$, ready for compression, is $\mathbf{Q} = \mathbf{M}_\alpha\mathbf{A}^{-1}\mathbf{M}_\beta^{T}$. For the preconditioner, a highly sparse approximation $\widetilde{\mathbf{Q}}$ will be used. Rewriting $\mathbf{Q} = (\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})^{-1}$ shows that $\widetilde{\mathbf{Q}}$ can be obtained by applying a standard sparse approximate inverse algorithm to the transformed operator $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$—in particular, without knowledge of the true inverse $\mathbf{A}^{-1}$. Note that to avoid forming $(\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})$ explicitly, which may incur significant fill-in, an approximate inverse algorithm that works on a linear operator (not necessarily a matrix) is required. One example, used in this research, is SAINV[7].

To summarize, an overview of the multiresolution approximate inverse method is given in figure 2.1.

3. The General Algorithm.

FIG. 3.1. *The transform algorithms of the Lifting Scheme*

The forward transform (standard basis $\rightarrow$ multiresolution basis)

```
• Start with the function values $f_1, \ldots, f_n$ at sample points $x_1, \ldots, x_n$.
• Let $\lambda_i^0 = f_i$ for all $i$, $\mathcal{C}^0 = \{x_1, \ldots, x_n\}$, and $j = 0$.
• Begin loop:
    • Split up the sample points $\mathcal{C}^j$ into two disjoint subsets, the fine nodes $\mathcal{F}^{j+1}$ and the coarse nodes $\mathcal{C}^{j+1}$.
    • Predict $\lambda_F^j$, the values at the fine nodes, from $\lambda_C^j$, the values at the coarse nodes, with some linear prediction operator $\mathbf{P}^j$: $\lambda_F^j \approx \mathbf{P}^j \lambda_C^j$.
    • Store the wavelet coefficient $\gamma_i^{j+1} = \lambda_i^j - (\mathbf{P}^j \lambda_C^j)_i$ for each fine node $x_i \in \mathcal{F}^{j+1}$.
    • Update the value at each coarse node by $\lambda_i^{j+1} = \lambda_i^j + (\mathbf{U}^j \gamma^{j+1})_i$ for each $x_i \in \mathcal{C}^{j+1}$ so that the required moments will be preserved. This update operator $\mathbf{U}^j$ must also be linear.
    • If $|\mathcal{C}^{j+1}|$ is small enough, below some constant, break out of the loop. Otherwise, set $j \leftarrow j+1$ and continue.
• Return $\lambda^j$ from the coarsest level along with the wavelet coefficients $\gamma^1, \ldots, \gamma^j$ from each level.
```

The inverse transform (multiresolution basis $\rightarrow$ standard basis)

```
• Start with $\lambda^j$ and the wavelet coefficients $\gamma^1, \ldots, \gamma^j$.
• Begin loop:
    • Reconstruct $\lambda_C^{j-1}$ at the coarse nodes by $\lambda_i^{j-1} = \lambda_i^j - (\mathbf{U}^{j-1}\gamma^j)_i$ for each $x_i \in \mathcal{C}^j$.
    • Reconstruct $\lambda_F^{j-1}$ at the fine nodes by $\lambda_i^{j-1} = \gamma_i^j + (\mathbf{P}^{j-1}\lambda_C^{j-1})_i$ for each $x_i \in \mathcal{F}^j$.
    • Continue with $j \leftarrow j-1$ until $j = 1$.
• Return $f_i = \lambda_i^0$ for all i.
```

3.1. The Basis Construction. The goal of the new basis $\alpha \otimes \beta$ is to convert "smoothness" in the standard basis to small coefficients that can be accurately approximated by zero. For irregular domains, the lifting scheme[35] for second generation wavelets is a natural choice. In this scheme, the basis is not constructed explicitly but rather the forward transform algorithms (from the standard basis to the multiresolution basis, called $\mathbf{M}_\alpha$ and $\mathbf{M}_\beta$ above) are designed to directly achieve good compression along with easy invertibility. Figure 3.1 gives summaries of the transform algorithms.

The essential idea is that where a function is smooth, its values can be accurately predicted from nearby neighbours, and so storing the prediction error results in small coefficients except near "rough" regions. Doing this in a hierarchy of levels gives rise to a multiresolution representation: wavelet coefficients at level j correspond to features on the scale of the grid resolution of the set of nodes $\mathcal{C}^j$ at level j. It should be noted that on unstructured meshes (and possibly for other reasons mentioned below) the prediction operator for each fine node may have different weights—unlike classical wavelets, where one set of convolution weights is used throughout the domain.

The update step is required in signal processing to prevent aliasing, where for example a high resolution singularity is propagated unchanged to lower resolutions; the update step is an averaging designed to make sure the signal is smooth enough to be faithfully represented at the next coarser level. However, in this context of compressing discrete Green's functions, this seems to be a liability as demonstrated in [11]. In the ideal case, all wavelet coefficients in $\mathbf{Q}$ will be very small except on the diagonal, where the discrete Green's function must have a singularity: finding a sparse approximate inverse for a nearly diagonal matrix is simple. However, the update step smooths out this inherent singularity, smearing the large diagonal entries to off-diagonals, resulting in a harder task for a sparse approximate inverse algorithm.

On the other hand, without the update step the basis can be viewed as just a generalization of the standard hierarchical basis[1, 40] (if regular refinement and linear interpolation for prediction are used, it is exactly the hierarchical basis; see section 4 for more details). While optimal scalability has been demonstrated with classical wavelets in [15, 16, 21, 22, 23, 24], it is well known that with just diagonal or block diagonal preconditioning the hierarchical basis is not optimal: the condition number of the preconditioned system slowly grows with the number of unknowns, particularly in higher dimensions. To get the optimal performance of methods such as multigrid, the basis must be stabilized[37, 38], making the basis functions from different levels at least approximately orthogonal. This is essentially what the update step does, smoothing the function at each level so that coarser levels don't see the high resolution features picked up at finer levels. Thus perhaps a theoretical analysis, beyond the scope of this paper, will show that the update step can be of value for multiple dimensions. However, the issue is further complicated by the fact that approximate inverses can be more effective in higher dimensions, where the Green's function decays faster and a sparse approximation is more feasible. For the rest of this paper we will not use the update step, leaving these questions for future research.

