INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2002; **40**:1571–1579 (DOI: 10.1002/fld.413)

Parallel implementation of an optimal two level additive Schwarz preconditioner for the 3-D finite element solution of elliptic partial differential equations

S. A. Nadeem[†] and P. K. Jimack[*,‡]

Computational PDE Unit, School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

SUMMARY

This paper presents a description of the extension and parallel implementation of a new two level additive Schwarz (AS) preconditioner for the solution of 3-D elliptic partial differential equations (PDEs). This preconditioner, introduced in Bank *et al.* (*SIAM J. Sci. Comput.* 2002; **23**:1818), is based upon the use of a novel form of overlap between the subdomains which makes use of a hierarchy of meshes: with just a single layer of overlapping elements at each level of the hierarchy. The generalization considered here is based upon the restricted AS approach reported in (*SIAM J. Sci. Comput.* 1999; **21**:792) and the parallel implementation is an extension of work in two dimensions (*Concurrency Comput. Practice Experience* 2001; **13**:327). Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: additive Schwarz; preconditioner; finite elements

1. INTRODUCTION

To begin a brief description of the optimal two level AS preconditioner introduced in Reference [1] is provided. The construction of this preconditioner is based upon the existence of a nested sequence of meshes on each subdomain, each with a single layer of overlap onto the neighbouring subdomains.

Let $\mathscr{T}_0$ be a coarse triangulation of the problem domain, Ω say, consisting of N_0 tetrahedral elements, $\tau_j^{(0)}$, such that $\tau_j^{(0)} = \bar{\tau}_j^{(0)}$,

$$\bar{\Omega} = \bigcup_{j=1}^{N_0} \tau_j^{(0)} \quad \text{and} \quad \mathscr{T}_0 = \{\tau_j^{(0)}\}_{j=1}^{N_0} \tag{1}$$

Also let diameter $(\tau_j^{(0)}) = O(H)$, and divide Ω into p *non-overlapping* subdomains Ω_i. These subdomains should be such that:

$$\bar{\Omega} = \bigcup_{i=1}^{p} \bar{\Omega}_i \tag{2}$$

* Correspondence to: P. K. Jimack, Computational PDE Unit, School of Computing, University of Leeds, Leeds LS2 9JT, U.K.
† Funded by the Government of Pakistan through a Quaid-e-Azam scholarship.
‡ E-mail: pkj@comp.leeds.ac.uk

Received 1 March 2002
Revised 12 June 2002

$$\Omega_i \cap \Omega_j = \phi \quad (i \neq j) \tag{3}$$

$$\bar{\Omega}_i = \bigcup_{j \in I_i} \tau_j^{(0)} \quad \text{where } I_i \subset \{1, \ldots, N_0\} \quad (I_i \neq \phi) \tag{4}$$

The coarse triangulation $\mathscr{T}_0$ may be refined several times, to produce a family, $\mathscr{T}_0, \ldots, \mathscr{T}_J$, where each triangulation in this family, $\mathscr{T}_k$, consists of N_k tetrahedral elements, $\tau_j^{(k)}$, such that

$$\bar{\Omega} = \bigcup_{j=1}^{N_k} \tau_j^{(k)} \quad \text{and} \quad \mathscr{T}_k = \{\tau_j^{(k)}\}_{j=1}^{N_k} \tag{5}$$

The successive mesh refinements that define this sequence of triangulations need not be global and may be non-conforming, however it is necessary that they satisfy a number of conditions:

1. $\tau \in \mathscr{T}_{k+1}$ implies that either
   (a) $\tau \in \mathscr{T}_k$, or
   (b) τ has been generated as a refinement of an element of $\mathscr{T}_k$ into eight regular children (by bisecting each edge of this parent of τ),
2. the level of any tetrahedra which share a common point can differ by at most one,
3. only tetrahedra at level k may be refined in the transition from $\mathscr{T}_k$ to $\mathscr{T}_{k+1}$.

