IMPLEMENTATION AND THEORETICAL ASPECTS OF THE BPX PRECONDITIONER IN THE THREE DIMENSIONAL LOCAL MESH REFINEMENT SETTING

BURAK AKSOYLU, MICHAEL HOLST, AND STEPHEN BOND

Abstract. In the setting of 3D local mesh refinement, we present the theoretical construction and the implementation aspects of the Bramble-Pasciak-Xu (BPX) preconditioner. The refinement under consideration is the 3D local red-green refinement procedure introduced by Bornemann-Erdmann-Kornhuber (BEK). We outline how to construct the theoretical optimality of the BPX preconditioner in the setting of elliptic second order PDEs. Hence, the resulting BPX preconditioner for the BEK refinement setting has provably optimal (linear) computational complexity per iteration, as well as having a uniformly bounded condition number. We provide detailed comparisons of the BPX preconditioner to hierarchical basis (HB) and wavelet modified HB preconditioners including the flop counts. Numerical experiments in 2D are presented for both the additive and multiplicative versions of the above preconditioners.

1. Introduction

In this article, we present the theoretical construction and the implementation aspects of the well-known Bramble-Pasciak-Xu (BPX) preconditioner. The BPX preconditioner is the parallelized or additive version of the multigrid preconditioner. The BPX preconditioner turned out to be an attractive choice for many applications because of this parallelization feature.

The classical treatment and the theoretical results of the BPX preconditioner were given primarily in the setting of uniform refinement. Extension of such results to local refinement has been realized with optimal computational and theoretical estimates in 2D. Optimality of the BPX preconditioner with generic local refinement was shown by Bramble-Pasciak [8], where the impact of the local smoother and the local projection operator on the estimates was carefully analyzed. The two primary results on the optimality of BPX preconditioner in the local refinement settings are due to Dahmen-Kunoth [9] and Bornemann-Yserentant [6]. Both works consider only two space dimensions, and in particular, the refinement strategies analyzed are restricted to 2D red-green refinement and 2D red refinement, respectively. (Green refinement means bisection and red refinement means quadrasection and octasection in 2D and 3D, respectively.) Furthermore, extensions of 2D estimates to a realistic 3D local mesh refinement procedure were established by Aksoylu [1], Aksoylu-Holst [3], and Aksoylu-Bond-Holst [2]. The refinement procedure of interest is a practical, implementable 3D local red-green refinement procedure introduced by Bornemann-Erdmann-Kornhuber (BEK) [5]. We will refer to this as the BEK refinement procedure. The results in [1, 2, 3] are valid for any spatial dimension $d \geq 1$, for nonsmooth PDE coefficients $p \in L_\infty(\Omega)$. The estimates provided for the BPX preconditioner throughout this article are provable.

Date: October 7, 2004; Keywords and phrases: Multilevel preconditioning, BPX, hierarchical bases, three dimensions, local mesh refinement, red-green refinement;

`baksoylu@ices.utexas.edu`, The Center for Subsurface Modelling (CSM) The University of Texas at Austin; `mholst@math.ucsd.edu`, Department of Mathematics University of California, San Diego; `sdbond@cs.uiuc.edu`, Department of Computer Science University of Illinois Urbana-Champaign.

Adaptive refinement techniques have become a crucial tool for many applications, and access to optimal or near-optimal multilevel preconditioners for locally refined mesh situations is of primary concern to computational scientists. Hence, we treat the BPX preconditioner in a more general framework for the sake of a comprehensive presentation. This is the framework of local refinement and multilevel preconditioning. We present the impact of 2D/3D local mesh refinement on the optimality (linear storage and time complexity) of multilevel preconditioners. Besides the BPX preconditioner, the preconditioners which can be expected to have somewhat favorable storage and time complexity in such local refinement scenarios are the hierarchical basis (HB) method and the wavelet modified (or stabilized) hierarchical basis (WMHB) method. All the above preconditioners belong to the class of additive Schwarz methods. We provide comparisons of these preconditioners.

The problem class we focus on here is linear second order partial differential equations (PDE) of the form:

$$-\nabla \cdot (p\ \nabla u) + q\ u = f,\ u \in H_0^1(\Omega). \tag{1}$$

Here, $f \in L_2(\Omega)$, $p, q \in L_\infty(\Omega)$, $p : \Omega \to L(\mathbb{R}^d, \mathbb{R}^d)$, $q : \Omega \to \mathbb{R}$, where p is a symmetric positive definite matrix function, and where q is a nonnegative function. Let $\mathcal{T}_0$ be a shape regular and quasiuniform initial partition of Ω into a finite number of d simplices, and generate $\mathcal{T}_1, \mathcal{T}_2, \ldots$ by refining the initial partition using red-green local refinement strategies in $d = 3$ spatial dimensions. Denote as $\mathcal{S}_j$ the simplicial linear C^0 finite element space corresponding to $\mathcal{T}_j$ equipped with zero boundary values. The set of nodal basis functions for $\mathcal{S}_j$ is denoted by $\Phi^{(j)} = \{\phi_i^{(j)}\}_{i=1}^{N_j}$ where $N_j = \dim\ \mathcal{S}_j$ is equal to the number of interior nodes in $\mathcal{T}_j$, representing the number of degrees of freedom in the discrete space. Successively refined finite element spaces will form the following nested sequence:

$$\mathcal{S}_0 \subset \mathcal{S}_1 \subset \ldots \subset \mathcal{S}_j \subset \ldots \subset H_0^1(\Omega).$$

Let the bilinear form and the functional associated with the weak formulation of (1) be denoted as

$$a(u, v) = \int_\Omega p\ \nabla u \cdot \nabla v + q\ u\ v\ dx,\ b(v) = \int_\Omega f\ v\ dx,\ u, v \in H_0^1(\Omega).$$

We consider primarily the following Galerkin formulation: Find $u \in \mathcal{S}_j$, such that

$$a(u, v) = b(v),\ \forall v \in \mathcal{S}_j. \tag{2}$$

The finite element approximation in $\mathcal{S}_j$ has the form $u^{(j)} = \sum_{i=1}^{N_j} u_i \phi_i^{(j)}$, where $u = (u_1, \ldots, u_{N_j})^T$ denotes the coefficients of $u^{(j)}$ with respect to $\Phi^{(j)}$. The resulting *discretization operator* $A_j = \{a(\phi_k^{(j)}, \phi_l^{(j)})\}_{k,l=1}^{N_j}$ must be inverted numerically to determine the coefficients u from the linear system:

$$A_j u = F_j, \tag{3}$$

where $F_j = \{b(\phi_l^{(j)})\}_{l=1}^{N_j}$. Our task is to solve (3) with optimal (linear) complexity in both storage and computation, where the finite element spaces $\mathcal{S}_j$ are built on locally refined meshes.

Outline of the paper. In §2, we outline the BPX preconditioner in the local refinement setting and make the connection between the (implementable) BPX preconditioner and the corresponding slice operator which is used in the norm equivalence to H^1-norm. §3 is dedicated to linear algebra in which we outline

the matrix representation of the BPX and HB preconditioners. In §4, we list the BEK refinement conditions. We give several theorems about the generation and size relations of the neighboring simplices, thereby establishing local (patchwise) quasiuniformity. In §5, we list the choices for local smoothers and elaborate on these in §6 and §7. In §7, we explicitly give an upper bound for the nodes introduced in the refinement region. This implies that one application of the BPX preconditioner to a function has linear (optimal) computational complexity. We present numerical experiments in §8. We conclude in §9.

