A New Parallel Domain Decomposition Preconditioner I: Application with an Adaptive Parallel Finite Element Solver

Randolph E. Bank[*]and Peter K. Jimack[†]

Abstract: Adaptive algorithms are of great importance in computational mechanics codes since they can allow both reliability, through the satisfaction of error tolerances, and efficiency, by ensuring that the total number of degrees of freedom present is as small as possible. Unfortunately, the successful incorporation of adaptivity into most software is a complex programming task, and this is especially true for parallel codes. This paper introduces a new parallel domain decomposition preconditioner which is ideally suited for use in an adaptive framework. Unlike conventional domain decomposition approaches, this technique requires each process to work on the entire domain but with a coarse mesh which has been locally refined only in the subdomain for which that process is responsible. In order to justify the proposed preconditioner it is presented as a natural development of existing domain decomposition and subspace iteration algorithms, and its implementation as part of a parallel mesh adaptivity algorithm, due to Bank and Holst [2], is also outlined. The paper concludes with the presentation and discussion of a number of provisional numerical results.

1 Introduction

Throughout this paper we will consider the parallel finite element solution of the following linear second order model problem.

Problem 1.1 *Find* $u \in H^1_E(\Omega)$ *such that*

$$\mathcal{A}(u,v) = \mathcal{F}(v), \quad \forall v \in H^1_0(\Omega) \ , \tag{1}$$

where $\Omega \in \Re^2$ *is the problem domain,*

$$H^1_E(\Omega) = \{u \in H^1(\Omega) : u|_{\partial\Omega_E} = u_E(\underline{x})\} \tag{2}$$

and

$$H^1_0(\Omega) = \{u \in H^1(\Omega) : u|_{\partial\Omega_E} = 0\} \ . \tag{3}$$

---

[*] Dept. of Mathematics, University of California at San Diego, CA 92093, USA. (The work of this author was supported by the National Science Foundation under contract DMS-9706090.)

[†] School of Computer Studies, University of Leeds, Leeds LS2 9JT, UK. (The work of this author was supported by the Leverhulme Trust.)

Here $\partial\Omega_E$ is the (non-empty) part of the boundary, $\partial\Omega$, upon which essential boundary conditions are imposed and $\mathcal{A}(\cdot,\cdot)$ and $\mathcal{F}(\cdot)$ are the bilinear and linear forms

$$\mathcal{A}(u,v) = \int_\Omega (A(\underline{x})\underline{\nabla} u)\cdot\underline{\nabla} v \; d\underline{x} \quad \text{and} \quad \mathcal{F}(v) = \int_\Omega fv \; d\underline{x} + \int_{\partial\Omega_N} gv \; ds \;, \tag{4}$$

where $A(\underline{x})$ is symmetric and strictly positive-definite, and $\partial\Omega_N = \partial\Omega - \partial\Omega_E$ is the part of the boundary subject to Neumann boundary conditions: $\frac{\partial u}{\partial n} = g(\underline{x})$.

In order to approximate this solution from a finite dimensional space of trial functions, $S^h(\Omega)$ say, it is necessary to solve the following discrete problem.

Problem 1.2 *Find* $u^h \in S^h(\Omega) \cap H^1_E(\Omega)$ *such that*

$$\mathcal{A}(u^h, v^h) = \mathcal{F}(v^h), \quad \forall v^h \in S^h(\Omega) \cap H^1_0(\Omega) \;. \tag{5}$$

This problem may in turn be expressed as the matrix equation

$$K\underline{u} = \underline{b} \;, \tag{6}$$

where K is the stiffness matrix, $\underline{b}$ is the load vector and $\underline{u}$ is a vector of nodal displacements which is to be determined.

The matrix K is strictly positive-definite and, for the usual choices of finite element trial space and basis (e.g. [17, 22]), sparse. Hence an iterative solution method for (6), such as the conjugate gradient method, [12], is most appropriate. Unfortunately, it is well known that when $S^h(\Omega)$ is a space of piecewise polynomial functions defined on a mesh of elements covering Ω with edge size h, $\mathcal{T}^h$ say, the condition number of K grows like $O(h^{-2})$ as $h \to 0$ (see [17] for example). For this reason it is necessary to apply a preconditioned version of the conjugate gradient algorithm for realistic mesh sizes h (again see [12]). (Note that, in this context, reference to meshes of size h means that there exist constants, c_1 and c_2, such that the length of each edge of every element in $\mathcal{T}^h$ lies in the interval $[c_1 h, c_2 h]$.)

There are many possible ways in which the system (6) can be preconditioned. Some of these are purely algebraic, such as incomplete Cholesky factorization [4, 5, 11, 18], whilst others make use of the underlying finite element derivation of the system, such as element-by-element preconditioning [16, 23] or domain decomposition preconditioners (e.g. [7, 10, 13, 14, 15, 20, 24, 25]), which are the subject of this paper.

The essential idea behind any preconditioning strategy is to find a positive-definite matrix, M say, that has two properties.

1. The matrix $M^{-1}K$ should have a small condition number.
2. The system $M\underline{s} = \underline{r}$ should be computationally cheap to solve.

(In fact the above properties refer to what is known as left preconditioning, where the system (6) is expressed as

$$(M^{-1}K)\underline{v} = M^{-1}\underline{b} \;. \tag{7}$$

This is the form of preconditioning that is considered in this paper, however all of the results may also be extended to symmetric or right preconditioning.) When seeking to solve the system (6) using preconditioning on a parallel computer there is an additional requirement.