(Here the level of a tetrahedron is defined to be the least value of k for which that tetrahedron is an element of $\mathscr{T}_k$.) In addition to the above it is also necessary that:

4. in the final mesh, $\mathscr{T}_J$, all sets of tetrahedra which share an edge which lies on the interface of any subdomain with any other subdomain have the same level as each other (i.e. the mesh is conforming along subdomain interfaces).

Having defined a decomposition of Ω into subdomains and a nested sequence of triangulations of Ω the restrictions of each of these triangulations onto each subdomain may be defined by

$$\Omega_{i,k} = \{\tau_j^{(k)} : \tau_j^{(k)} \subset \bar{\Omega}_i\} \tag{6}$$

In order to introduce a certain amount of overlap between neighbouring subdomains

$$\tilde{\Omega}_{i,k} = \{\tau_j^{(k)} : \tau_j^{(k)} \quad \text{has a common point with } \bar{\Omega}_i\} \tag{7}$$

is also defined. Following this the finite element (FE) spaces associated with these local triangulations may be introduced. Let G be some triangulation and denote by $\mathscr{S}(G)$ the space of continuous piecewise linear functions on G. Then the following definitions are made:

$$\mathscr{W} = \mathscr{S}(\mathscr{T}_J) \tag{8}$$

$$\mathscr{W}_0 = \mathscr{S}(\mathscr{T}_0) \tag{9}$$

$$\mathscr{W}_{i,k} = \mathscr{S}(\Omega_{i,k}) \tag{10}$$

$$\tilde{\mathscr{W}}_{i,k} = \mathscr{S}(\tilde{\Omega}_{i,k}) \tag{11}$$

$$\tilde{\mathscr{W}}_i = \tilde{\mathscr{W}}_{i,0} + \cdots + \tilde{\mathscr{W}}_{i,J} \tag{12}$$

It is evident that

$$\mathscr{W} = \mathscr{W}_0 + \tilde{\mathscr{W}}_1 + \cdots + \tilde{\mathscr{W}}_p \tag{13}$$

where, as in (2), p is the number of subdomains. This is the decomposition that forms the basis of the two level additive Schwarz preconditioner (see Reference [2] for example), M say, in Reference [1]. Hence, for a global FE stiffness matrix A, this preconditioner takes the form

$$M^{-1} = \sum_{i=0}^{p} R_i^{\mathrm{T}} A_i^{-1} R_i \tag{14}$$

where (using the usual local bases for $\mathscr{W}$, $\mathscr{W}_0$ and $\tilde{\mathscr{W}}_i$) R_i is the rectangular matrix representing the L^2 projection from $\mathscr{W}$ to $\mathscr{W}_0$ (when $i=0$) or $\tilde{\mathscr{W}}_i$ (when $i>0$), and A_i is the FE stiffness matrix corresponding to the subspace $\mathscr{W}_0$ (when $i=0$) or $\tilde{\mathscr{W}}_i$ (when $i>0$).

2. A NON-SYMMETRIC GENERALIZATION OF THE PRECONDITIONER

It is proved in Reference [1] that, for a symmetric positive definite (SPD) stiffness matrix A, (14) is an optimal two level preconditioner. That is, the condition number of $M^{-1}A$ is bounded independently of h, H and p, where the tetrahedral elements at level J have a diameter of $O(h)$. Following the ideas of [3], for example, one may also consider a *restricted* version of this preconditioner that is not symmetric even when each A_i is. This is obtained by replacing the R_i^{T} terms in (14) with simpler alternatives which significantly reduce the amount of computation and communication required in a parallel implementation. The form of this restricted preconditioner is given by