2. Preliminaries

In the uniform refinement setting, the parallelized or additive version of the multigrid method, also known as the BPX preconditioner, is defined as follows:

$$Xu := \sum_{j=0}^{J} 2^{j(d-2)} \sum_{i=1}^{N_j} (u, \phi_i^{(j)}) \phi_i^{(j)}. \tag{4}$$

In the local refinement setting, in order to maintain optimal computational complexity, the smoother is restricted to a local space $\tilde{\mathcal{S}}_j$, typically

$$(I_j - I_{j-1})\, \mathcal{S}_j \subseteq \tilde{\mathcal{S}}_j \subset \mathcal{S}_j, \tag{5}$$

where $I_j : L_2(\Omega) \to \mathcal{S}_j$ denotes the finite element interpolation operator. The subspace genereated by the nodal basis functions corresponding to *fine or newly created* degrees of freedom (DOF) on level j is denoted by $(I_j - I_{j-1})\, \mathcal{S}_j$.

The basic restriction on the refinement procedure is that it remains *nested*. In other words, tetrahedra of level j which are not candidates for further refinement will never be touched in the future. Let Ω_j denote the refinement region, namely, the union of the supports of basis functions which are introduced at level j. Due to nested refinement $\Omega_j \subset \Omega_{j-1}$. Then the following hierarchy holds:

$$\Omega_J \subset \Omega_{J-1} \subset \cdots \subset \Omega_0 = \Omega.$$

The main ingredient in the local refinement setting for the analysis of the BPX preconditioner is the *local quasi-interpolant* which is given in [3]:

$$\tilde{Q}_j : L_2(\Omega) \to \mathcal{S}_j. \tag{6}$$

In fact, in our construction this operator will turn out to be a *projection*, i.e. L_2-self-adjoint and $\tilde{Q}_j^2 = \tilde{Q}_j$. For $u \in \mathcal{S}_J$, the projection $\tilde{Q}_j$ will have the *local* property that $(\tilde{Q}_j - \tilde{Q}_{j-1})u$ vanishes outside of Ω_j. A detailed discussion on the vanishing property is also presented by Oswald [14, page 94]. Let the *local smoothing operator* be the following symmetric positive definite operator (examples of these are given in [7]):

$$\tilde{R}_j : \tilde{\mathcal{S}}_j \to \tilde{\mathcal{S}}_j,$$

where $\tilde{\mathcal{S}}_j := (\tilde{Q}_j - \tilde{Q}_{j-1})\mathcal{S}_J$. Moreover, the following assumption (which naturally holds) will be enforced on $\tilde{R}_j$:

$$2^{-2j}\|v\|^2 \eqsim (\tilde{R}_j v, v), \;\; v \in \tilde{\mathcal{S}}_j. \tag{7}$$

This choice for $\tilde{\mathcal{S}}_j$ indicates that the smoother acts on a local collection of DOF which will give rise to optimal computational complexity per iteration. Nodes–equivalently, DOF–corresponding to $\tilde{\mathcal{S}}_j$ and their cardinality will be denoted by

$\tilde{\mathcal{N}}_j$ and $\tilde{N}_j$, respectively. We now define the BPX preconditioner for the local refinement setting as:

$$Xu := \sum_{j=0}^{J} 2^{j(d-2)} \sum_{i \in \tilde{\mathcal{N}}_j} (u, \phi_i^{(j)}) \phi_i^{(j)}. \tag{8}$$

The multilevel splitting of $u \in \mathcal{S}_J$ using (6) is then

$$u = \sum_{j=0}^{J} (\tilde{Q}_j - \tilde{Q}_{j-1}) u, \tag{9}$$

with $\tilde{Q}_{-1} = 0$ and $\tilde{Q}_J$ is the identity on $\mathcal{S}_J$. If $\tilde{Q}_j$ is a local projection such as a local quasi-interpolant, then the individual terms in this splitting are locally supported. The main difference in the analysis between the local and uniform refinement cases lies in the choice of the projection. Namely, in the uniform refinement case, the L_2-projection, $Q_j : L_2(\Omega) \to \mathcal{S}_j$, is used for the splitting in (9). But since $(Q_j - Q_{j-1})u$ has global support, it is not a practical choice that will lead to an optimal method. Therefore, in the local refinement case, we employ the *local projection* $\tilde{Q}_j$ which allows for optimal computational complexity. While the analysis is be based on the use of $\tilde{Q}_j$, all results also hold for projections which are globally supported such as Q_j with the exception of the optimal computational complexity result in (19). In particular, the main optimal norm equivalence result (10) holds for Q_j as well as $\tilde{Q}_j$.

Throughout this article we use the following standard notation: for $x, y \in \mathbb{R}^+$ and universal constants $c_1, c_2 \in \mathbb{R}^+$, we write:

$$x \eqsim y \quad \text{if} \quad c_1 y \leq x \leq c_2 y.$$

We first move to a general analysis framework where (8) becomes a special case. For the construction of the theoretical framework, the following operator will be referred as the BPX preconditioner for $u \in \mathcal{S}_J$.

$$Bu := \sum_{j=0}^{J} \tilde{R}_j (\tilde{Q}_j - \tilde{Q}_{j-1}) u.$$

Utilizing the splitting (9), one can write $u = \sum_{j=0}^{J} u^{(j)^f}$, where $u^{(j)^f} := (\tilde{Q}_j - \tilde{Q}_{j-1})u$. Note that B can be written as a diagonal operator using (9):

$$B = \operatorname{diag}(\tilde{R}_0 \tilde{Q}_0, \tilde{R}_1(\tilde{Q}_1 - \tilde{Q}_0), \ldots, \tilde{R}_J(\tilde{Q}_J - \tilde{Q}_{J-1})).$$

Using the projection properties, one can observe that

$$\tilde{R}_j(\tilde{Q}_j - \tilde{Q}_{j-1}) \tilde{R}_j^{-1} (\tilde{Q}_j - \tilde{Q}_{j-1}) u^{(j)^f} = u^{(j)^f}.$$

Then,

$$B^{-1} = \operatorname{diag}(\tilde{R}_0^{-1} \tilde{Q}_0, \tilde{R}_1^{-1}(\tilde{Q}_1 - \tilde{Q}_0), \ldots, \tilde{R}_J^{-1}(\tilde{Q}_J - \tilde{Q}_{J-1})).$$

The ultimate goal is the prove the following norm equivalence

$$(B^{-1}u, u) \eqsim (Au, u),$$

which will give $\kappa(BA) \eqsim 1$. To reach this goal, we use the *slice operator* induced by the splitting (9):

$$Cu := \sum_{j=0}^{J} 2^{2j}(\tilde{Q}_j - \tilde{Q}_{j-1})u.$$

Now, we can link the two operators by using (7).

$$\begin{aligned}(B^{-1}u, u) &= \sum_{j=0}^{J}(\tilde{R}_j^{-1}(\tilde{Q}_j - \tilde{Q}_{j-1})u, (\tilde{Q}_j - \tilde{Q}_{j-1})u) \\ &\eqsim \sum_{j=0}^{J} 2^{2j}\|(\tilde{Q}_j - \tilde{Q}_{j-1})u\|_{L_2}^2 = (Cu, u).\end{aligned}$$

Therefore, one focuses entirely on establishing the following norm equivalence:

$$(Cu, u) \eqsim (Au, u). \tag{10}$$

We conclude the theoretical construction by indicating that the norm equivalence (10) is equivalent to the statement of establishing the optimality of the BPX preconditioner. In the following sections, we concentrate on the implementation aspects of the BPX preconditioner.