Without the update step, the forward transform is simplified, and in fact all wavelet coefficients at all levels can be computed simultaneously. The forward transform $\mathbf{M}$ can easily be written explicitly as a triangular matrix multiply and the inverse transform as a triangular solve, as in figure 3.2. As long as each prediction operation can be done in constant time, i.e. each $\mathbf{P}^j$ has a bounded number of nonzeros per row, it is clear that the forward and inverse transforms can both be done in $O(n)$ time in serial. In terms of parallel computation, the forward transform is as good as a single sparse matrix multiply, whereas the inverse transform can naturally be done in $O(\log n)$ steps (with smaller and smaller matrix multiplies) assuming a geometric decline in the size of the $\mathcal{C}^j$.

There are two big issues that need to be resolved when constructing the basis. The first is how to coarsen; how to select the sets $\mathcal{C}^j$ of coarse nodes at each level. So far, we have concentrated on independent set heuristics similar to those used in unstructured multigrid[17] and algebraic multigrid[31]. The second issue is how to define the prediction operators $\mathbf{P}^j$ at each level. Standard linear interpolation or higher order polynomial interpolation is a possibility, but for robustness in difficult problems we have found more sophisticated techniques are necessary[11].

Looking at the compressed inverse $\mathbf{Q} = \mathbf{M}_\alpha \mathbf{A}^{-1} \mathbf{M}_\beta^T$, notice that the transform $\mathbf{M}_\alpha$ is being applied to each of the columns of $\mathbf{A}^{-1}$. From the equation $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$, observe that column i of $\mathbf{A}^{-1}$ is actually the discrete solution to the PDE $\mathcal{L}u(x) = \delta(x - x_i)$. Thus the functions compressed by α satisfy the homogeneous PDE $\mathcal{L}u = 0$ almost everywhere, which we can take as our definition of "smooth". Predicting the value at a fine node i from nearby coarse nodes can be done by solving a discrete form of

FIG. 3.2. *The multiresolution basis transforms without update steps expressed in terms of triangular matrices*

The forward transform (standard basis → multiresolution basis)

$$\begin{pmatrix}\gamma^1\\ \gamma^2\\ \vdots\\ \gamma^j\\ \lambda^j\end{pmatrix} = \mathbf{Mf} = \left(\begin{array}{c|c} \mathbf{I} & -\mathbf{P}^1 \\ \hline & \begin{array}{c|c} \mathbf{I} & -\mathbf{P}^2 \\ \hline & \begin{array}{c|c} \ddots & \vdots \\ \hline & \begin{array}{cc} \mathbf{I} & -\mathbf{P}^j \\ & \mathbf{I} \end{array} \end{array} \end{array} \end{array}\right) \begin{pmatrix}\mathbf{f}_{F1}\\ \mathbf{f}_{F2}\\ \vdots\\ \mathbf{f}_{Fj}\\ \mathbf{f}_{Cj}\end{pmatrix}$$

The inverse transform (multiresolution basis → standard basis)

$$\mathbf{f} = \mathbf{M}^{-1}\gamma = \left(\begin{array}{c|c} \mathbf{I} & -\mathbf{P}^1 \\ \hline & \begin{array}{c|c} \ddots & \vdots \\ \hline & \begin{array}{cc} \mathbf{I} & -\mathbf{P}^j \\ & \mathbf{I} \end{array} \end{array} \end{array}\right)^{-1} \begin{pmatrix}\gamma^1\\ \vdots\\ \gamma^j\\ \lambda^j\end{pmatrix}$$

$\mathcal{L}u(x_i) = 0$ using the coarse nodes as specified "boundary" data.

Similarly, the β transform is applied to the rows of $\mathbf{A}^{-1}$, which are discrete solutions to the adjoint problems, so β may be constructed with the adjoint operator $\mathcal{L}^*$ in mind. In particular, for self-adjoint problems it makes sense to take $\alpha = \beta$; for highly non-self-adjoint problems it will be important to have $\alpha \neq \beta$.

Finally it should be noted that for some problems—e.g. with oscillatory coefficients, strong indefiniteness, or complicated convection streamlines—it may be too difficult to construct very coarse yet useful representations of the Green's function. Though ideas from homogenization theory may help, it's likely that there will be a lower limit to the resolutions that are useful to consider. In this case, it is probably wisest to limit the multiresolution bases to a few levels and instead concentrate resources on the approximate inverse.

3.2. The Approximate Inverse.

The transformed operator $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$ may be multiplied out explicitly at which point any approximate inverse algorithm may be used. However, in doing so substantial extra fill-in is incurred, increasing the cost of the preconditioner construction and application as well as storage requirements. A more attractive route is to use an approximate inverse algorithm that doesn't require explicit knowledge of the matrix, and thus can precondition an operator known only in this factored form.

Many popular algorithms can, in their simplest form, be adapted to this context. One example is the MR method of [19] which requires no modification. However, other approximate inverses require more thought. For example, a sparsity pattern must be specified *a priori* for some methods, and it is not clear how to do so for efficient and robust performance here. Avoiding this issue, we have chosen to adopt SAINV[5, 6, 7]