$$\tilde{M}^{-1} = \sum_{i=0}^{p} \Pi_i^{\mathrm{T}} A_i^{-1} R_i \tag{15}$$

where Π_i^{T} is a simple prolongation matrix, all entries of which are zero with the following exceptions. For $i>0$, if the node at which the nth local basis function of $\tilde{\mathscr{W}}_i$ is non-zero is inside Ω_i then the entry in position (m,n) of Π_i^{T} is 1, where m is the number of this node in $\mathscr{T}_J$. Also, if the node at which the nth local basis function of $\tilde{\mathscr{W}}_i$ is non-zero is on the interface of Ω_i then the entry in position (m,n) of Π_i^{T} is $1/q$, where m is as before and q is the number of neighbouring subdomains that share this interface node. For $i=0$, entry (m,n) of Π_0^{T} is equal to 1 where m is the number of node in $\mathscr{T}_J$ which corresponds to each node n in $\mathscr{T}_0$.

Unlike for (14) a proof of optimality of preconditioner (15) is not offered, however empirical evidence suggests that it may be optimal and that it performs better than (14) for typical test problems. For example, Table I shows the number of iterations required to solve a simple 3-D potential flow problem of the form

$$-\Delta\phi = f \quad \text{in } \Omega = (0,2)\times(0,1)\times(0,1) \tag{16}$$

subject to Dirichlet boundary conditions throughout $\partial\Omega$. A Galerkin FE discretization is used on a sequence of uniformly refined meshes and the 2-norm of the residual is reduced by a factor of 10^5 in each case. Since (16) is a self-adjoint problem the discretization yields

Table I. The performance of the symmetric AS preconditioner (14) and the reduced AS preconditioner (15) on the Galerkin FE discretization of (16).

	Preconditioner (14)				Preconditioner (15)			
Elements/Procs.	2	4	8	16	2	4	8	16
6144	6	11	16	19	3	3	4	4
49 152	7	12	17	19	3	5	5	6
393 216	8	13	19	20	4	6	7	8
3 145 728	9	13	19	20	4	8	9	9

Figures quoted represent the number of iterations required to reduce the initial residual by a factor of 10^5 on a sequence of uniform refinements of an initial mesh of 768 tetrahedral elements.

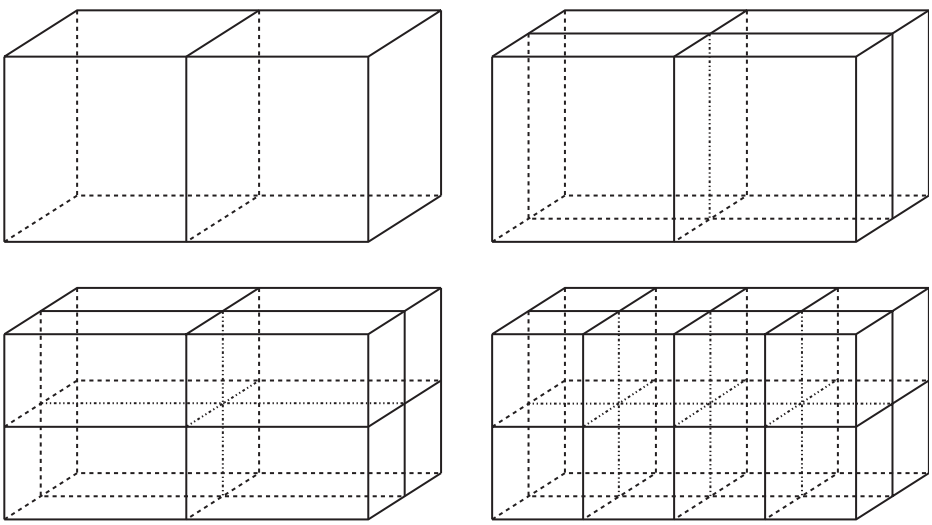

Figure 1. An illustration of the partitioning strategy, based upon recursive co-ordinate bisection, used to obtain 2, 4, 8 and 16 subdomains for our test problems where $\Omega = (0,2) \times (0,1) \times (0,1)$.

a SPD stiffness matrix A. In this case the preconditioner M, given by (14), is also SPD however $\tilde{M}$, given by (15), is not. Hence, although the conjugate gradient method [4, 5] may be used with M as the preconditioner, a more general iterative technique, such as GMRES [6], must be used in the restricted case. Nevertheless, Table I clearly demonstrates that this minor disadvantage of (15) over (14) is more than compensated for by the reduction in the number of iterations required. Furthermore, the computational (and, in the parallel implementation, communication) cost of applying $\tilde{M}$ at each iteration is less than that of applying M.