3. Overview of the multilevel methods

Let the prolongation operator from level $j-1$ to j be denoted by

$$P_{j-1}^{j} \in \mathbb{R}^{\tilde{N}_j \times \tilde{N}_{j-1}},$$

and also denote the prolongation operator from level j to J as:

$$P_j \equiv P_j^J = P_{J-1}^J \dots P_j^{j+1} \in \mathbb{R}^{N_J \times \tilde{N}_j},$$

where P_J^J is defined to be the rectangular identity matrix $I \in \mathbb{R}^{N_J \times \tilde{N}_{J-1}}$. Then the matrix representation of (4) becomes [18]:

$$X = \sum_{j=0}^{J} 2^{j(d-2)} P_j P_j^t.$$

One can also introduce a version with an explicit smoother G_j:

$$X = \sum_{j=0}^{J} P_j G_j P_j^t.$$

Throughout this article, the smoother $G_j \in \mathbb{R}^{\tilde{N}_j \times \tilde{N}_j}$ is a symmetric Gauss-Seidel iteration. Namely, $G_j = (D_j + U_j)^{-1} D_j (D_j + L_j)^{-1}$ where $A_j = D_j + L_j + U_j$ with $\tilde{N}_0 = N_0$. Note that G_j satisfies (7).

If the smoother is restricted to the space generated by *fine or newly created* basis functions, *i.e.* $\tilde{\mathcal{S}}_j := (I_j - I_{j-1})\, \mathcal{S}_j$, then (4) corresponds to the additive HB preconditioner in [19]:

$$X_{\text{HB}} u = \sum_{j=0}^{J} 2^{j(d-2)} \sum_{i=N_{j-1}+1}^{N_j} (u, \phi_i^{(j)})\phi_i^{(j)}, \ u \in \mathcal{S}_J. \tag{11}$$

The matrix representation of (11) is formed from matrices H_j which are simply the tails of the P_j corresponding to newly introduced DOF in the fine space. In other

words, $H_j \in \mathbb{R}^{N_J \times (N_j - N_{j-1})}$ is given by only keeping the fine columns (the last $N_j - N_{j-1}$ columns of P_j). Hence, the matrix representation of (11) becomes:

$$X_{\text{HB}} = \sum_{j=0}^{J} 2^{j(d-2)} H_j H_j^t.$$

Finally, we note that the splitting in (9) gives rise to a direct decomposition $\mathcal{S}_j = \mathcal{S}_{j-1} \oplus \mathcal{S}_j^f$, hence A_j can be represented by a two-by-two block form:

$$A_j = \begin{array}{c} \left[\begin{array}{cc} A_{j-1} & A_{12}^{(j)} \\ A_{21}^{(j)} & A_{22}^{(j)} \end{array} \right] \end{array} \begin{array}{l} \} \quad \mathcal{S}_{j-1} \\ \} \quad \mathcal{S}_j^f \end{array}, \tag{12}$$

where A_{j-1}, $A_{12}^{(j)}$, $A_{21}^{(j)}$, and $A_{22}^{(j)}$ correspond to coarse-coarse, coarse-fine, fine-coarse, and fine-fine interactions respectively.

4. The BEK refinement procedure

Our interest is to show optimality of the BPX preconditioner for the 3D local red-green refinement introduced by Bornemann-Erdmann-Kornhuber [5]. This 3D red-green refinement is practical, easy to implement, and numerical experiments were presented in [5]. A similar refinement procedure was analyzed by Bey [4]; in particular, the same green closure strategy was used in both papers. While these refinement procedures are known to be asymptotically non-degenerate (and thus produce shape regular simplices at every level of refinement), shape regularity is insufficient to construct a stable Riesz basis for finite element spaces on locally adapted meshes. To construct a stable Riesz basis we will need to establish patch-wise quasiuniformity as in [9]. Riesz bases are out of the scope of this article, but are discussed in detail in [3]. As a result, d-vertex adjacency relationships that are independent of shape regularity of the elements must be established between neighboring tetrahedra as done in [9] for triangles.

We first list a number of geometric assumptions we make concerning the underlying mesh. Let $\Omega \subset \mathbb{R}^3$ be a polyhedral domain. We assume that the triangulation $\mathcal{T}_j$ of Ω at level j is a collection of tetrahedra with mutually disjoint interiors which cover $\Omega = \bigcup_{\tau \in \mathcal{T}_j} \tau$. We want to generate successive refinements $\mathcal{T}_0, \mathcal{T}_1, \ldots$ which satisfy the following conditions:

Assumption 4.1. Nestedness: *Each tetrahedron (son) $\tau \in \mathcal{T}_j$ is covered by exactly one tetrahedron (father) $\tau' \in \mathcal{T}_{j-1}$, and any corner of τ is either a corner or an edge midpoint of τ'.*

Assumption 4.2. Conformity: *The intersection of any two tetrahedra $\tau, \tau' \in \mathcal{T}_j$ is either empty, a common vertex, a common edge or a common face.*

Assumption 4.3. Nondegeneracy: *The interior angles of all tetrahedra in the refinement sequence $\mathcal{T}_0, \mathcal{T}_1, \ldots$ are bounded away from zero.*

A regular (red) refinement subdivides a tetrahedron τ into 8 equal volume subtetrahedra. We connect the edges of each face as in 2D regular refinement. We then cut off four subtetrahedra at the corners which are congruent to τ. An octahedron with three parallelograms remains in the interior. Cutting the octahedron along the two faces of these parallelograms, we obtain four more subtetrahedra which are

not necessarily congruent to τ. We choose the diagonal of the parallelogram so that the successive refinements always preserve nondegeneracy [1, 4, 13, 20].

If a tetrahedron is marked for regular refinement, the resulting triangulation violates conformity A.4.2. Nonconformity is then remedied by irregular (green) refinement. In 3D, there are altogether $2^6 = 64$ possible edge refinements, of which 62 are irregular. One must pay extra attention to irregular refinement in the implementation due to the large number of possible nonconforming configurations. Bey [4] gives a methodical way of handling irregular cases. Using symmetry arguments, the 62 irregular cases can be divided into 9 different types. To ensure that the interior angles remain bounded away from zero, we enforce the following additional conditions. (Identical assumptions were made in [9] for their 2D refinement analogue.)