FIG. 3.3. *The SAINV algorithm, using MATLAB colon notation for submatrices.*

```
• Take as input matrix $\mathbf{A}$ of dimension $n$ and a drop tolerance $\delta$, generally around 0.1 or 0.01.
• Set $\mathbf{W} \leftarrow \mathbf{I}$ and $\mathbf{Z} \leftarrow \mathbf{I}$.
• for $j = 1$ to $n$ do
    • Compute $l = \mathbf{A}Z_{:,j}$ and $u = \mathbf{A}^T W_{:,j}$.
    • Set $D_{jj} \leftarrow (u^T Z_{:,j})^{-1}$, the inverse pivot.
    • Rescale $L = lD_{jj}$ and $U = uD_{jj}$.
    • Compute the biconjugation coefficients
      $C_W = L^T W_{:,j+1:n}$ and $C_Z = U^T Z_{:,j+1:n}$.
    • Update the remaining columns of $\mathbf{W}$ and $\mathbf{Z}$ with sparsified outer-products (i.e. only updating with the entries of magnitude greater than $\delta$): $W_{:,j+1:n} \leftarrow W_{:,j+1:n} - \textit{sparsify}(W_{:,j}C_W)$
      and $Z_{:,j+1:n} \leftarrow Z_{:,j+1:n} - \textit{sparsify}(Z_{:,j}C_Z)$
```

for this first study, which constructs the sparsity pattern during construction. In particular we use an outer-product based version of the algorithm[13] that doesn't require any knowledge of the sparsity pattern of the operator for efficient performance. We note that for efficiency during the construction phase, the basis transforms and matrix multiplies must be done in fully sparse mode. Figure 3.3 gives the algorithm, which by biconjugation (along the lines of Modified Gram-Schmidt) applied to two copies of the identity matrix constructs upper triangular matrices $\mathbf{W}$ and $\mathbf{Z}$ and a diagonal matrix $\mathbf{D}$ such that $\mathbf{A}^{-1} \approx \mathbf{ZDW}^T$. It can be simplified in the case of symmetric matrices to construct just $\mathbf{Z}$ (which is equal to $\mathbf{W}$) with half the work and storage. It has the advantage that for positive definite matrices it is guaranteed to produce a positive definite preconditioner, though breakdown is possible in the general case, and has generally been shown to be very robust[7].

One important issue for factored approximate inverses is the ordering of the rows and columns of the matrix. As demonstrated in [8, 12, 13], performance can be significantly improved by an appropriate reordering—e.g. nested dissection (we use the Metis routine[29]). On the other hand, one might argue that if the multiresolution bases here are constructed correctly, the transformed $\mathbf{A}$ will be well enough conditioned that ordering isn't needed. However, it seems doubtful that the multiresolution framework will be robust enough to handle all problems on its own. What we desire for tough problems is a multiresolution basis construction algorithm which "fails gracefully", i.e. never makes $\mathbf{A}$ worse conditioned even though it may not provide adequate improvement. In this case, the power of the approximate inverse will hopefully show through, provided we have taken care of the ordering.

3.2.1. Ordering.

Unfortunately, typical ordering algorithms require the explicit structure of the matrix so this is a nontrivial step in this context; some analysis is required.

Before going further, recall the graph theory notation often used in sparse matrix ordering. With a given $n \times n$ matrix $\mathbf{B}$, associate the graph $G_{\mathbf{B}}$, or simply G if the context makes it clear, defined on nodes $\{1, \dots, n\}$ with a directed edge $i \to j$ if and only if $B_{ij} \neq 0$ (for $i \neq j$). Thus the nonzero structure of $\mathbf{B}$ and the graph $G_{\mathbf{B}}$ may be identified. As an abbreviation, write $i \to j$ to mean the statement that the directed edge $i \to j$ exists in G. The neighbourhood of a node i is the set of nodes j such that $i \to j$. A path is a sequence of distinct nodes $i_1, \ldots, i_k$ such that $i_1 \to i_2$, $i_2 \to i_3, \ldots$, and $i_{k-1} \to i_k$, often written $i_1 \to \cdots \to i_k$, or simply $i_1 \rightsquigarrow i_k$. The transitive closure G^* of a graph G is one constructed on the same nodes but having $i \to j$ whenever $i \rightsquigarrow j$ in G. For a fuller treatment, see [25, 26].

As is shown in [26], assuming here and for the rest of this section that there is no felicitous cancellation, the structure of $\mathbf{B}^{-1}$ is given by the transitive closure of the graph of $G_{\mathbf{B}}$. As shown before, the forward transform $\mathbf{M}_\alpha$ can be simply expressed as a triangular matrix (when there are no update steps). Then the graph of $\mathbf{M}_\alpha$ satisfies $i \to j$ if and only if at some level i is a fine node whose prediction uses coarse node j. Therefore the graph of $\mathbf{M}_\alpha^{-1}$ has $i \to j$ if and only if there is a chain of prediction dependencies $i \rightsquigarrow j$.

Define the support of a node j to be the set $supp(j)$ of nodes i such that $(\mathbf{M}_\alpha^{-1})_{ij} \neq 0$—this is actually the support of the j'th multiresolution basis function. From the transitive closure characterization of inverses, observe that the supports have a nested structure: if $i \in supp(j)$ then $supp(i) \subset supp(j)$. Notice that if j is a fine node at the highest resolution level, $supp(j) = \{j\}$, but that if j is at the lowest resolution level its support may be very dense—showing that it is important to not multiply out the inverse transform explicitly.

Now examine the structure of $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$. Assume that $\mathbf{A}$ has symmetric structure ($A_{ij} \neq 0$ if and only if $A_{ji} \neq 0$) and $\mathbf{M}_\beta$ and $\mathbf{M}_\alpha$ have the same structure, i.e. that the two bases have the same hierarchy of levels and the same prediction dependencies[4]. Then the product has symmetric structure, and one can speak unambiguously about coarse/fine nodes and the support of a node. Observe

$$\begin{aligned}(\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})_{ij} &= \sum_{k=1}^{n}\sum_{l=1}^{n}(\mathbf{M}_\beta^{-T})_{ik}A_{kl}(\mathbf{M}_\alpha^{-1})_{lj} \\ &= \sum_{k=1}^{n}\sum_{l=1}^{n}(\mathbf{M}_\beta^{-1})_{ki}A_{kl}(\mathbf{M}_\alpha^{-1})_{lj}\end{aligned}$$

Then $(\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1})_{ij} \neq 0$ if and only if there exist nodes k and l with $k \in supp(i)$, $l \in supp(j)$, and $k \to l$ in $\mathbf{A}$. In other words, $i \to j$ in the product if and only if their supports are adjacent in $\mathbf{A}$. Using the nested structure of the supports, it is then clear that the neighbourhood of any node j contains the neighbourhoods of all nodes in $supp(j)$.