It should be noted that the results presented in Table I, and all other results presented in this paper, are for one particular set of partitions of the domain $\Omega = (0,2) \times (0,1) \times (0,1)$. These are obtained by using a very simple recursive co-ordinate bisection algorithm (e.g. Reference [7]) and, in this case, provide compact subdomains with a relatively small surface-area to volume ratio, as illustrated in Figure 1. It is shown in Reference [8] that the shape

Table II. The performance of the reduced AS preconditioner (15) on the streamline diffusion FE discretization of (17) for two choices of ε.

	$\varepsilon = 10^{-2}$				$\varepsilon = 10^{-3}$			
Elements/Procs.	2	4	8	16	2	4	8	16
6144	3	4	4	6	5	5	5	7
49 152	3	4	4	6	4	5	5	7
393 216	3	4	5	7	4	5	5	6
3 145 728	3	5	6	8	3	4	5	7

Figures quoted represent the number of iterations required to reduce the initial residual by a factor of 10^5 on a sequence of uniform refinements of an initial mesh of 768 tetrahedral elements.

of the subdomains can have some bearing on the precise performance of preconditioner (15), however, this issue will not be addressed here.

A further advantage of the preconditioner $\tilde{M}$ over M is that it may be applied quite naturally to the solution of non-symmetric or indefinite linear systems arising from the discretization of non-self-adjoint PDEs. For example, Table II shows the number of preconditioned GMRES iterations required to solve two convection–diffusion problems of the form

$$-\varepsilon\Delta u + \underline{b}\cdot\underline{\nabla}u = f \quad \text{in } \Omega = (0,2)\times(0,1)\times(0,1) \tag{17}$$

on the same sequence of meshes as used in Table I. In each case $\underline{b} = (1,0,0)^{\mathrm{T}}$, Dirichlet boundary conditions have been applied throughout $\partial\Omega$ and a stabilized (streamline diffusion [9]) FE discretization has been used. Again only a very slow growth in the number of iterations is observed as h is decreased or p is increased.

3. PARALLEL IMPLEMENTATION ISSUES

The parallel implementation of the preconditioned GMRES solver using $\tilde{M}$ as the preconditioner may be undertaken in a number of ways. In particular, the coarse grid solve, $A_0^{-1}\underline{z}_0 = \underline{r}_0$ say, that must be applied at each iteration can be undertaken either in parallel or sequentially. For the two level algorithm described here it is expected that this coarse grid problem will be relatively small and so the overhead of a parallel solve is likely to be prohibitive. Hence, this problem is solved sequentially and, following the philosophy of [10], it is repeated on all processors rather than being undertaken on a single processor and the result broadcast, as in Reference [11] for example.

In fact, for the parallel implementation used in this work it has been assumed that each subdomain is assigned to a single processor, so that the coarse grid solve may be combined with each of the local subdomain solves as follows. Each processor, i say, takes its own copy of the partitioned coarse mesh $\mathscr{T}_0$ but only refines this mesh inside $\tilde{\Omega}_{i,k-1}$ at step k of the refinement process (i.e. from $\mathscr{T}_{k-1}$ to $\mathscr{T}_k$). The continuous piecewise linear FE spaces on the resulting meshes are then