3. The system $M\underline{s} = \underline{r}$ should be easy to solve in parallel.

The first of these properties is required because the rate of convergence of the preconditioned conjugate gradient (PCG) algorithm is dependent upon the condition number of the preconditioned matrix $M^{-1}K$, and the other properties should hold because the major computational step at each iteration of the PCG algorithm is the solution of a system of the form $M\underline{s} = \underline{r}$ ([12]). As we will see, the domain decomposition algorithm proposed here can satisfy all three of these requirements.

2 Background

In this section we give a brief description of the theoretical background behind domain decomposition algorithms by considering them in the context of subspace correction methods ([24]). In particular we follow the abstract framework laid out in [24] in order to describe the common class of parallel domain decomposition preconditioners known as additive Schwartz methods (also see [8, 9, 13, 20, 25] for example). For the simplicity of this description we will assume that $u_E(\underline{x}) \equiv 0$ (see (2)) and define $\mathcal{V} = S^h(\Omega) \cap H_0^1(\Omega)$ to be both the trial space and the test space in (5). We will also assume that the triangulation $\mathcal{T}^h$ may be obtained by the uniform refinement of some coarser triangulation, $\mathcal{T}^H$ say, of the domain Ω, and we will define $\mathcal{V}_0$ to be the corresponding finite element space $S^H(\Omega) \cap H_0^1(\Omega)$ defined on $\mathcal{T}^H$ (see Fig. 1 for an example of such meshes).

Having introduced a coarse mesh $\mathcal{T}^H$ it is now possible to decompose Ω into (possibly overlapping) subdomains, $\Omega_1, ..., \Omega_p$ say, which are each the union of triangles in $\mathcal{T}^H$ (see Fig. 2 for an example with $p = 2$). We now define the spaces $H_0^1(\Omega_i)$, for $i = 1, ..., p$, by

$$H_0^1(\Omega_i) = \{u \in H^1(\Omega) : u(\underline{x}) = 0 \;\; \forall \underline{x} \in (\Omega - \bar{\Omega}_i) \cup \partial\Omega_E\} \; , \tag{8}$$

and the corresponding finite dimensional spaces $\mathcal{V}_i = S^h(\Omega) \cap H_0^1(\Omega_i)$. Note that these local spaces, $\mathcal{V}_i$, form a decomposition of the finite element space $\mathcal{V}$:

$$\mathcal{V} = \sum_{i=1}^{p} \mathcal{V}_i \; . \tag{9}$$

Thus, for each $v \in \mathcal{V}$, there exist a (not necessarily unique) combination of $v_i \in \mathcal{V}_i$ $(i = 1, ..., p)$ such that $v = \sum_{i=1}^{p} v_i$.

Given any space decomposition of the form (9), the additive Schwartz algorithm defines a preconditioner, M^{-1}, for K in (6) in the following manner. Let $\bar{Q}_i$ and $\bar{P}_i$ be orthogonal projections from $\mathcal{V}$ to $\mathcal{V}_i$ (for $i = 1, ..., p$) given by

$$\int_\Omega (\bar{Q}_i u) v_i \, d\underline{x} \;=\; \int_\Omega u v_i \, d\underline{x} \quad \forall u \in \mathcal{V}, v_i \in \mathcal{V}_i \tag{10}$$

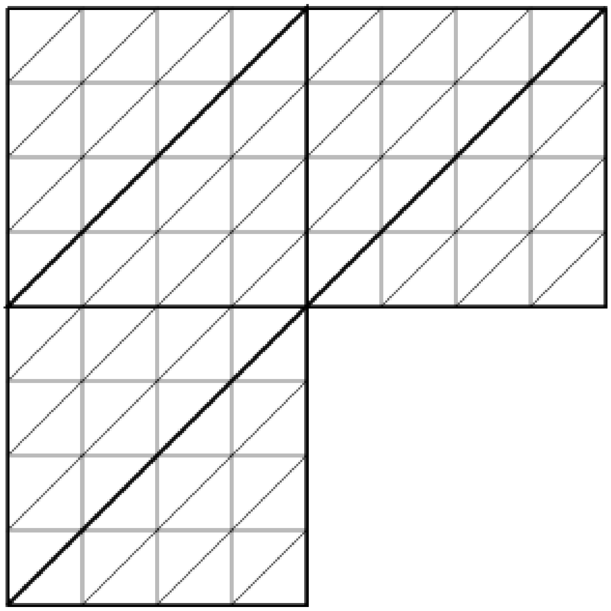

Figure 1: An example of how the mesh $\mathcal{T}^h$ might be obtained by the uniform refinement of a coarser mesh $\mathcal{T}^H$.

and

$$\int_\Omega (A(\underline{x})\underline{\nabla}(\bar{P}_i u)) \cdot \underline{\nabla} v_i \, d\underline{x} \;=\; \int_\Omega (A(\underline{x})\underline{\nabla} u) \cdot \underline{\nabla} v_i \, d\underline{x} \quad \forall u \in \mathcal{V}, v_i \in \mathcal{V}_i \tag{11}$$

respectively. Also define $\mathcal{A}_i$ to be the restriction of $\mathcal{A}$ to $\mathcal{V}_i \times \mathcal{V}_i$ given by:

$$\mathcal{A}_i(u_i, v_i) = \mathcal{A}(u_i, v_i), \quad \forall u_i, v_i \in \mathcal{V}_i \; . \tag{12}$$