Now, the location of nonzeros in column i of the upper inverse triangular factor $\mathbf{Z}$ of a symmetrically-structured matrix $\mathbf{B}$ can be characterized as follows[12]: $\mathbf{Z}$ has a nonzero for each node before i and reachable from i via paths in $\mathbf{B}$ using nodes before i.

Consider the effect of swapping the positions of $i \neq j$ in some ordering, when $i \in supp(j)$. Clearly the number of nonzeros in columns in $\mathbf{Z}$ ordered before both i and j or after both will not be changed. However, the columns in between may be altered. Since the neighbourhood of j contains the neighbourhood of i, any nodes reachable on paths through i are reachable through j, but not necessarily the other

[4]So far, relatively good results have been obtained under this assumption, which makes the following analysis much easier. However, it may prove useful to relax this requirement for convection-dominated problems, where predicting from the upwind nodes suggests that the structure of M_α and M_β should be different: convection for the adjoint problem (handled by β) is in the opposite direction from the original problem (handled by α).

FIG. 3.4. *Modifying an ordering to respect the multiresolution basis.*

```
• Take as input the structure of $\mathbf{M}_\alpha$ or $\mathbf{M}_\beta$ (multiplied out).
• For $i = 1, \dots, n$
    • Set $numdep(i)$ = number of nodes $j$ with $j \to i$, not including $i$ itself.
    • Set $waiting(i)$ to false.
• Initialize a queue with room for $n$ entries, empty at first.
• Set $i = 1$, the first node to attempt to order.
• Set $j = 1$, the first index into the modified ordering $p$.
• While $j \le n$
    • If the queue is not empty then
        • Remove the first node $k$ from the front of the queue.
        • Set $p_j = k$ and $j \leftarrow j + 1$.
        • Consider, in order, each $l \neq k$ with $k \to l$ and $waiting(l)$ true;
          decrement $numdep(l)$, and if this is 0 set $waiting(l)$ to false and
          append $l$ to the queue.
    • Else if $numdep(i) = 0$ then
        • Set $p_j = i$, $j \leftarrow j + 1$, and $i \leftarrow i + 1$.
        • Consider, in order, each $l \neq i$ with $i \to l$ and $waiting(l)$ true;
          decrement $numdep(l)$, and if this is 0 set $waiting(l)$ to false and
          append $l$ to the queue.
    • Else ($numdep(i) > 0$)
        • Set $waiting(i)$ to true, and $i \leftarrow i + 1$.
• Return the modified ordering $p$.
```

way around. Therefore ordering i before j can't result in more nonzeros in $\mathbf{Z}$, but putting j before i might.

Thus any ordering of the nodes should respect j ordered after all other nodes in *supp*(j). Since *supp*(j) is the set of i such that $(\mathbf{M}_\alpha^{-1})_{ij} \neq 0$, this is equivalent to requiring that i be ordered before j whenever $i \rightsquigarrow j$ in $\mathbf{M}_\alpha$. This is clearly equivalent to ordering i before j whenever $i \to j$ in $\mathbf{M}_\alpha$, which can be enforced by the algorithm in figure 3.4.

Essentially the algorithm outputs the nodes in the existing order except when a coarse node comes before any of its fine dependents. Then the coarse node is made to wait until all the fine dependents have been ordered, at which point it's put on a queue to be ordered as soon as possible. The value *numdep*(i) serves as a counter of how many fine nodes dependent on i have yet to be ordered—since i is only put into p when this reaches zero, the ordering must be consistent.

The initialization loop, assuming sparse storage of the matrix, takes time on the order of the number of nonzeros in the matrix, which should be $O(n)$. The complexity of the main loop is a little more difficult to prove:

First note that both i and j begin at 1 and never are decremented. Let $d = \sum_{i=1}^{n} numdep(i)$, so before the main loop begins $d = nnz(\mathbf{M}_\alpha) - n$, the number of off-diagonal nonzeros in $\mathbf{M}_\alpha$. Values in *numdep* are never incremented so d never increases.

A node can only be marked as waiting in the final else clause, and since i is incremented there it can never be marked as waiting again. The only way an entry in *numdep* can be decremented to zero is if it had been marked as waiting, and when it hits 0 its marked as not waiting, so it can never be decremented past 0. Therefore d is always non-negative.

Suppose i is incremented past $n+1$—this can only happen if $i = n+1$ at the start of an iteration with the queue empty. There must be some unordered nodes left, as otherwise j would have been incremented past n and the loop would have stopped. If any of the unordered nodes had *numdep* equal to zero, they either would have started at zero, in which case the first else clause would have been executed for that value of i, or they would have been decremented to zero and added to the queue—in either case implying that they must now be ordered, a contradiction. Thus all the unordered nodes have positive *numdep* counters. However, some unordered node v must be from the finest resolution level of all unordered nodes, and so cannot have any unordered dependent fine nodes—and so must have $\mathit{numdep}(v) = 0$, a contradiction. Therefore i never is incremented past $n+1$.

Clearly j can never be incremented past $n+1$ thanks to the loop condition. Therefore, since in each iteration either j is incremented, i is incremented, or at least one of the values in *numdep* is decremented, there can be at most $n + \mathit{nnz}(\mathbf{M}_\alpha)$ iterations. In fact, assuming constant time queue operations (e.g. as in a simple array implementation) the time spent in the main loop is $O(n) + O(\mathit{nnz}(\mathbf{M}_\alpha))$, which again should be $O(n)$. Thus the entire algorithm runs in $O(n)$ time.

Now consider the following simple scheme: order $\mathbf{A}$ with Nested Dissection, and then run the above algorithm to make the ordering consistent with the multiresolution basis. The only worry is that the modification will destroy the good fill-reducing qualities of the original ordering. However, the bulk of the nodes should be at the finest level and thus have trivial supports, so the modification can't change their relative order. The only nodes that can be greatly affected by the ordering modification are the very coarse nodes, which are in a very small minority. Thus the potential damage is very limited. Experiments have confirmed that this isn't much worse (but far cheaper) than applying Nested Dissection to the multiplied out $\mathbf{M}_\beta^{-T}\mathbf{A}\mathbf{M}_\alpha^{-1}$.