$$\mathscr{U}_i = \mathscr{W}_0 \cup \tilde{\mathscr{W}}_i \tag{18}$$

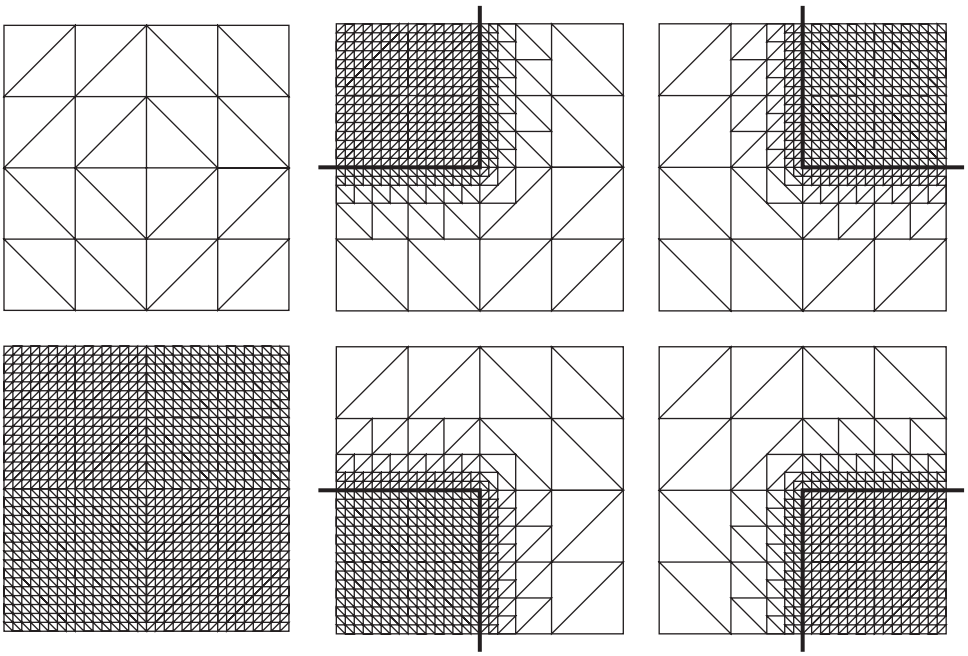

Figure 2. A 2-D illustration of a simple coarse mesh (top left) and a corresponding uniformly refined final mesh (bottom left) along with the actual meshes generated on four different processors (where the coarse mesh has been locally refined only inside $\tilde{\Omega}_{i,k}$ at level k of the refinement process).

for $i = 1, \ldots, p$. Corresponding meshes are illustrated for a simple 2-D example in Figure 2. Note that in this figure there are a number of 'slave' nodes in each processor's mesh which cause these meshes to be non-conforming. The solution values at these nodes are not free: they are determined by the nodal values at the ends of the edges on which the slave nodes lie. For a practical implementation it turns out to be simpler to allow the solution values at these nodes to be free by performing an interior refinement of those elements on the unrefined sides of the edges that have 'hanging' nodes on them. In the 2-D example of Figure 2 this simply involves bisecting all triangular elements containing a hanging node however for 3-D problems, such as those considered in this paper, this intermediate refinement is a little more complicated (see Reference [12] for details of the 'green' refinement stencils used here).

Having obtained a mesh on each processor it is possible for processor i to assemble the local FE problem for its mesh independently of the other processors. For a linear PDE this algebraic system may be expressed as

$$\begin{bmatrix} A_i & 0 & B_i \\ 0 & \bar{A}_i & \bar{B}_i \\ C_1 & \bar{C}_2 & A_{i,s} + \bar{A}_{i,s} \end{bmatrix} \begin{bmatrix} \underline{u}_i \\ \bar{\underline{u}}_i \\ \underline{u}_{i,s} \end{bmatrix} = \begin{bmatrix} \underline{f}_i \\ \bar{\underline{f}}_i \\ \underline{f}_{i,s} + \bar{\underline{f}}_{i,s} \end{bmatrix} \tag{19}$$