Note that in matrix notation $\bar{Q}_i$ and $\bar{P}_i$ may be expressed as rectangular matrices, Q_i and $P_i \in \Re^{n_i \times n}$, and that Q_i is such that

$$K_i = Q_i K Q_i^T \; , \tag{13}$$

where $K_i \in \Re^{n_i \times n_i}$ is the stiffness matrix corresponding to $\mathcal{A}_i$ (derived in the same way that $K \in \Re^{n \times n}$ is derived from $\mathcal{A}$ above). The additive Schwartz (parallel subspace correction) preconditioner for (6) is then given by

$$M^{-1} = \sum_{i=1}^{p} Q_i^T K_i^{-1} Q_i \; . \tag{14}$$

Note that each of the subdomain solves ($K_i^{-1}\underline{r}_i$), required when solving the system $M\underline{s} = \underline{r}$ at each PCG iteration, may be performed concurrently. For simplicity we will assume here that all such subdomain solves are exact.

We now quote, without proofs, a number of results that apply to preconditioners of the form (14). Full details of these results may be found in [24], including proofs which are generalized to the case of inexact subdomain solves at each PCG iteration, along with many more results concerning subspace correction algorithms. The first lemma demonstrates that the proposed preconditioner is positive-definite as required.

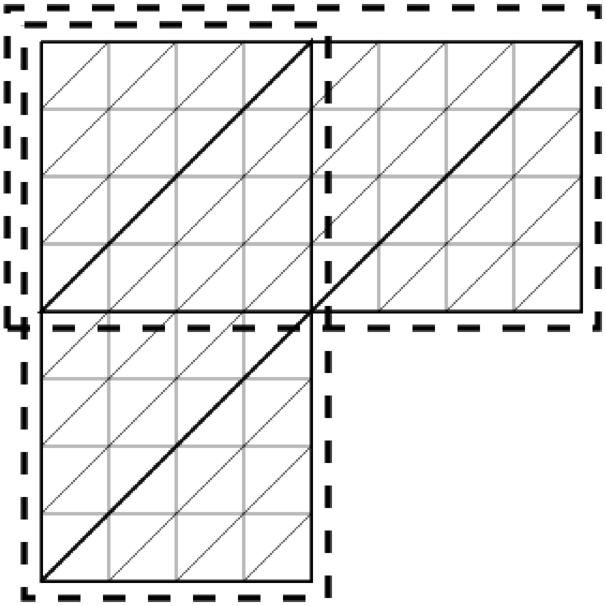

Figure 2: An example of how the meshes from Fig. 1 might be decomposed into two overlapping subdomains.

Lemma 2.1 *The matrix M defined from (14) is symmetric and positive-definite.*

We now give some conditions under which it is possible to derive bounds on the maximum and minimum eigenvalues of $M^{-1}K$.

Theorem 2.2 *Assume that there exist constants C_1 and C_2 that satisfy the following conditions.*

1. *For all $v \in \mathcal{V}$ there are $v_i \in \mathcal{V}_i$ such that $v = \sum_{i=1}^{p} v_i$ and*

$$\sum_{i=1}^{p} \mathcal{A}_i(v_i, v_i) \leq C_1 \mathcal{A}(v, v) \ . \tag{15}$$

2. *For all $S \subset \{1, ..., p\} \times \{1, ..., p\}$ and $u_i, v_i \in \mathcal{V}_i$ $(i = 1, ..., p)$*

$$\sum_{(i,j)\in S} \mathcal{A}(\bar{P}_i u_i, \bar{P}_j v_j) \leq C_2 \left(\sum_{i=1}^{p} \mathcal{A}(\bar{P}_i u_i, u_i) \right)^{\frac{1}{2}} \left(\sum_{i=1}^{p} \mathcal{A}(\bar{P}_j v_j, v_j) \right)^{\frac{1}{2}} \ . \tag{16}$$

Then, for M^{-1} given by (14),

$$\kappa(M^{-1}K) \leq C_1 C_2 \ , \tag{17}$$

where $\kappa(M^{-1}K)$ is the spectral condition number of $M^{-1}K$.

It transpires that the second of the conditions required for the above theorem is quite easy to fulfill.

Lemma 2.3 *Condition 2 in the statement of Theorem 2.2 is satisfied with* $C_2 = p$*: the number of subspaces. Moreover, for the specific choices of* $\mathcal{V}_i$ *defined in this section, the condition is also satisfied with* $C_2 = n_c$*: the minimum number of colours required to colour the subdomains,* Ω_i*, in such a way that no neighbours are the same colour.*

Finding a constant C_1 to fulfill the first of the conditions required by Theorem 2.2 is not so straightforward however. This is the key to proving strong results about the quality of the proposed preconditioner since it is easy to derive from the above the following corollary.