Assumption 4.4. *Irregular tetrahedra are not refined further.*

Assumption 4.5. *Only tetrahedra $\tau \in \mathcal{T}_j$ with $L(\tau) = j$ are refined for the construction of $\mathcal{T}_{j+1}$, where $L(\tau) = \min\{j : \tau \in \mathcal{T}_j\}$ denotes the level of τ.*

One should note that the restrictive character of A.4.4 and A.4.5 can be eliminated by a modification on the sequence of the tetrahedralizations [4]. On the other hand, it is straightforward to enforce both assumptions in a typical local refinement algorithm by minor modifications of the supporting datastructures for tetrahedral elements (cf. [10]). In any event, the proof technique requires both assumptions hold. The last refinement condition enforced for the possible 62 irregularly refined tetrahedra is stated as the following.

Assumption 4.6. *If three or more edges are refined and do not belong to a common face, then the tetrahedron is refined regularly.*

We note that the d-vertex adjacency generation bound for simplices in $\mathbb{R}^d$ which are adjacent at d vertices is the primary result required in the support of a basis function, and depends delicately on the particular details of the local refinement procedure rather than on shape regularity of the elements. The generation bound for simplices which are adjacent at $d-1, d-2, \ldots$ vertices follows by using the shape regularity and the generation bound established for d-vertex adjacency. We provide rigorous generation bounds for all the adjacency types mentioned in the lemmas to follow when $d = 3$. The 2D version appeared in [9]; the 3D extension is as described below without any additional framework.

Lemma 4.1. *Let τ and τ' be two tetrahedra in $\mathcal{T}_j$ sharing a common face f. Then*

$$|L(\tau) - L(\tau')| \leq 1. \tag{13}$$

Proof. If $L(\tau) = L(\tau')$, then $0 \leq 1$, there is nothing to show. Without loss of generality, assume that $L(\tau) < L(\tau')$. Proof requires a detailed and systematic analysis. To show the line of reasoning, we first list the facts used in the proof:

(1) $L(\tau') \leq j$ because by assumption $\tau' \in \mathcal{T}_j$. Then, $L(\tau) < j$.
(2) By assumption $\tau \in \mathcal{T}_j$, meaning that τ was never refined from the level it was born $L(\tau)$ to level j.
(3) Let τ'' be the father of τ'. Then $L(\tau'') = L(\tau') - 1 < j$.
(4) $L(\tau) < L(\tau')$ by assumption, implying $L(\tau) \leq L(\tau'')$.
(5) By (2), τ belongs to all the triangulations from $L(\tau)$ to j, in particular $\tau \in \mathcal{T}_{L(\tau'')}$, where by (3) $L(\tau'') < j$.

f is the common face of τ and τ' on level j. If τ' is obtained by regular refinement of its father τ'', then f is still the common face of τ and τ''. By (5) both $\tau,\ \tau'' \in \mathcal{T}_{L(\tau'')}$. Then, A.4.2 implies that f is the common face of τ and τ''. Hence, τ' must have been irregular.

On the other hand, $L(\tau) \leq L(\tau') - 1 = L(\tau'')$. Next, we proceed by eliminating the possibility that $L(\tau) < L(\tau'')$. If so, we repeat the above reasoning, and τ'' becomes irregular. τ'' is already the father of the irregular τ', contradicting A.4.4 for level $L(\tau'')$. Hence $L(\tau) = L(\tau'') = L(\tau') - 1$ concludes the proof. □

By A.4.4 and A.4.5, every tetrahedron at any $\mathcal{T}_j$ is geometrically similar to some tetrahedron in $\mathcal{T}_0$ or to a tetrahedron arising from an irregular refinement of some tetrahedron in $\mathcal{T}_0$. Then, there exist absolute constants $c_1,\ c_2$ such that

$$c_1 \ \mathrm{diam}(\bar{\tau})\ 2^{-L(\tau)} \leq \mathrm{diam}(\tau) \leq c_2 \ \mathrm{diam}(\bar{\tau})\ 2^{-L(\tau)}, \tag{14}$$

where $\bar{\tau}$ is the father of τ in the initial mesh. The lemma below follows by shape regularity and (13).

Lemma 4.2. *Let τ, τ' and ζ, ζ' be the tetrahedra in $\mathcal{T}_j$ sharing a common edge (two vertices) and a common vertex, respectively. Then there exist finite numbers V and E depending on the shape regularity such that*

$$|L(\tau) - L(\tau')| \leq V, \tag{15}$$

$$|L(\zeta) - L(\zeta')| \leq E. \tag{16}$$

Consequently, simplices in the support of a basis function are comparable in size as indicated in (17). This is usually called *patchwise quasiuniformity*. Furthermore, it was shown in [1] that patchwise quasiuniformity (17) holds for 3D marked tetrahedron bisection by Joe and Liu [11] and for 2D newest vertex bisection by Sewell [15] and Mitchell [12]. Due to restrictive nature of the proof technique, we focus on refinement procedures which obey A.4.4 and A.4.5.

Lemma 4.3. *There is a constant depending on the shape regularity of $\mathcal{T}_j$ and the quasiuniformity of $\mathcal{T}_0$, such that*

$$\frac{\mathrm{diam}(\tau)}{\mathrm{diam}(\tau')} \leq c, \ \forall \tau, \tau' \in \mathcal{T}_j, \ \tau \cap \tau' \neq \emptyset. \tag{17}$$

Proof. τ and τ' are either face-adjacent (d vertices), edge-adjacent ($d-1$ vertices), or vertex-adjacent, and are handled by (13), (16), (15), respectively.

$$\begin{aligned} \frac{\mathrm{diam}(\tau)}{\mathrm{diam}(\tau')} &\leq c\ 2^{|L(\tau)-L(\tau')|}\ \frac{\mathrm{diam}(\bar{\tau})}{\mathrm{diam}(\bar{\tau'})} \text{ (by (14))} \\ &\leq c\ 2^{\max\{1,E,V\}}\ \gamma^{(0)} \text{ (by (13), (16), (15) and quasiuniformity of } \mathcal{T}_0) \end{aligned}$$

□

5. Local smoothing and choices for $\tilde{S}_j$

Only in the presence of a geometric increase in the number of DOF, the same assumption for optimality of a single classical (*i.e.* smoother acting on all DOF) multigrid or BPX iteration, does the cost per iteration remain optimal. In the case of local refinement, the classical BPX preconditioner (4) can easily be suboptimal because of the suboptimal cost per iteration (see Figure 3). On the other hand, the HB preconditioner (11) suffers from a suboptimal iteration count.

As mentioned in §2, the above deficiencies of the preconditioners (4) and (11) can be overcome by restricting the smoother to a space $\tilde{\mathcal{S}}_j$ as in (5). Recalling that we have a nested refinement, the only restriction enforced on the DOF over which the smoother acts is that they are contained in the region of refinement, *i.e.* $\tilde{\mathcal{N}}_j \subset \Omega_j$.

There are three popular choices for $\tilde{\mathcal{N}}_j$. These choices are obtained by the following DOF corresponding to:

- **(DOF-1)** The basis functions with supports that intersect Ω_j [6, 8, 9].
- **(DOF-2)** The basis functions with supports that are contained in Ω_j [14].
- **(DOF-3)** Created by red refinement and their corresponding coarse DOF.