Here $\underline{u}_i$ are the unknowns inside Ω_i, $\bar{\underline{u}}_i$ are the unknowns outside $\bar{\Omega}_i$ and $\underline{u}_{i,s}$ are the unknowns on the interface of Ω_i. Correspondingly, A_i, B_i, C_i, $A_{i,s}$ and $\underline{f}_{i,s}$ are matrix and vector blocks

assembled over elements inside Ω_i, and $\bar{A}_i$, $\bar{B}_i$, $\bar{C}_i$, $\bar{A}_{i,s}$ and $\underline{\bar{f}}_{i,s}$ are matrix and vector blocks assembled over elements outside Ω_i on processor i.

Note that in addition to the block notation used in (19) the stiffness matrix A may also be written in a global block structure:

$$A = \begin{bmatrix} A_1 & & & B_1 P_1 \\ & \ddots & & \vdots \\ & & A_p & B_p P_p \\ P_1^{\mathrm{T}} C_1 & \cdots & P_p^{\mathrm{T}} C_p & A_I \end{bmatrix} \tag{20}$$

where $A_I = \sum_{i=1}^{p} P_i^{\mathrm{T}} A_{i,s} P_i$ for some Boolean prolongation matrices P_i^{T} which are used to extend a vector of nodal values on the interface of Ω_i to a larger vector of all subdomain interface values. Hence a matrix–vector product, $\underline{w} = A\underline{v}$ say, may be calculated in parallel as

$$\underline{w}_i = A_i \underline{v}_i + B_i P_i \underline{v}_I \tag{21}$$

$$\underline{w}_I = \sum_{i=1}^{p} (P_i^{\mathrm{T}} C_i \underline{v}_i + P_i^{\mathrm{T}} A_{i,s} P_i \underline{v}_I) \tag{22}$$

$$= \sum_{i=1}^{p} P_i^{\mathrm{T}} \underline{w}_{i,s} \quad \text{say} \tag{23}$$

Similarly, a vector inner product may also be calculated efficiently in parallel using the blocks that are stored on each processor.

Hence, the only remaining computationally significant task in the preconditioned GMRES algorithm to be considered is the parallel implementation of the preconditioner itself. The overall effect of the modifications outlined at the start of this section are to yield a preconditioner that now takes the form

$$\hat{M}^{-1} = \sum_{i=1}^{p} \hat{\Pi}_i^{\mathrm{T}} \hat{A}_i^{-1} \hat{R}_i \tag{24}$$

where $\hat{R}_i$ is the projection matrix from $\mathscr{T}_J$ to the mesh stored on processor i, $\hat{A}_i$ is the matrix on the left-hand side of (19) and $\hat{\Pi}_i^{\mathrm{T}}$ is the simple prolongation matrix which combines Π_0^{T} and Π_i^{T} in (15). There are two main implementation issues associated with solving a preconditioning system of the form $\hat{M}\underline{z} = \underline{r}$ at each iteration in parallel. The first of these is the calculation of the restriction $\hat{R}_i \underline{r}$, which requires an all-to-one communication for *each* processor i. The second main issue that must be addressed is the solution of subproblems of the form $\hat{A}_i \underline{z} = \hat{R}_i \underline{r}$ on each processor. These systems are of form (19), however, and may therefore be solved simultaneously on each processor without the need for any further communication. The prolongation operations $\hat{\Pi}_i^{\mathrm{T}} \underline{z}$ are extremely cheap to compute.

In Table III some parallel timings are presented for the solution of the convection–diffusion test problem (14) on the finest meshes used in Table II (i.e. using 3 145 728 tetrahedral elements). The subproblems of form (19) have been solved sequentially (and only approximately, for improved overall efficiency) using an incomplete LU preconditioned GMRES solver [13]

Table III. Timings for the parallel solution using the stabilized FE discretization of (17) for two choices of ε.