Corollary 2.4 *Provided (15) is satisfied for some constant* C_1 *then*

$$\kappa(M^{-1}K) \leq pC_1 \ . \tag{18}$$

From (18) it is clear that in order to obtain an effective preconditioner we must choose a set of subspaces which permit C_1 in (15) to be as small as possible. In particular, it is desirable that C_1 should only grow slowly (if at all) as the finite element mesh, $\mathcal{T}^h$, and the coarse substructure, $\mathcal{T}^H$, are refined. Unfortunately, the decomposition described in equations (8) to (9) does not permit such a choice of C_1 since it is entirely local in nature and so significant reductions in the low frequency error components can require many PCG iterations. This is easily rectified however by the introduction of an extra, coarse grid, term in the preconditioner (14):

$$M^{-1} = \sum_{i=0}^{p} Q_i^T K_i^{-1} Q_i \ . \tag{19}$$

Here Q_0 is the $\Re^{N \times n}$ matrix corresponding to the orthogonal projection, $\bar{Q}_0$, from $\mathcal{V}$ to the coarse grid space $\mathcal{V}_0$, given by,

$$\int_\Omega (\bar{Q}_0 u) v_0 \, d\underline{x} \ = \ \int_\Omega u v_0 \, d\underline{x} \quad \forall u \in \mathcal{V}, v_0 \in \mathcal{V}_0 \ , \tag{20}$$

and K_0 is the $\Re^{N \times N}$ stiffness matrix derived from $\mathcal{A}_0$, the restriction of $\mathcal{A}$ to $\mathcal{V}_0 \times \mathcal{V}_0$ given by

$$\mathcal{A}_0(u_0, v_0) = \mathcal{A}(u_0, v_0), \quad \forall u_0, v_0 \in \mathcal{V}_0 \ . \tag{21}$$

Hence

$$K_0 = Q_0 K Q_0^T \ . \tag{22}$$

The following result is also proved in [24], and applies to the new preconditioner defined in (19).

Lemma 2.5 *Provided the overlap between the subdomains* Ω_i *is of size* $O(H)$*, where* H *represents the mesh size of* $\mathcal{T}^H$*, then there exists* C_1 *which is independent of* h*,* H *and* p*, such that for any* $v \in \mathcal{V}$ *there are* $v_i \in \mathcal{V}_i$ *such that* $v = \sum_{i=0}^{p} v_i$ *and*

$$\sum_{i=0}^{p} \mathcal{A}_i(v_i, v_i) \leq C_1 \mathcal{A}(v, v) \ . \tag{23}$$

Hence $\kappa(M^{-1}K) \leq C_1(p+1)$.

(Note that there are now $p+1$ subspaces $\mathcal{V}_i$ to be counted in Lemma 2.3.) In fact, a slightly more general result can be proved for an overlap in the subdomains of $O(\delta)$; in which case it may be shown that $C_1 = O(1 + (H/\delta)^2)$.

The preconditioner given by (19) is the typical form of the additive Schwartz preconditioner with a coarse grid. For $i = 1, ..., p$ each of the subspace problems may be solved concurrently using data that is local to Ω_i only. Whilst the problem for $i = 0$ is also independent of the others, it is a global problem and so causes more difficulties from the point of view of a parallel implementation. Different strategies are possible for the solution of this problem, such as to solve it on a single processor (e.g. [15]) or to solve it in parallel using the domain decomposition approach recursively (e.g. [6, 25]). The latter approach is an example of a multilevel method. Such methods are known to perform very well on globally refined sequences of meshes but are considerably more complex to implement in parallel when combined with an adaptive algorithm which uses local mesh refinement. In the next section we introduce an alternative way of handling the coarse grid component that is an essential part of any domain decomposition preconditioner. This is designed specifically for use with parallel solvers based upon local mesh refinement, whilst still maintaining the good spectral properties of more conventional preconditioners (such as in Lemma 2.5 for example).

3 A New Preconditioner

In this section we derive a new parallel preconditioner for finite element systems of the form (6) in three distinct stages. Our goal is to develop an effective preconditioner which fits naturally alongside adaptive software for the solution of problems of the form (1).

In [2] Bank and Holst suggest a new approach to the parallel implementation of adaptive finite element solvers in two and three dimensions. They consider the use of local mesh refinement in the following manner. First the problem, (1) for example, is solved once on a single processor on a coarse mesh $\mathcal{T}^H$. This coarse mesh is then partitioned between the available processors based upon local *a posteriori* error estimates for the initial crude solution. The objective of the partition is to ensure that each subregion has about the same total error, so that the following steps will be reasonably well load-balanced. Next, each processor solves the entire problem adaptively on the entire coarse mesh, but is only permitted to refine this mesh within the subregion that was assigned to it (and possibly in the immediate neighbourhood of this subregion too; see below). The target number of elements and grid points for the mesh on each processor is approximately equal, even though the number of coarse mesh elements may differ significantly. A *final fine mesh* may now be defined to be the union of the local refinements of the subregions "owned" by each processor — although this mesh is never actually assembled on a single processor. If the local error estimates used as the basis of the refinement are reliable and the solution of (1) is smooth then this final mesh will match at the subregion boundaries, however a practical implementation should always test that this is the case and fix the mesh locally when it fails to match.

Version 1

In the first version of the domain decomposition algorithm described here we allow each processor to refine the coarse mesh not only in those elements which it owns, but also in the coarse elements that are their immediate neighbours. There may be some further mesh refinement required outside of this extended region too in order to keep the mesh conforming: in this situation we constrain any new nodes on the midpoints of edges to have solution values which are the average of those at the two end points. If $\mathcal{W}_i$ represents the finite element space defined on the mesh generated this way on processor i it then follows that

$$\mathcal{W}_i = \mathcal{V}_i \cup \mathcal{V}_0 \ . \tag{24}$$

Here, the $\mathcal{V}_i$ are the same spaces as defined in the previous section and, for $i = 1, ..., p$, correspond to subdomains Ω_i with an overlap of $O(H)$. Hence the following result must hold by analogy with Lemma 2.5.