The intereseting ones are (DOF-1) and (DOF-3) and we would like to elaborate on these. We used (DOF-1) in our numerical experiments and we will provide a provably optimal computational complexity for (DOF-3) in §7.

6. Onering neighbors–(DOF-1)

We call the set in (DOF-1) as *onering* of fine DOF, namely, the set which contains fine DOF and their immediate neighboring coarse DOF. The onering neighbors of the fine nodes can be directly determined by the sparsity pattern of the fine-fine subblock $A_{22}^{(j)}$ of the stiffness matrix in (12). The set of DOF over which the BPX method smooths is simply the union of the column locations of nonzero entries corresponding to fine DOF. Using this observation, HB smoother can easily be modified to be a BPX smoother. In short, the BPX preconditioner in the flavor of (DOF-1) is obtained by restricting i in (4) to the following set:

$$\text{ONERING}^{(j)} = \{onering(ii) : \ ii = N_{j-1}+1, \ldots, N_j\}.$$

This is the BPX preconditioner used for 2D local refinement numerical experiments.

7. Local smoothing computational complexity–(DOF-3)

(DOF-3) is equivalent to the following set:

$$\tilde{\mathcal{N}}_j = \{i = N_{j-1}+1, \ldots, N_j\} \bigcup \{i : \phi_i^{(j)} \neq \phi_i^{(j-1)}, \ i = 1, \ldots, N_{j-1}\}, \tag{18}$$

and the corresponding space over which the smoother acts:

$$\tilde{\mathcal{S}}_j = \text{span}\left[\bigcup \{\phi_i^{(j)}\}_{i=N_{j-1}+1}^{N_j} \bigcup \{\phi_i^{(j)} \neq \phi_i^{(j-1)}\}_{i=1}^{N_{j-1}}\right].$$

This set is used in the BEK construction [5] and we utilize this set for the estimates of 3D local refinement. Since green refinement simply bisects a simplex, the modified basis function is the same as the one before the bisection due to linear interpolation. So the set of DOF in (18) corresponds to DOF created by red refinement and corresponding coarse DOF (father DOF). The following crucial result from [5] establishes a bound for the number of nodes used for smoothing. This indicates that the BPX preconditioner has provably optimal (linear) computational complexity per iteration on the resulting 3D mesh produced by the BEK refinement procedure.

Lemma 7.1. *The total number of nodes used for smoothing satisfies the bound:*

$$\sum_{j=0}^{J} \tilde{N}_j \leq \frac{5}{3} N_J - \frac{2}{3} N_0. \tag{19}$$

Proof. See [5, Lemma 1]. □

The above lemma constitutes the first computational complexity optimality result in 3D for the BPX preconditioner. A similar result for 2D red-green refinement was given by Oswald [14, page 95]. In the general case of local smoothing operators which involve smoothing over newly created basis functions plus some additional set of local neighboring basis functions, one can extend the arguments from [5] and [14] using shape regularity.

8. Numerical Experiments

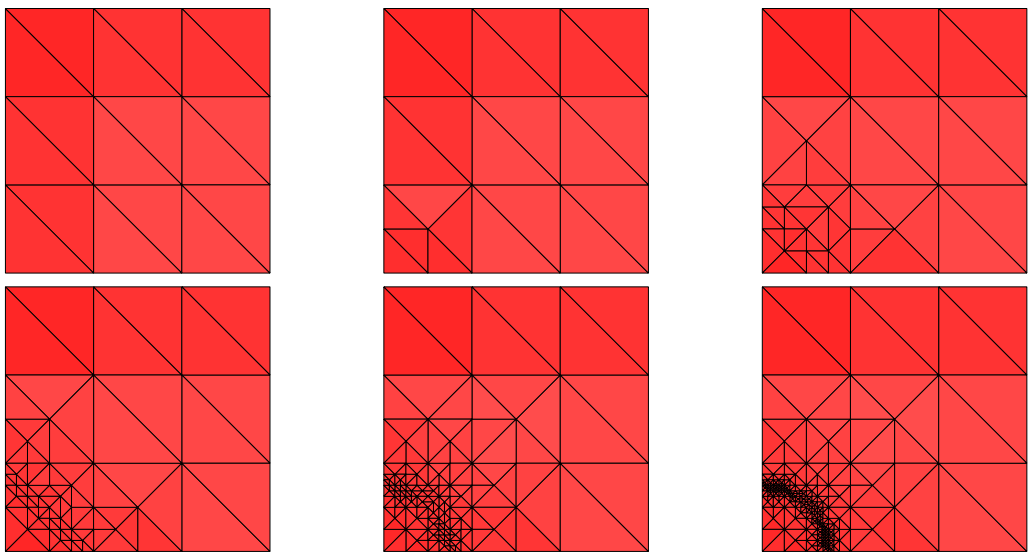

Figure 1. Adaptive mesh, experiment set I.

The test problem is as follows:

$$\begin{aligned} -\nabla \cdot (p\ \nabla u) + q\ u &= f,\ x \in \Omega \subset \mathbb{R}^2, \\ n \cdot (p\ \nabla u) &= g,\ \text{on } \Gamma_N, \\ u &= 0,\ \text{on } \Gamma_D, \end{aligned}$$

where $\Omega = [0,1] \times [0,1]$ and

$$p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \text{and } q = 1.$$

The source term f is constructed so that the true solution is $u = \sin \pi x \sin \pi y$. We present two experiment sets in which adaptivity is driven by a geometric criterion. Namely, the simplices which intersect with the quarter circle centered at the origin with radius 0.25 and 0.05, in experiment sets I and II respectively, are repeatedly marked for further refinement.

• Boundary conditions for the domain in experiment set I:

$$\begin{aligned} \Gamma_N &= \{(x,y) : x = 0, 0 < y < 1\} \cup \{(x,y) : x = 1, 0 < y < 1\} \\ \Gamma_D &= \{(x,y) : 0 \le x \le 1, y = 0\} \cup \{(x,y) : 0 \le x \le 1, y = 1\}. \end{aligned}$$

• Boundary conditions for the domain in experiment set II:

$$\begin{aligned}\Gamma_N &= \{(x,y) : 0 \le x \le 1, y = 0\} \cup \{(x,y) : 0 \le x \le 1, y = 1\} \\ &\quad \cup \{(x,y) : x = 0, 0 \le y \le 1\} \cup \{(x,y) : x = 1, 0 \le y \le 1\}.\end{aligned}$$

Stopping criterion: $\|\text{error}\|_A < 10^{-7}$.

In experiment set I, red-green refinement subdivides simplices intersecting an arc of radius 0.25 which gives rise to a rapid increase in the number of DOF. Although we have an adaptive refinement strategy, this indeed creates a geometric increase in the number of DOF, see Figure 1. Experiment set II is designed so that a small number of DOF is introduced at each level. In order to do this, green refinement subdivides simplices intersecting a smaller arc with radius 0.05.

TABLE 1. MCLite iteration counts for various methods, red-green refinement driven by geometric refinement, experiment set I.