4. Relationships with Other Methods.

Before proceeding to our actual implementation and testing for unstructured two-dimensional problems, it is instructive to compare the new algorithm with some other multiresolution methods.

As mentioned before, the basis transforms can be expressed as triangular matrices with unit block diagonals, so the algorithm could be viewed as a highly-parallel variant of multilevel ILU (e.g. [4, 3, 9, 32, 33, 34]) with an approximate inverse replacing $\mathbf{D}^{-1}$ for the approximate $\mathbf{LDU}$ factorization.

Another viewpoint comes from noting that the operators $\binom{\mathbf{P}_\alpha}{\mathbf{I}}$ and $(\mathbf{P}_\beta^T \; \mathbf{I})$ within the transforms for α and β correspond to node-nested multigrid's prolongation and restriction respectively. The application of the preconditioner then can be thought of as the multigrid-like algorithm in figure 4.1. The key difference between this and multigrid is that the smoothing is performed in one step, and only at the coarsest level for each variable, instead of being interleaved with restriction and prolongation. (See [36] for an example of approximate inverses used as smoothers in multigrid.) This is similar to but not exactly the same as additive multigrid, i.e. BPX[10].

The hierarchical basis preconditioners (e.g. [1, 40]) are very similar to the new preconditioner. In these, the original system is transformed into a new multiresolution basis, and a simple preconditioner such as block Jacobi is applied. Extending this, the multiresolution approximate inverse naturally works with unsymmetric bases ($\alpha \neq$

FIG. 4.1. *A multigrid-like interpretation of the multiresolution approximate inverse algorithm.*

- Successively restrict the function to coarser and coarser levels by $\mathbf{M}_\beta^{-T}$.
- Smooth all variables at their coarsest level only—including couplings between variables at different levels—by $\widetilde{\mathbf{Q}}$, possibly doing an exact solve on the coarsest level if the approximate inverse is dense enough there.
- Prolong the smoothed multiresolution representation back to the original variables by $\mathbf{M}_\alpha^{-1}$.

β) better adapted to the problem and also allows for coupling between variables at different levels in the preconditioner $\widetilde{\mathbf{Q}}$.

More sophisticated multiresolution bases (but otherwise essentially the same algorithm as the hierarchical basis method) are used in wavelet methods, e.g. [15, 16, 21, 22, 23, 24, 37, 38]. In particular, these bases are more stable than simple hierarchical bases, in the sense that the the multiresolution norm is equivalent to the standard norm, which results in optimal scalability. As mentioned before when discussing the update step in the present method this issue hasn't been resolved here, and it appears that we currently only achieve the suboptimal scalability of the hierarchical basis method.

Finally, there have already been proposed wavelet–approximate inverse combinations in [18] and [21]. Both of these works used classical wavelets, though the latter featured a generalization which correctly treats non-periodic boundary conditions.

5. Implementation.

This section illustrates two ways to generate the multiresolution basis, one geometric and one algebraic. The important thing to keep in mind here is not the exact heuristics used, but rather that exactly the same techniques used for other node-nested unstructured multilevel methods are used here. The new viewpoint of compressing the discrete Green's function provides additional insight, but this part of the problem is the same.

5.1. Geometric Implementation in Two Dimensions.

In this section, we describe a geometric-oriented implementation for scalar second order elliptic problems on unstructured triangular meshes. We restrict ourselves to 2D since the geometric complexity of remeshing in 3D is daunting.

The first issue to be dealt with is discretization. In this geometric implementation, the PDE is rediscretized on coarser and coarser meshes, so it is imperative to have a discretization which is stable and reasonably accurate even for large meshes. In this paper upwinding is used for convection, and harmonic averaging for diffusion; more accurate schemes such as [39] could be used instead. Obviously there could be aliasing problems with highly oscillatory coefficients with the simple rediscretization used here—without resorting to algebraic methods, the only solution would be an analytic homogenization, but we have not investigated this.

The second issue is how to choose coarse nodes. Of course, in some applications an appropriate hierarchy of nested meshes is already available, but in general an automatic procedure for generating the hierarchy is needed. The simplest approach, used for multigrid in [17], is to consider the graph of a triangulation of the current set of nodes and to select a greedily-chosen maximal independent subset as the next coarser level. These nodes can then be retriangulated for the next coarsening. Under the assumption that the edges of the triangulation represent strong couplings between unknowns, the maximality condition ensures that every fine node has at least one strongly coupled coarse node from which it can be predicted, and the independence condition ensures that there won't be too many coarse nodes.

This assumption breaks down for anisotropic PDE's or anisotropic meshes; "semi-coarsening" is needed here, where coarsening only takes place in the directions of strong coupling. Heuristically this can be implemented by rediscretizing the PDE on the coarser mesh, and then disregarding the edges corresponding to small offdiagonal nonzeros when constructing the maximal independent set. Then every fine node is guaranteed to have a strongly coupled coarse node, where the strength of coupling is measured by the size of the nonzero in the discretization. A reasonable measure of coupling strength is, for example:

$$|A_{ij}| + |A_{ji}| > \epsilon ||(|A_i| + |A_j|)||$$

for some norm of the matrix columns, and with $\epsilon = 0.1$ say. All reasonable heuristics appear to work equally well after a little tuning of ϵ on small test problems.