Lemma 3.1 *Let $\mathcal{V}$ be the global finite element space defined on the final fine mesh defined above. Then there exists C_1, independent of h, H and p, such that for any $v \in \mathcal{V}$ there are $w_i \in \mathcal{W}_i$ such that $v = \sum_{i=1}^{p} w_i$ and*

$$\sum_{i=1}^{p} \mathcal{A}_i(w_i, w_i) \leq C_1 \mathcal{A}(v, v) \ . \tag{25}$$

From this lemma and the results of the previous section it follows that the domain decomposition preconditioner of the form (14) (but with the stiffness matrices, K_i, now formed on the global coarse mesh with its local refinement in and around the subregion owned by processor i) is such that $\kappa(M^{-1}K) \leq (p+1)C_1$.

Note that there is no longer a need for a separate coarse mesh solve to be completed as part of this preconditioner since this global aspect is now dealt with on each processor, $i = 1, ..., p$, in a natural manner. Furthermore, the independence of C_1 from h and H is also due to the large overlap in the regions of refinement on each processor. Whilst a smaller overlap, of $O(h)$ rather than $O(H)$ say, would cause this desirable theoretical property to be lost, it might still result in a more efficient procedure overall since the work at each PCG iteration would be significantly reduced. Thus, provided the total number of iterations does not increase by too much, the total computational cost could decrease.

Version 2

In this version we still apply an additive Schwartz preconditioner of the form (14) but now for slightly smaller spaces $\mathcal{W}_i$ (and their smaller stiffness matrices, $K_i \in \Re^{m_i \times m_i}$ say) than in Version 1 above. On processor i we allow the coarse mesh to be refined only in the coarse elements owned by that processor plus an additional "layer" of elements immediately around this subregion. Note that this layer is of width $O(h)$ rather than $O(H)$ as in the previous version. As before, some further refinement of the mesh on each processor ($i = 1, ..., p$) is also required beyond this additional layer in order to keep the mesh conforming: unlike in Version 1 however we choose not to place any constraints on the nodal values at the extra midpoint nodes that are introduced. This has the theoretical disadvantage that the spaces $\mathcal{W}_i$ are no longer strictly subspaces of $\mathcal{V}$ but appears to make little difference in practice, other than being more straightforward to implement.

Version 3

In order to ensure that the preconditioners M of the form (14) defined in each of the algorithms above are symmetric and positive-definite, the prolongation matrix on each processor is always chosen to be the transpose of the projection matrix Q_i. In Version 2 this transpose is in $\Re^{n \times m_i}$ and so a more general form that the preconditioner could take would be

$$M^{-1} = \sum_{i=1}^{p} R_i^T K_i^{-1} Q_i \ , \qquad (26)$$

where $R_i^T \in \Re^{n \times m_i}$ is some other prolongation operator. In general this choice of M^{-1} will not be symmetric or positive-definite and therefore nor will $M^{-1}K$. This means that the PCG algorithm cannot be applied and so some other iterative technique should be used. This could take the form of a fixed point iteration (e.g. [3]) or could be some other Krylov subspace algorithm such as the preconditioned GMRES algorithm (PGM), [1, 12, 21], for example.

In the implementation considered here we choose R_i^T to be the unique matrix that maps $\underline{\zeta} \in \Re^{m_i}$ to $\underline{z} \in \Re^n$ in the following manner (where the index j is used to enumerate the n nodes in the final fine mesh).

$$\begin{array}{rcll} z_j &=& \zeta_k & \text{when node } j \text{ is in the interior of} \\ &&& \text{the subregion owned by processor } i \\ &&& \text{(and is numbered } k \text{ on processor } i\text{),} \\ \\ z_j &=& \zeta_k/\nu_k & \text{when node } j \text{ is on the boundary of} \\ &&& \text{the subregion owned by processor } i\text{,} \\ &&& \text{(and is numbered } k \text{ on processor } i\text{),} \\ \\ z_j &=& 0 & \text{otherwise.} \end{array}$$

Here ν_k represents the total number of processors for which node k of mesh i lies on their subregion boundary. Hence M^{-1} is a form of weighted average of $K_i^{-1}Q_i$ on each of the meshes. The logic behind this choice of M^{-1} is that it weights the information from the fine mesh on each processor most heavily when assembling that processor's contribution to the overall preconditioner. It is anticipated that the improvement that results from this weighting will more than compensate for the additional work and storage required when using PGM rather than PCG.

4 Computational Examples

Details of the efficient parallel implementation of the algorithms introduced in the previous section are provided in [3]. In this section we seek to demonstrate the potential of these algorithms by considering their application to two linear test problems of the form (1). In order to asses their performance as preconditioners, convergence results have been collected on a sequence of uniformly refined meshes using exact sub-problem solves at each iteration. Generalizations to problems with local adaptivity, inexact sub-problem solves, non-self-adjoint and nonlinear partial differential equations are all discussed in the next section.

The test problems that we consider here are as follows.

Problem 4.1

$$\begin{aligned} -\underline{\nabla} \cdot (\underline{\nabla} u) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1) \ , \\ u &= g \quad \forall \underline{x} \in \partial\Omega \ . \end{aligned}$$

Problem 4.2

$$\begin{aligned} -\underline{\nabla} \cdot \left(\begin{pmatrix} 10^2 & 0 \\ 0 & 1 \end{pmatrix} \underline{\nabla} u \right) &= f \quad \forall \underline{x} \in \Omega \equiv (0,1) \times (0,1) \ , \\ u &= g \quad \forall \underline{x} \in \partial\Omega \ . \end{aligned}$$

In each case f has been chosen to permit the exact solution $u = g$ for some quadratic choice of g.