Levels	1	2	3	4	5	6	7	8
MG	1	4	7	7	7	6	6	6
M.BPX	1	4	7	7	7	7	6	6
HBMG	1	10	19	28	32	37	45	56
WMHBMG	1	6	12	13	16	17	17	17
PCG-MG	1	3	4	5	5	5	5	5
PCG-M.BPX	1	3	5	5	5	5	5	5
PCG-HBMG	1	3	7	10	12	14	15	16
PCG-WMHBMG	1	3	7	7	9	9	9	9
PCG-A.MG	1	8	13	17	20	21	23	24
PCG-BPX	1	6	12	14	17	17	18	18
PCG-HB	1	5	14	21	26	32	38	41
PCG-WMHB	1	5	12	15	19	20	21	21
Nodes	16	19	31	55	117	219	429	835
DOF	8	10	21	43	102	202	410	814

In all the experiments, we utilize a direct coarsest level solve and the smoother is a symmetric Gauss-Seidel iteration. Set of DOF on which the smoother acts is the fundamental difference between the methods. Classical multigrid methods smooth on all DOF, whereas HB-like methods smooth only on fine DOF. WMHB style methods smooth as HB methods do, but in a different basis. BPX methods smooth on the onering of the fine DOF, which is more than HB methods but less than classical multigrid.

There are four multiplicative methods under consideration: MG, M.BPX, HBMG, and WMHBMG. The following is a guide to the tables and figures below. MG will refer to classical multigrid, in particular corresponds to the standard V-cycle implementation. HBMG corresponds exactly to the MG algorithm, but where pre- and post-smoothing are restricted to fine DOF. M.BPX refers to multiplicative version of BPX with the smoother is restricted to fine DOF and their immediate coarse neighbors which are often called as the *onering* neighbors. The onering neighbors of the fine nodes can be directly determined by the sparsity pattern of the fine-fine subblock A_{22} of the stiffness matrix in (12). The set of DOF over which the BPX

TABLE 2. MCLite iteration counts for various methods, green refinement driven by geometric refinement, experiment set II.

Levels	1	2	3	4	5	6	7
	8	9	10	11	12	13	14
MG	1	3	4	3	4	4	3
	4	4	4	4	4	4	4
M.BPX	1	4	4	4	4	4	4
	4	5	5	5	5	5	5
HBMG	1	13	14	16	22	25	26
	30	32	32	36	38	42	44
WMHBMG	1	8	11	11	12	12	12
	13	15	15	15	15	15	15
PCG-MG	1	2	3	3	3	4	3
	3	4	3	4	3	3	3
PCG-M.BPX	1	2	3	4	4	3	4
	4	4	4	4	4	4	4
PCG-HBMG	1	2	5	7	8	9	10
	10	11	12	11	12	13	13
PCG-WMHBMG	1	2	5	6	6	7	7
	8	8	8	8	8	8	8
PCG-A.MG	1	10	13	15	18	20	21
	23	25	26	28	28	28	29
PCG-BPX	1	6	10	11	13	14	15
	16	18	19	19	20	20	21
PCG-HB	1	3	9	11	14	18	20
	22	24	27	30	32	34	36
PCG-WMHB	1	3	9	12	14	16	17
	19	20	20	22	23	23	23
Nodes=DOF	289	290	296	299	309	319	331
	349	388	423	489	567	679	837

method smooths is simply the union of the column locations of nonzero entries corresponding to fine DOF. Using this observation, HBMG smoother can easily be modified to be a BPX smoother. WMHBMG is similar to HBMG, in that both are multiplicative methods, but the difference is in the basis used. In particular, the change of basis matrices are different as a result of the wavelet stabilization, where the L_2-projection to coarser finite element spaces is approximated by two Jacobi iterations.

PCG stands for the preconditioned conjugate gradient method. PCG-A.MG, PCG-BPX, PCG-HB, and PCG-WMHB involve the use of additive MG, PBX, HB, and WMHB as preconditioners for CG, respectively. HB and WMHB are additive versions of HBMG and WMHBMG respectively. Each preconditioner is implemented in a manner similar to that described in [16, 17].

Finally, note that *Nodes* denotes the total number of nodes in the simplicial mesh, including Dirichlet and Neumann nodes. The iterative methods view DOF as the

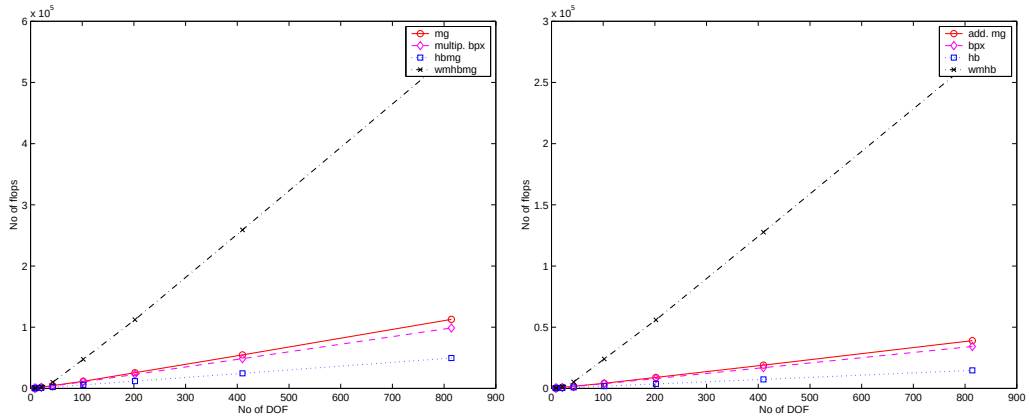

FIGURE 2. Flop counts for single iteration of multiplicative (left) and additive (right) methods, experiment set I.

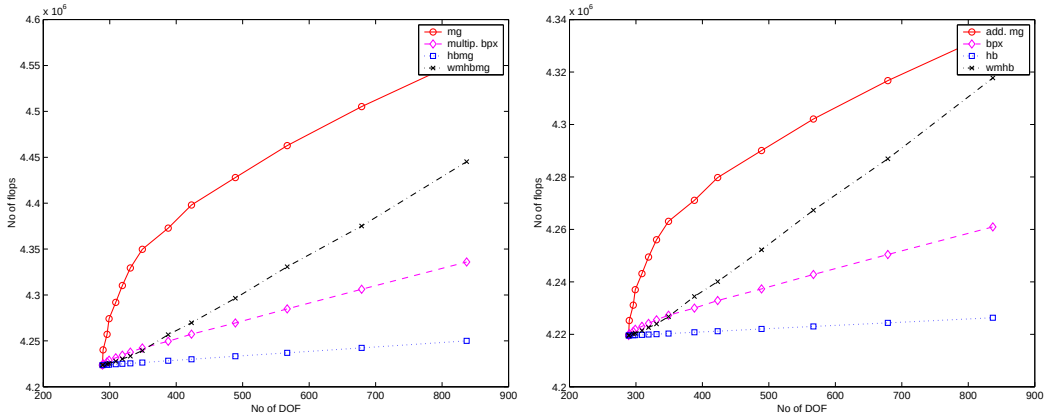

FIGURE 3. Flop counts for single iteration of multiplicative (left) and additive (right) methods, experiment set II.

union of the unknowns corresponding to interior and Neumann/Robin boundary DOF, and these are denoted as such.