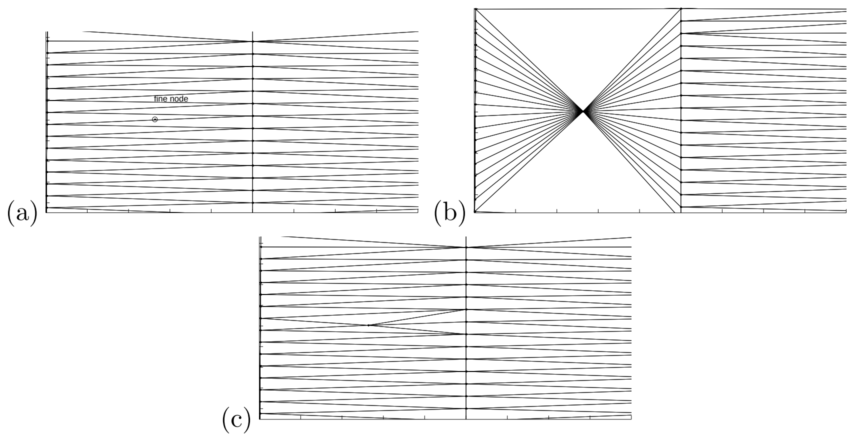
The other problem with anisotropic PDE's lies in retriangulation of the coarse nodes. At coarse levels with anisotropically distributed nodes, Delaunay triangulation (which ignores the PDE of course) may produce very poor meshes which don't reflect the anisotropy. Some form of coefficient-adapted triangulation is needed, such as breaking the region into subregions with more-or-less constant coefficients, changing coordinates in each subregion to make the PDE isotropic, Delaunay triangulating in the new coordinates, and then stitching the triangulated subregions back together.

The final issue is how to do prediction. The most robust technique is PDE-prediction, where as discussed earlier the value at a fine node is taken from the approximate solution of the homogeneous PDE or its adjoint with neighbouring coarse nodes as Dirichlet boundary data. With unstructured triangular meshes this is easily done by triangulating the fine node together with the few surrounding coarse nodes, then rediscretizing the PDE at just the fine node to give a single linear equation for the value there. To guarantee sparsity in the prediction operator, the coarse nodes are selected as the vertices of the coarse triangle containing the fine node, possibly with any of the three vertices on the other sides of the triangle's edges according to the Delaunay criterion for edge-swapping. If the fine node is on a convex boundary, the coarse triangle that comes nearest to containing it is used. Note that these neighbouring coarse nodes can be found in $O(1)$ time by doing a breadth-first search from the fine node in the fine mesh. Also we restrict the retriangulation so that at most 6 coarse nodes are used to predict the fine node, thus guaranteeing a fast (linear time) construction, sparse basis transforms, and agreement with stretched meshes (see figure 5.1 for what could go wrong with unrestricted retriangulation).

These same issues appear in any unstructured multilevel algorithm, and in particular, algorithms developed for coarsening and interpolation in multigrid et. al. can be used here, just as the algorithms given above could be used for other multilevel methods.

5.1.1. Sample Results. In [11] several example problems were given, showing that for a variety of 2D problems the new method is superior to a plain approximate inverse. For the same storage (including the basis transforms as well as the approximate inverse) and similar flop counts per iteration, the multiresolution algorithm provided several times faster convergence, even on the smallest problems. As problem sizes increased, the number of iterations for the multiresolution method grew

Fig. 5.1. *When retriangulating around a fine node in a stretched mesh (a), problems can arise if no limits are put on the Delaunay insertion algorithm since many distant coarse nodes are connected to the fine node (b), but a restricted retriangulation solves this issue as well as being far more efficient (c).*



much more slowly than for the plain approximate inverse. (Regrettably the convergence still wasn't mesh-independent, but appeared to grow like the number of levels squared, similar to hierarchical basis methods. As noted in the first part, perhaps further analysis of the update step will produce an optimal preconditioner.)

We first present results for three problems from [11] in 5.1. HEAT is a large (10 units) backwards Euler timestep of the heat equation $u_t = \nabla^2 u$ on the unit disc with Neumann boundary conditions $\nabla u \cdot \hat{n} = \text{sign}(\cos(20\theta))$ and a previous timestep of $u = 0$ for $x < 0$ and $u = 1$ for $x > 0$, on an exponentially stretched mesh. ANISO is an anisotropic discontinuous problem from [20], with $1000u_{xx} + u_{yy} = f$ on the south-west and north-east quarters of the unit square, $u_{xx} + 1000u_{yy} = f$ on the others, $f = sin(10\pi y)$, homogenous Neumann boundaries for $y > 0.25$, and the Dirichlet boundary condition $u = x$ for $y < 0.25$. REACTOR is a discontinuous indefinite problem on the unit disc of the form $\nabla \cdot K\nabla u + cu = f$, with $K = 1$, $c = 0.3$, $f = -1$ in 21 small interior discs, $K = 0.005$, $c = -0.2$, $f = -1$ for the rest of the inner disc $r < 0.9$, and $K = 10^{-6}$, $c = 0$, $f = 0$ for $r > 0.9$.

The multiresolution bases included enough levels so that the coarsest had about 100 nodes. Drop tolerances in AINV were selected to give approximately the same storage (including prediction operators) for each preconditioner: $\approx 7n$ nonzeros for a problem with n nodes. CG was used for the SPD problems and BiCGstab for the rest, with convergence flagged when the 2-norm of the residual was reduced by a factor of 10^{-6} from a starting guess of all zeros, giving up at 1000 iterations.

However, two examples of difficulties for the geometric approach arose in ANISO without the special coefficient-adapted retriangulation, and in ROTATE. The latter is a non-self-adjoint, convection-dominated problem based on a solid-body rotation of a disc (circular streamlines): see table 5.2. ROTATE couldn't be effectively solved with a complete hierarchy; the best results were obtained with only two coarse levels, precluding any improvement in the asymptotic rate of convergence.

5.2. An Algebraic Alternative. Some of the difficulties encountered in solving ANISO and ROTATE are really just artifacts of trying to rediscretize the problem on very coarse triangular meshes. A simple triangular mesh cannot easily match the

TABLE 5.1

Iteration counts for example problems of varying sizes, with the standard basis and a problem-adapted multiresolution basis for AINV. The number of unknowns starts at $n = 4939$ for HEAT, $n = 900$ for ANISO and at $n = 4195$ for REACTOR.