Tables 1 and 2 below show the number of iterations required by each of the preconditioners in the previous section to reduce the 2-norm of the initial residual by a factor of 10^6 for these two problems respectively. In each case a 256 element coarse mesh has been used and the final fine grids have between 4096 and 1048576 elements. The number of processors goes from 2 to 16.

Inspection of these tables shows that there is very little to choose between the two symmetric versions of the preconditioner (Version 1 and Version 2) in terms of the number of iterations required. For the more demanding of the two test problems, Problem 4.2, the use of the generous overlap of $O(H)$ can be seen to result in slightly fewer iterations than with an overlap of $O(h)$ as the mesh is refined. It should be noted however that the matrix problems that must be solved by each processor in Version 1 are considerably larger than those that must be solved in Version 2, which result from the smaller overlap. Hence this second version generally produces the required solution with substantially fewer floating point operations that Version 1.

It is also apparent from these tables that any growth in the condition number of $M^{-1}K$ as h is refined is extremely slow (if at all) for all three versions of the preconditioner. In addition, the non-symmetric preconditioner, Version 3, always requires significantly fewer iterations that either of the two symmetric versions. Even allowing for the fact that slightly more work is required to complete an average iteration of the PGM algorithm compared to the PCG algorithm, Version 3 still proves to be substantially more efficient that the other two. It is the use of this preconditioner that we propose in this paper, and in the following section its possibilities are discussed in more detail.

Fine mesh	Version 1	Version 2	Version 3	Procs.
4096	6	6	3	2
16384	6	6	3	
65536	6	6	3	
262144	6	6	3	
1048576	6	6	3	
4096	8	9	3	4
16384	8	8	3	
65536	8	8	3	
262144	8	7	3	
1048576	8	7	3	
4096	13	12	4	8
16384	13	12	4	
65536	12	12	4	
262144	11	11	4	
1048576	11	11	4	
4096	14	14	4	16
16384	14	13	4	
65536	14	13	4	
262144	14	12	4	
1048576	13	12	4	

Table 1: The number of iterations required to reduce the 2-norm of the residual by a factor of 10^6 using the three versions of the algorithm described in Section 3 when solving Problem 4.1 using piecewise linear finite elements.

5 Discussion

In the previous section it is demonstrated that the non-symmetric preconditioner that we propose appears to perform very well when applied with the PGM algorithm for solving finite element equations of the form (6). At each iteration of this algorithm it is necessary to solve a system of the form $M\underline{s} = \underline{r}$, which, by (26), is equivalent to

$$\underline{s} = \sum_{i=1}^{p} R_i^T K_i^{-1} Q_i \underline{r} \ . \qquad (27)$$

If we define $\underline{r}_i \in \Re^{m_i}$ to be $Q_i\underline{r}$, then each processor must solve its own system of the form

$$K_i \underline{s}_i = \underline{r}_i \qquad (28)$$

at each iteration. So far we have assumed that these systems are always solved exactly, however this is an unnecessary expense. In practice, a sufficiently accurate approximate solution to (28) is always adequate at each iteration: the theoretical justification for this being provided in [24] for example.

Given that the matrices K_i are sparse (they are the conventional finite element stiffness matrices for the meshes generated on each processor $i = 1, ..., p$),

Fine mesh	Version 1	Version 2	Version 3	Procs.
4096	7	8	5	2
16384	7	8	5	
65536	7	8	5	
262144	7	8	6	
1048576	7	8	6	
4096	11	12	5	4
16384	12	13	6	
65536	12	13	6	
262144	11	12	7	
1048576	10	12	7	
4096	14	15	7	8
16384	14	16	8	
65536	13	16	8	
262144	13	17	8	
1048576	13	17	9	
4096	18	19	7	16
16384	18	19	8	
65536	19	20	9	
262144	18	21	10	
1048576	18	21	10	

Table 2: The number of iterations required to reduce the 2-norm of the residual by a factor of 10^6 using the three versions of the algorithm described in Section 3 when solving Problem 4.2 using piecewise linear finite elements.

it makes good sense to use an iterative method to solve these local problems approximately at each iteration. In our implementation we use a preconditioned conjugate gradient algorithm with an algebraic preconditioner based upon an incomplete factorization of K_i (see [4, 5] for example). Numerical experiment suggests that it is sufficient to find approximate solutions of the systems (28) for which the residual is decreased by a factor of about 10^2 to 10^3 in order to get the best balance between the cheapness of each PGM iteration and keeping the total number of these outer iterations from becoming too large.

Recall from Section 3 that the main motivation for proposing this preconditioner is that it should be straightforward to apply in conjunction with parallel adaptivity. In order to illustrate that this is indeed the case, we now consider a third test problem. This takes the same form as Problem 4.1 but now the source term f is chosen such that the analytic solution is given by:

$$u = (1 - (2x_1 - 1)^{100})(1 - (x_2 - 1)^{100}) \quad \forall \underline{x} \in \Omega = (0,1) \times (0,1)\ . \tag{29}$$

Note that this solution is unity in the interior of Ω but tends to zero very rapidly in a thin layer (of width ≈ 0.02) near to the boundary; allowing the Dirichlet condition $u = 0$ to be satisfied throughout $\partial\Omega$.