The refinement procedure utilized in the experiments is fundamentally the same as the 2D red-green described in [3]. We, however, remove the restrictive conditions that the simplices for level $j+1$ have to be created from the simplices at level j and the bisected (green refined) simplices cannot be further refined. Even in this case the claimed results seem to hold. Experiments are done in the MCLite module of the FEtk package. Several key routines from this implementation, used to produce most of the numerical results in this paper, are given in the appendix.

Iteration counts are reported in Tables 1 and 2. The optimality of M.BPX, BPX, WMHBMG and WMHB is evidenced in each of the experiments. We observed a constant number of iterations independent of the number of DOF in each case. HB and HBMG methods suffer from a logarithmic increase in the number of iterations. Among all the methods tested, the M.BPX is the closest to MG in terms of low iteration counts.

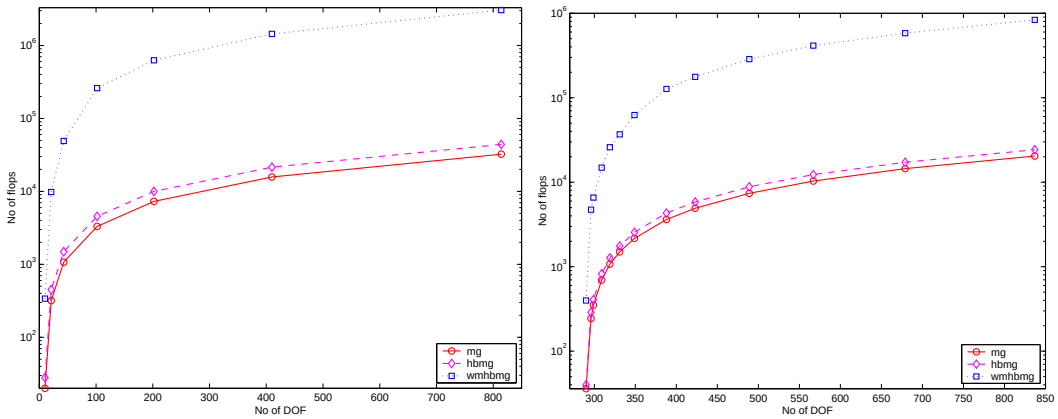

FIGURE 4. Flop counts for variational conditions for experiment set I (left) and experiment set II (right).

However, it should be clearly noted that in the experiments we present below, the cost per iteration of the various methods can differ substantially. We report flop counts of a single iteration of the above methods, see Figures 2 and 3. In experiment set I, the cost per iteration is linear for all the methods. The WMHB and WMHBMG methods are the most expensive ones. We would like to emphasize that the refinement in experiment set I cannot be a good example for adaptive refinement given the geometric increase in the number of DOF. MG exploits this geometric increase and enjoys a linear computational complexity. Experiment set II is more realistic in the sense that the refinement is highly adaptive and introduces a small number of DOF at each level. One can now observe a suboptimal (logarithmic) computational complexity for MG-like methods in such realistic scenarios. In accordance with the theoretical justification, under highly adaptive refinement MG methods will asymptotically be suboptimal. Moreover, storage complexity severely prevents MG-like methods from being a viable tool for large and highly adaptive settings.

Coarser representations of the finest level system (3) are algebraically formed by enforcing variational conditions. Some methods require further stabilizations in the from of matrix-matrix products. These form the so-called *preprocessing* step in multilevel methods. The computational cost of variational conditions is the same regardless of having a multiplicative or an additive version of the same method; see Figure 4. This computational cost is orders of magnitude cheaper than the cost of a single iteration. However, this is the step where the storage complexity can dominate the overall complexity. Due to memory bandwidth problems on conventional machines, one should be very careful with the choice of datastructures. Since only the $A_{11} = A_{coarse}$ subblock of A is formed for the next coarser level, the cost of variational conditions for MG, M.BPX, A.MG, and BPX is the cheapest among all the methods. On the other hand, HBMG and HB require stabilizations of A_{12} and A_{21} using the hierarchical basis. The WMHBMG and WMHB methods are more demanding by requiring stabilizations of A_{12}, A_{21}, and A_{22} in (12) using the wavelet modified hierarchical basis. Wavelet structure creates denser change of basis matrix than that of the hierarchical basis. Therefore, preprocessing in the WMHB and WMHBMG methods is the most expensive among all the methods.

9. Conclusion

In this paper, we examined the Bramble-Pasciak-Xu (BPX) preconditioner in the setting of 3D local mesh refinement and numerically compared a number of additive and multiplicative multilevel iterative preconditioners in the setting of two-dimensional local mesh refinements. We outlined the theoretical framework to prove the BPX preconditioner's optimality–uniformly bounded condition number–on locally refined 3D meshes based on an easily implementable red-green mesh refinement, *i.e.* BEK, procedure. All of the results here require no smoothness assumptions on the PDE coefficients.

While standard multilevel methods are effective for uniform refinement-based discretizations of elliptic equations, they tend to be less effective for algebraic systems which arise from discretizations on locally refined meshes, losing their optimal behavior in both storage and computational complexity. To address the practical computational complexity of implementable versions of BPX and WMHB, we indicated how the number of degrees of freedom used for the smoothing step can be shown to be bounded by a constant times the number of degrees of freedom introduced at that level of refinement. This indicates that practical implementable versions of the BPX and WMHB preconditioners for the 3D local refinement setting considered here have provably optimal (linear) computational complexity per iteration, as well as having a uniformly bounded condition number. A detailed analysis of both the storage and per-iteration computational issues arising with BPX and WMHB implementations can be found in [2].

The presented numerical experiments illustrated the effectiveness of the BPX and stabilized HB methods in adaptive regimes. As expected, multigrid methods are most effective in terms of iteration counts (remaining a small constant as the DOF increase), but the suboptimal complexity per iteration in the local refinement setting makes the BPX methods the most attractive. Furthermore, storage complexity prohibits MG methods from being a viable tool for large and highly adaptive settings. In addition, both the additive and multiplicative WMHB-based methods and preconditioners demonstrated similar constant iteration requirements with increasing DOF, yet the cost per iteration remains optimal (linear) even in the local refinement setting. Consequently in highly adaptive regimes, the BPX methods prove to be the most effective, and the WMHB methods become the second most effective. The superiority of the BPX and WMHB methods would be more striking in large three-dimensional problems.