Problem	Method	n	$\approx 4n$	$\approx 16n$
HEAT	Standard	125	215	432
	Multiresolution	23	25	28
ANISO	Standard	37	67	111
	Multiresolution	12	14	18
REACTOR	Standard	181	355	744
	Multiresolution	89	141	132

TABLE 5.2

Iteration counts for examples of difficulties in geometric approach. Delaunay retriangulation ignoring anisotropy causes problems for ANISO; attempting to coarsely discretize curved streamlines causes problems for ROTATE. The number of unknowns starts at $n = 900$ for ANISO and at $n = 1195$ for ROTATE.

Problem	n	$\approx 4n$	$\approx 16n$	$\approx 64n$
ANISO	350	545	*	*
ROTATE	73	135	297	879

changing anisotropies of ANISO or curved streamlines of ROTATE yet both of these problems would appear to permit very coarse representations. This motivates the use of algebraic methods for basis construction, where no auxiliary meshes are used; everything is generated from the original matrix alone, hopefully avoiding geometric pitfalls in doing so. This is also an advantage in 3D, where unstructured remeshing can be difficult.

For the prediction operators, we first decide which coarse nodes will be used to predict each fine node: we select the strongly coupled coarse nodes that are either adjacent to the fine node or one of its fine neighbours. Next we determine the weights for each coarse node in the prediction.

The simplest method we try, labelled M1, is to predict the fine node value as a weighted mean of the coarse node values, with (positive) weights proportional to the magnitude of the appropriate off-diagonal entries in the matrix $\mathbf{A}$.

A potentially more accurate method, M2, is based on solving the homogeneous PDE at the fine node with boundary values specified at the surrounding coarse nodes, as in the geometric approach. In fact, the matrix gives us an equation for each node involving it and its neighbours. Unfortunately, the equation at the fine node in general involves neighbouring fine nodes as well as coarse nodes, and so we cannot stop here. Including the equations at those fine nodes would again, in general, involve more fine nodes or coarse nodes we're not using in the prediction, so the system still won't be closed. However, we can use method M1 to interpolate these unknown values from the coarse neighbours instead, and then solve the closed system for the desired fine value. This is similar to element-free AMGe[28], where an additional layer of nodes is used.

The next issue is how to generate the discretized matrix at coarser levels than the original. While these matrices are not used in the preconditioner, they are required to generate the prediction operators using the above schemes, and allow us to use the maximal independent set algorithm for coarse node selection from the previous section. The most natural choice for these coarser versions of $\mathbf{A}$ is the Petrov-Galerkin approximation to the Schur complement, as in multigrid:

$$\mathbf{A}_{coarse} = (\mathbf{P}_\beta^T\ \mathbf{I})\mathbf{A}\begin{pmatrix}\mathbf{P}_\alpha \\ \mathbf{I}\end{pmatrix}$$

Unfortunately we encountered a difficulty with this approach: for unstructured problems with a reasonable number of coarse nodes used to predict each fine node, the coarse versions of $\mathbf{A}$ quickly become dense. From a finite element perspective, the supports of the coarse basis functions have too much overlap. Perhaps with more tuning of the strong connection heuristic in the coarse node selection this could have been averted, but we looked for a more automatic approach instead.

The bulk of the extra nonzeros in the coarse versions of $\mathbf{A}$ are very small, and thus a viable approach is to simply filter out the small nonzeros at each level (perhaps with diagonal compensation) as is done in multilevel ILU[4, 3, 9, 32, 33, 32]. However, for anisotropic problems such a filter may be unreliable, destroying essential topology in the problem—it cannot distinguish between the small, negligible entries resulting from excessive overlap of coarse basis functions, and the small but non-negligible entries representing weak couplings in the original PDE (that increase in relative strength as semi-coarsening proceeds).

Our solution is to use two sets of prediction operators. One is stored for the basis transform, and the other is used temporarily just to generate the coarsened matrices. The prediction operators in the second set are much sparser, with structures chosen so that excessive fill-in is impossible (e.g. if the initial matrix has a planar graph, so do the coarsened matrices). They are poorer quality than the operators in the first set, but since they are better adapted to the problem than simple polynomial interpolation they should be superior to a standard rediscretization on the coarse nodes.

The nonzero structure of the second set of prediction operators is determined in two stages. In the first stage, each fine node is assigned the coarse node to which it is most strongly coupled, giving one nonzero per row in the prediction operator. In this way, the fine nodes are disjointly partitioned into clusters around the coarse nodes. From the FEM perspective this guarantees at most unit element overlap between coarse basis functions. From the graph theory perspective this guarantees that the graph of the coarse matrix is the result of a sequence of edge contractions (the edges coupling each fine node to its chosen coarse node) from the original matrix—and this means that graph properties such as planarity or being a triangulation are preserved. In the second stage, additional nonzeros are added to improve the quality of prediction, but only when they don't incur any extra fill in the coarse matrix. A greedy algorithm is used, considering the coarse nodes in order of how few fine dependencies they have, adding as many connections to the neighbouring fine nodes (in order of connection strength) as possible.

5.2.1. Sample Results.
We now present the results of using the algebraic approach in solving ANISO and ROTATE. Table 5.3 gives iteration counts for these problems with both prediction methods. Clearly there is more research to be done as M2 is better than M1 for ANISO, but unexpectedly worse for ROTATE.

6. Conclusions.
We have presented a new preconditioner designed for the high-performance solution of linear systems derived from elliptic PDE's. We combine the scalability of multiresolution methods with the robustness of approximate inverses to give something useful for large problems on unstructured meshes with anisotropies,

TABLE 5.3
Iteration counts for example problems of varying sizes, solved with algebraic methods. The number of unknowns starts at $n = 961$ for ANISO and at $n = 307$ for ROTATE.

Problem	Method	n	$\approx 2n$	$\approx 4n$	$\approx 8n$
ANISO	M1	12	15	17	19
	M2	10	10	13	13
ROTATE	M1	10	14	19	23
	M2	24	32	41	63

strong convection, or even indefinite reaction terms. The key idea is to create a sparse approximate inverse expressed in a multiresolution basis which compresses the discrete Green's function.