Once more a coarse triangulation, $\mathcal{T}^H$, containing just 256 elements was used when solving this problem on 2, 4, 8 and 16 processors. The final fine mesh for this problems contains up to seven levels of refinement (hence $h \approx 0.001$ at the highest level) and has almost 90000 vertices and 176000 elements. The vast majority of these elements are situated in the transition layer near to the boundary as illustrated by the first mesh shown in Fig. 3. The coarse mesh $\mathcal{T}^H$ is partitioned in such a way that the number of refined elements in each subregion is approximately equal. In the cases where $p = 8$ and $p = 16$ this means that there are differing numbers of coarse elements in each subregion (between 8 and 40 when $p = 16$ for example) — the partitions into 2 or 4 may be computed more simply however, using the symmetry of the problem and the coarse mesh (again see Fig. 3).

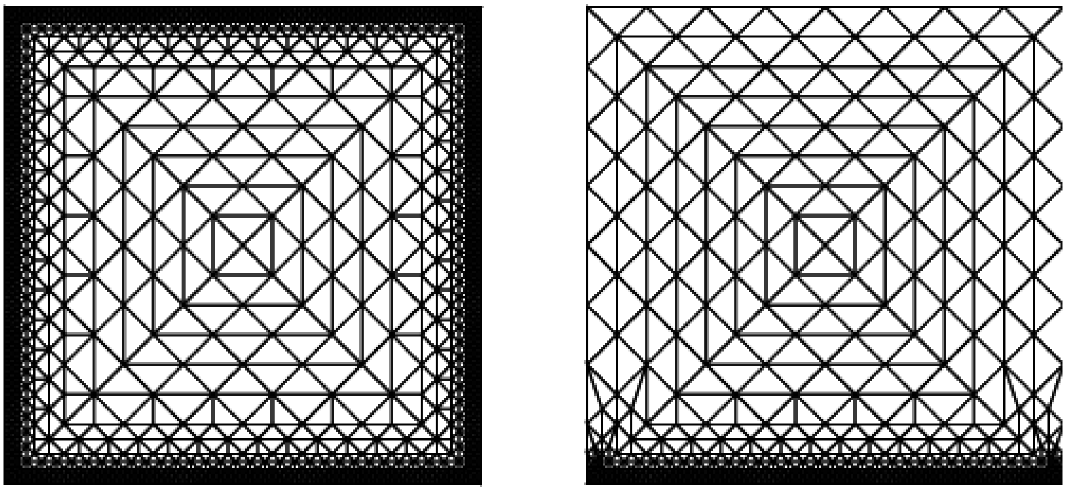

Figure 3: The final fine mesh (left) and a typical mesh on one processor when $p = 4$ (right) for the problem considered in Section 5. Here the coarse mesh contains 256 elements and at most three levels of refinement are permitted.

Table 3 shows the number of outer iterations of the PGM algorithm and the total cpu times required on an SGI Origin 2000 when solving this test problem with 2, 4, 8 and 16 processors. These cpu times, in seconds, are the sums of the times taken by each processor however they *exclude* any time taken for inter-processor communication and other parallel overheads. This therefore allows a comparison to be made between the different solution algorithms, independently of the efficiency of their parallel implementation. This comparison is complemented by the inclusion of the conventional single processor solution time using the PCG algorithm to achieve the required reduction factor of 10^6 in the residual. This single processor time is the best that was achieved over a number of different trials allowing different amounts of fill-in in the (incomplete factorization) preconditioner. The purpose of the table is to demonstrate that even on a single processor the algorithm that we propose is competitive with the best equivalent sequential algorithm. This is clearly of great importance since there is little value in having a parallel algorithm that scales well unless it is also efficient as a single processor algorithm in its own right.

Processors	1	2	4	8	16
Iterations	1	4	4	5	5
Total cpu	7.31s	8.07s	7.32s	7.88s	7.64s

Table 3: A comparison of the relative performance of the different solution algorithms corresponding to differing numbers of processors.

It may also be observed from Table 3 that the average cost per iteration reduces as p increases. This is to be expected since the approximate solutions of the $m_i \times m_i$ systems (28) on each processor have a complexity that is worse than $O(m_i)$. Hence, it is faster to solve $2p$ of these systems of size $m_i/2$ than it is to solve p systems of size m_i. To compensate for this however it is clear, from the results in Tables 1 and 2 as well as Table 3, that as p increases the number of outer iterations required to reach a fixed relative convergence tolerance will also go up. It appears from our provisional results that, provided there are sufficient elements in the coarse mesh, $\mathcal{T}^H$, these two effects generally combine in such a way that the overall sequential solution time does not grow significantly with p.

Full details of the efficient parallel implementation of the above algorithm may be found in [3]. In that paper the algorithm is applied to a more general class of problem than (1); taking the form

$$-\underline{\nabla} \cdot (A(\underline{x})\underline{\nabla} u) + \underline{b} \cdot \underline{\nabla} u + cu = f \qquad \text{on } \Omega \subset \Re^2 \tag{30}$$

(where A is symmetric and strictly positive-definite and $c \geq 0$). Given that the preconditioner proposed here makes no use of the symmetry in (1), its application to non-self-adjoint problems such as (30) poses no significant new problems. Furthermore, the technique may also be successfully applied to the parallel solution of certain nonlinear partial differential equations: here the nonlinear algebraic equations that result from the finite element discretizations on each processor may be solved with a quasi-Newton method, such as [19], where the linear system that arises at each Newton iteration can be solved using the PGM algorithm with a preconditioner of the form (26). Again see [3] for further details.