10. Appendix: Implementation Highlights

```
function [u]=additive(f,lev,hb);
%%% Additive methods: A.MG, BPX, HB, WMHB

%%% prolongation, stiffness, change of basis, one-ring
global P_12 P_23 P_34 P_45 A_1 A_2 A_3 A_4 A_5;
global S_2 S_3 S_4 S_5 ONER_2 ONER_3 ONER_4 ONER_5;
global A_hb smthKey exactC bpx;

%%% get the stiffness matrix on this level
A = eval(['A_' num2str(lev) ]);

if (lev == 1)
```

```
   if (exactC) u = A \ f; else u = f; end
else
   ONER = eval(['ONER_' num2str(lev)]);

   %%% recover the dimensions
   P = eval(['P_' num2str(lev-1) num2str(lev)]);
   [r c] = size(P);

   %%% shorthand for the top and tail of vectors/matrices
   top_  = 1:c;
   tail_ = (c+1):r;

   if (hb)
      u = zeros(r,1);

      %%% Get the change of basis matrix for this level
      S = eval(['S_' num2str(lev)]);
      %%% Transform f into the HB basis
      f = f + S'*f;
      %%% fine smoothing by symmetric Gauss-Seidel
      u = smooth_point(A_hb,u,f,smthKey,2,lev);
   else %%% additive MG
      u = zeros(c,1);
      d = zeros(r,1);
      %%% smoothing by symmetric Gauss-Seidel
      if (bpx)
         d = smooth_point(A,d,f,smthKey,3,lev);
      else %%% mg
         d = smooth_point(A,d,f,smthKey,1,lev);
      end;

      %%% coarse grid restriction: f = P'*f;
      f(top_,1) = f(top_,1) + P(tail_,:)'*f(tail_,1);
   end;

   %%% Recursion
   u(top_,1) = additive(f(top_,1),lev-1,hb);

   if (hb)
      %%% Transform u into the HB basis
      u = u + S*u;
   else
      %%% interpolate result: u = P*u;
      u(tail_,1) = P(tail_,:)*u(top_,1);
      if (bpx)
         u(ONER) = u(ONER) + d(ONER);
      else %%% mg
         u = u + d;
      end;
   end;
end;

function [u]=multiplicative(b,lev,hb);
```

```
%%% Multiplicative methods: MG, M.BPX, HBMG, WMHBMG

%%% prolongation, stiffness, change of basis, one-ring
global P_12 P_23 P_34 P_45 A_1 A_2 A_3 A_4 A_5;
global level S_2 S_3 S_4 S_5 ONER_2 ONER_3 ONER_4 ONER_5;
global A_hb smthKey exactC bpx;

%%% get the stiffness matrix on this level
A = eval(['A_' num2str(lev)]);

if (lev == 1)
   if (exactC) u = A \ b; else u = b; end;
else
   ONER = eval(['ONER_' num2str(lev)]);

   %%% recover the dimensions
   P = eval(['P_' num2str(lev-1) num2str(lev)]);
   [r c] = size(P);

   %%% shorthand for the top and tail of vectors/matrices
   top_  = 1:c;
   tail_ = (c+1):r;

   if (hb)
      u = zeros(r,1);
      f = b;

      %%% Get the change of basis matrix for this level
      S = eval(['S_' num2str(lev)]);
      %%% Transform f into the HB basis
      f = f + S'*f;
      %%% pre-smoothing by symmetric Gauss-Seidel
      u = smooth_point(A_hb,u,f,smthKey,2,lev);
      %%% correct f using smoother result
      f(top_,1) = f(top_,1) - A_hb(top_,tail_)*u(tail_,1);
   else %%% mg/bpx
      u = zeros(c,1);
      d = zeros(r,1);
      if (bpx)
         d = smooth_point(A,d,b,smthKey,3,lev);
      else %%% mg
         d = smooth_point(A,d,b,smthKey,1,lev);
      end;

      %%% coarse grid defect restriction: f = P'*(b - A*d);
      if (bpx)
         f = b - A(:,ONER)*d(ONER);
      else %%% mg
         f = b - A*d;
      end

      f(top_,1) = f(top_,1) + P(tail_,:)'*f(tail_,1);
   end;
```

```
   %%% Recursion
   u(top_,1) = multiplicative(f(top_,1),lev-1,hb);

   if (hb)
      %%% correct f using the coarse solve result
      f(tail_,1) = f(tail_,1) - A_hb(tail_,top_)*u(top_,1);

      %%% post-smoothing by symmetric Gauss-Seidel
      u = smooth_point(A_hb,u,f,smthKey,2,lev);

      %%% transform u back into the nodal basis
      u = u + S*u;
   else %%% mg/bpx
      %%% interpolate result: u = P*u;
      u(tail_,1) =  P(tail_,:)*u(top_,1);

      if (bpx)
         u(ONER) = u(ONER) + d(ONER);
         u = smooth_point(A,u,b,smthKey,3,lev);
      else %%% mg
         u = u + d;
         u = smooth_point(A,u,b,smthKey,1,lev);
      end
end;
```

References

[1] B. Aksoylu, *Adaptive Multilevel Numerical Methods with Applications in Diffusive Biomolecular Reactions*, PhD thesis, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2001.

[2] B. Aksoylu, S. Bond, and M. Holst, *An odyssey into local refinement and multilevel preconditioning III: Implementation and numerical experiments*, SIAM J. Sci. Comput., 25 (2003), pp. 478–498.

[3] B. Aksoylu and M. Holst, *An odyssey into local refinement and multilevel preconditioning: I. Optimality of the BPX preconditioner II. Stabilizing hierarchical basis methods*, SIAM J. Numer. Anal., (2002). in review.

[4] J. Bey, *Tetrahedral grid refinement*, Computing, 55 (1995), pp. 271–288.

[5] F. Bornemann, B. Erdmann, and R. Kornhuber, *Adaptive multilevel methods in three space dimensions*, Intl. J. for Numer. Meth. in Eng., 36 (1993), pp. 3187–3203.

[6] F. Bornemann and H. Yserentant, *A basic norm equivalence for the theory of multilevel methods*, Numer. Math., 64 (1993), pp. 455–476.

[7] J. H. Bramble and J. E. Pasciak, *The analysis of smoothers for multigrid algorithms*, Math. Comp., 58 (1992), pp. 467–488.

[8] ———, *New estimates for multilevel algorithms including the V-cycle*, Math. Comp., 60 (1993), pp. 447–471.

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

[10] M. Holst, *Adaptive numerical treatment of elliptic systems on manifolds*, Advances in Computational Mathematics, 15 (2002), pp. 139–191.

[11] B. Joe and A. Liu, *Quality local refinement of tetrahedral meshes based on bisection*, SIAM J. Sci. Comput., 16 (1995), pp. 1269–1291.

[12] W. F. Mitchell, *Unified Multilevel Adaptive Finite Element Methods for Elliptic Problems*, PhD thesis, Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1988.

[13] M. E. G. Ong, *Uniform refinement of a tetrehedron*, SIAM J. Sci. Comput., 15 (1994), pp. 1134–1144.

[14] P. Oswald, *Multilevel Finite Element Approximation Theory and Applications*, Teubner Skripten zur Numerik, B. G. Teubner, Stuttgart, 1994.

[15] E. G. Sewell, *Automatic generation of triangulations for piecewise polynomial approximation*, PhD thesis, Department of Mathematics, Purdue University, West Lafayette, IN, 1972.

[16] P. S. Vassilevski and J. Wang, *Stabilizing the hierarchical basis by approximate wavelets, I: Theory*, Numer. Linear Alg. Appl., 4 Number 2 (1997), pp. 103–126.

[17] ———, *Stabilizing the hierarchical basis by approximate wavelets, II: Implementation and numerical experiments*, SIAM J. Sci. Comput., 20 Number 2 (1998), pp. 490–514.

[18] J. Xu and J. Qin, *Some remarks on a multigrid preconditioner*, SIAM J. Sci. Comput., 15 (1994), pp. 172–184.

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

[20] S. Zhang, *Multilevel iterative techniques*, PhD thesis, Pennsylvania State University, 1988.