In implementing the method we have worked with a factored approximate inverse algorithm, solving the problem of ordering the unknowns in the new basis. We have also investigated both geometrical and algebraic methods for constructing the basis for unstructured problems.

Unfortunately it appears that the method doesn't scale any better than hierarchical basis methods; while this is much better than simple approximate inverses, it is sub-optimal. However, the power of the approximate inverse in addition to problem adapted interpolation means convergence is generally better than hierarchical basis methods. Robustness is particularly gained for problems where an effective complete multilevel decomposition cannot be found: the new method can truncate the hierarchy at an appropriate level and the approximate inverse can take care of the rest.

As a result, we don't expect the method to be competitive with a well tuned multigrid algorithm for well-behaved problems, but it may be of use for more difficult problems where robustness is critical.

REFERENCES

[1] R. E. Bank and T. Dupont, *Analysis of a two-level scheme for solving finite element equations*, report CNA-159, Center for Numerical Analysis, The University of Texas at Austin, Austin, TX (1980).

[2] R. E. Bank, T. Dupont, and H. Yserentant, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[3] R. Bank and R. Smith, *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., vol. 20, no. 4, pp. 1349–1364.

[4] R. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., vol. 82, no. 4, 1999, pp. 543–576.

[5] M. Benzi, C. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135-1149.

[6] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), no. 3, pp. 968-994.

[7] M. Benzi, J. K. Cullum, and M. Tůma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[8] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), no. 5, pp. 1851-1868.

[9] E. Botta and F. Wubs, *MRILU: an effective algebraic multi-level ILU-preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 4 (1999).

[10] J. Bramble, J. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), no. 191, pp. 1–22.

[11] R. Bridson, *Multi-resolution approximate inverses*, Masters thesis, University of Waterloo, 1999. Available at http://www.stanford.edu/~rbridson/download/thesis.ps.gz

[12] R. Bridson and W.-P. Tang, *Ordering, anisotropy, and factored sparse approximate inverses*, SIAM J. Sci. Comput., 21 (2000), no. 3, pp. 867-882.

[13] R. Bridson and W.-P. Tang, *Refining an approximate inverse*, J. Comp. Appl. Math, 123 (2000), Numerical Analysis 2000, vol. III: Linear algebra, pp. 293–306.

[14] R. Bridson and W.-P. Tang, *A structural diagnosis of some IC orderings*, SIAM J. Sci. Comput., 22 (2000), no. 5, pp. 1527-1532.

[15] C. Canuto, A. Tabacco, and K. Urban, *The wavelet element method. Part I: construction and analysis*, Appl. Comput. Harm. Anal., 6 (1999), pp. 1–52.

[16] C. Canuto, A. Tabacco, and K. Urban, *The wavelet element method. Part II: realization and additional features in 2D*, Appl. Comput. Harm. Anal., 8 (2000), pp. 123–165.

[17] T. Chan and B. Smith, *Domain decomposition and multigrid algorithms for elliptic problems on unstructured meshes*, Contemporary Math., 180 (1994), pp. 175–189.

[18] T. Chan, W.-P. Tang, and W. L. Wan, *Wavelet sparse approximate inverse preconditioners*, BIT, 37 (3), 1997.

[19] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (3), May 1998.

[20] S. Clift, H. Simon, and W.-P. Tang, *Spectral ordering techniques for incomplete LU preconditioners for CG methods*, manuscript.

[21] A. Cohen and R. Masson, *Wavelet methods for second-order elliptic problems, preconditioning, and adaptivity*, SIAM J. Sci. Comput., 21 (3), pp. 1006-1026.

[22] W. Dahmen, *Wavelet methods for PDEs—some recent developments*, IGPM report no. 183, RWTH Aachen, Dec. 1999.

[23] W. Dahmen and A. Kunoth, *Multilevel preconditioning*, Num. Math., 63 (1992), pp. 315–344.

[24] W. Dahmen and R. Stevenson, *Element-by-element construction of wavelets satisfying stability and moment conditions*, SIAM J. Numer. Anal., 37 (1999), pp. 319–325.

[25] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[26] J. Gilbert, *Predicting structure in sparse matrix computations*, SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 62–79.

[27] M. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), no. 3, pp. 838–853.

[28] V. E. Henson and P. Vassilevski, *Element-free AMGe: general algorithms for computing interpolation weights*, presentation at Copper Mountain Conference on Iterative Methods, May 2000.

[29] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1999), no. 1, pp. 359–392 (electronic).

[30] L. Kolotilina and A. Yeremin, *Factorized sparse approximate inverse preconditionings I. theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[31] J. Ruge and K. Stuben, *Algebraic multigrid*, in *Multigrid methods*, ed. S. McCormick, *Frontiers in Applied Mathematics*, 3 SIAM: Philadelphia, 1987.

[32] Y. Saad, *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., vol. 17, no. 4 (1996).

[33] Y. Saad and B. Suchomel, *ARMS: an alegraic recursive multilevel solver for general sparse linear systems*, technical report umsi-99-107, Computer Science Dept., University of Minnesota.

[34] Y. Saad and J. Zhang, *BILUM: block versions of multi-elimination and multi-level ILU preconditioners for general sparse linear systems*, SIAM J. Sci. Comput., 20, no. 6, pp. 2103–2121.

[35] W. Sweldens, *The lifting scheme: a construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1997), no. 2, pp. 511-546.

[36] W.-P. Tang and W. L. Wan, *A sparse approximate inverse smoother for multigrid*, to appear in SIAM J. Matrix Anal. Appl.

[37] P. S. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets, I: theory*, Numer. Lin. Alg. App., 4 (1997), no. 2, pp. 103–126.

[38] P. S. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets, II: implementation and numerical results*, SIAM J. Sci. Comput., 20 (1998), no. 2, pp. 490–514.

[39] J. Xu and L. Zikatanov, *A monotone finite element scheme for convection-diffusion equations*, Math. Comp., 68:228 (1999), pp. 1429-1446.

[40] H. Yserentant, *On the multilevel splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.