6 Conclusions

In this paper we have introduced a new parallel domain decomposition preconditioner as a natural extension of conventional additive Schwartz techniques. The novel features of this new algorithm are: (i) that each processor works over the entire problem domain, but with a mesh that is locally refined in and around its own subregion, and (ii) that the preconditioner sacrifices symmetry in order to improve its overall quality. The preconditioner is designed to work well as part of an adaptive finite element solution procedure and its performance has been demonstrated on problems with both uniformly and non-uniformly refined meshes. These provisional results suggest that the approach proposed may have significant potential.

Acknowledgements

The work of REB was supported by the National Science Foundation under contract DMS-9706090 and the work of PKJ, whilst visiting UCSD, was supported by a Research Grant from the Leverhulme Trust.

References

[1] S.F. Ashby, T.A. Manteuffel and P.E. Taylor, *"A Taxonomy for Conjugate Gradient Methods"*, SIAM J. on Numerical Analysis, 27, 1542–1568, 1990.

[2] R.E. Bank and M. Holst, *"A New Paradigm for Parallel Adaptive Meshing Algorithms"*, in preparation, 1999.

[3] R.E. Bank and P.K. Jimack, *"A New Parallel Domain Decomposition Method for the Adaptive Finite Element Solution of Elliptic Partial Differential Equations"*, in preparation, 1999.

[4] R.E. Bank and R.K. Smith, *"The Incomplete Factorization Multigraph Algorithm"*, to appear in SIAM J. on Scientific Computing, 1999.

[5] R.E. Bank and C. Wagner, *"Multilevel ILU Decomposition"*, to appear in Numerische Mathematik, 1999.

[6] J. Bramble, J. Pasciak and J. Xu, *"Parallel Multilevel Preconditioners"*, Mathematics of Computation, 55, 1–21, 1990.

[7] T.F. Chan and J.P. Shao, *"Parallel Complexity of Domain Decomposition Methods and Optimal Coarse Grid Size"*, Parallel Computing, 21, 1033–1049, 1995.

[8] M. Dryja and O.B. Widlund, *"Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems"*, in Third International Symposium on Domain Decomposition Methods (T.F. Chan *et al*, eds.), SIAM publications, Philadelphia, 1990.

[9] M. Dryja and O.B. Widlund, *"Some Domain Decomposition Algorithms for Elliptic Problems"*, in Iterative Methods for Large Linear Systems, Academic Press, 1990.

[10] C. Farhat, J. Mandel and F.X. Roux, *"Optimal Convergence Properties of the FETI Domain Decomposition Method"*, Computer Methods for Applied Mechanics and Engineering, 115, 365–385, 1994.

[11] A. George and J.W. Liu, *"Computer Solution of Large Sparse Positive Definite Systems"*, Prentice Hall, 1981.

[12] G.H. Golub and C.F. Van Loan, *"Matrix Computations"*, John Hopkins Press, 3rd edition, 1996.

[13] M. Griebel and P. Oswald, *"On Additive Schwartz Preconditioners for Sparse Grid Discretizations"*, Numerische Mathematik, 66, 449–463, 1994.

[14] W.D. Gropp and D.E. Keyes, *"Parallel Performance of Domain-Decomposed Preconditioned Krylov Methods for PDEs with Locally Uniform Refinement"*, SIAM J. on Scientific Computing, 13, 128–145, 1992.

[15] D.C. Hodgson and P.K. Jimack, *"A Domain Decomposition Preconditioner for a Parallel Finite Element Solver on Distributed Unstructured Grids"*, Parallel Computing, 23, 1157–1181, 1997.

[16] T.J.R. Hughes, I. Levit and J. Winget, *"An Element-by-Element Solution Algorithm for Problems of Structural and Solid Mechanics"*, Computer Methods for Applied Mechanics and Engineering, 36, 241–254, 1983.

[17] C. Johnson *"Numerical Solution of Partial Differential Equations by the Finite Element Method"*, Cambridge University Press, 1987.

[18] T.A. Mantueffel, *"Shifted Incomplete Cholesky Factorization"*, in Sparse Matrix Proceedings (I.S. Duff and G.W. Stewart eds.), SIAM Publications, Philadelphia, 1978.

[19] M. Pernice and H.F. Walker, *"NITSOL: A Newton Iterative Solver for Nonlinear Systems"*, SIAM J. on Scientific Computing, 19, 302–318, 1998.

[20] W. Rachowicz, *"An Overlapping Domain Decomposition Preconditioner for an Anisotropic h-Adaptive Finite Element Method"*, Computer Methods for Applied Mechanics and Engineering, 127, 269–292, 1995.

[21] Y. Saad and M. Schultz, *"GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems"*, SIAM J. on Scientific Computing, 7, 856–869, 1986.

[22] G. Strang, G.J. Fix: *"An Analysis of the Finite Element Method"*, Prentice-Hall, Englewood-Cliffs, 1973.

[23] A.J. Wathen, *"An Analysis of some Element-by-Element Techniques"*, Computer Methods for Applied Mechanics and Engineering, 74, 271–287, 1989.

[24] J. Xu, *"Iterative Methods by Space Decomposition and Subspace Correction"*, SIAM Review, 34, 581–613, 1992.

[25] X. Zhang, *"Multilevel Schwartz Methods"*, Numerische Mathematik, 63, 521–539, 1992.