	$\varepsilon = 10^{-2}$					$\varepsilon = 10^{-3}$				
Processors	1	2	4	8	16	1	2	4	8	16
Solution time	770.65	484.53	347.61	228.39	136.79	688.12	442.44	277.78	187.16	108.75
Speed-up	—	1.6	2.2	3.4	5.6	—	1.6	2.5	3.7	6.3

The solution times are quoted in seconds and the speed-ups are relative to the best sequential solution time.

and this is also the solver used for the sequential times quoted. All timings are in seconds on a shared memory SG Origin2000 computer.

4. DISCUSSION

There are a number of factors that limit the efficiency of the current parallel implementation. In all cases the sequential time of the AS preconditioned solver is greater than that of the sequential solver used (based upon ILU preconditioning, which does not naturally parallelize), so even a perfect parallel implementation would not deliver 100% efficiency. Furthermore, the mesh partitioning strategy used in this work is extremely basic and could certainly be improved to yield faster convergence and, in the case of 16 or more subdomains, better load-balancing. Also, our decision to solve the coarse grid problem on every processor does result in a greater total amount of communication and, whilst much of this can be overlapped with useful computations, some additional parallel overhead is observed.

Overall, however, in this paper the potential of the two level AS preconditioner considered has been demonstrated for the solution of a range of FE problems in three dimensions. Not only does the preconditioner yield optimal convergence rates for symmetric problems, as predicted in Reference [1], but it also appears to perform optimally in its non-symmetric reduced form and when applied to non-symmetric model flow problems such as convection–diffusion equations. Furthermore, the practical parallel implementation of this approach has been successfully demonstrated and encouraging results have been obtained on moderate numbers of processors. Further work is underway to better understand the effects of different mesh partitioning strategies so as to improve the parallel efficiency and to apply the technique as part of an adaptive strategy, as in References [10, 14] for example. Extensions to non-linear equations and systems are also being investigated.

REFERENCES

1. Bank RE, Jimack PK, Nadeem SA, Nepomnyaschikh SV. A weakly overlapping domain decomposition method for the finite element solution of elliptic partial differential equations. *SIAM Journal on Scientific Computing* 2002; **23**:1818–1842.
2. Smith B, Bjorstad P, Gropp W. *Domain Decomposition*: *Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press: Cambridge, 1996.
3. Cai X-C, Sarkis M. An restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* 1999; **21**:792–797.
4. Ashby SF, Manteuffel TA, Taylor PE. A taxonomy for conjugate gradient methods. *SIAM Journal on Numerical Analysis* 1990; **27**:1542–1568.

5. Golub GH, van Loan CF. *Matrix Computations* (3rd edn). John Hopkins Press: Baltimore, MD, 1996.
6. Saad Y, Schultz M. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1986; **7**:856–869.
7. Simon HD. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 1991; **2**:135–148.
8. Nadeem SA. Parallel domain decomposition preconditioning for the adaptive finite element solution of elliptic problems in three dimensions. *Ph.D. Thesis*, University of Leeds, 2001.
9. Johnson C. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press: Cambridge, 1987.
10. Bank RE, Holst M. A new paradigm for parallel adaptive meshing algorithms. *SIAM Journal on Scientific Computing* 2000; **22**:1411–1443.
11. Hodgson DC, Jimack PK. A domain decomposition preconditioner for a parallel finite element solver on distributed unstructured grids. *Parallel Computing* 1997; **23**:1157–1181.
12. Speares W, Berzins M. A 3-D unstructured mesh adaptation algorithm for time-dependent shock dominated problems. *International Journal for Numerical Methods in Fluids* 1997; **25**:81–104.
13. Saad Y. SPARSKIT: a basic tool kit for sparse matrix computations, version 2. *Technical Report*, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1994.
14. Bank RE, Jimack PK. A new parallel domain decomposition method for the adaptive finite element solution of elliptic partial differential equations. *Concurrency and Computation*: *Practice and Experience* 2001; **13**:327–350.

