This is a repository copy of *A parallel domain decomposition algorithm for the adaptive finite element solution of 3-D convection-diffusion problems*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/1867/

Book Section:

A Parallel Domain Decomposition Algorithm for the Adaptive Finite Element Solution of 3-D Convection-Diffusion Problems

Peter K. Jimack and Sarfraz A. Nadeem

Computational PDEs Unit, School of Computing,
University of Leeds, Leeds, LS2 9JT, UK
{pkj, sarfraz}@comp.leeds.ac.uk
http://www.comp.leeds.ac.uk/pkj/

Abstract. In this paper we extend our previous work on the use of domain decomposition (DD) preconditioning for the parallel finite element (FE) solution of three-dimensional elliptic problems [3, 6] and convection-dominated problems [7, 8] to include the use of local mesh refinement. The preconditioner that we use is based upon a hierarchical finite element mesh that is partitioned at the coarsest level. The individual subdomain problems are then solved on meshes that have a single layer of overlap at each level of refinement in the mesh. Results are presented to demonstrate that this preconditioner leads to iteration counts that appear to be independent of the level of refinement in the final mesh, even in the case where this refinement is local in nature: as produced by an adaptive finite element solver for example.

1 Introduction

In [3] we introduce a new two level additive Schwarz (AS) DD preconditioner based upon the existence of a nested sequence of meshes on each subdomain. This preconditioner is proved to be optimal for a particular class of symmetric self-adjoint partial differential equations (PDEs) in both two and three dimensions. In order to construct the preconditioner it is necessary to generate a decomposition of the finite element space, $\mathcal{W}$ say, in the following manner. The description here is given in two dimensions for simplicity but the extension to a tetrahedral mesh in three dimensions is straightforward (and is outlined explicitly in [10]).

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

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

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

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

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

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

We now permit $\mathcal{T}_0$ to be refined several times, to produce a family of triangulations, $\mathcal{T}_0, ..., \mathcal{T}_J$, where each triangulation, $\mathcal{T}_k$, consists of N_k triangular elements, $\tau_j^{(k)}$, such that

$$\overline{\Omega} = \bigcup_{j=1}^{N_k} \tau_j^{(k)} \quad \text{and} \quad \mathcal{T}_k = \{\tau_j^{(k)}\}_{j=1}^{N_k} \, . \tag{5}$$

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

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

(Here the level of a triangle is defined to be the least value of k for which that triangle is an element of $\mathcal{T}_k$.) In addition to the above we will also require that:

4. in the final mesh, $\mathcal{T}_J$, all pairs of triangles 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 we next define the restrictions of each of these triangulations onto each subdomain by

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

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

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

Following this we introduce the FE spaces associated with these local triangulations. Let G be some triangulation and denote by $\mathcal{S}(G)$ the space of continuous piecewise linear functions on G. Then we can make the following definitions:

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

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

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

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

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

It is evident that

$$\mathcal{W} = \tilde{\mathcal{W}}_0 + \tilde{\mathcal{W}}_1 + ... + \tilde{\mathcal{W}}_p \tag{13}$$

and this is the decomposition that forms the basis of the two level additive Schwarz preconditioner (see [13] for example), M say, in [3]. Hence, for a global FE stiffness matrix A, this preconditioner takes the form:

$$M^{-1} = \sum_{i=0}^{p} \tilde{R}_i^T \tilde{A}_i^{-1} \tilde{R}_i \tag{14}$$

where (using the usual local bases for $\mathcal{W}$ and $\tilde{\mathcal{W}}_i$) $\tilde{R}_i$ is the rectangular matrix representing the L^2 projection from $\mathcal{W}$ to $\tilde{\mathcal{W}}_i$, and $\tilde{A}_i$ is the FE stiffness matrix corresponding to the subspace $\tilde{\mathcal{W}}_i$.

A parallel implementation of the above preconditioner is described in 2-d in [2] and in 3-d in [7], for example. In both of these cases, the coarse grid solve is combined with each of the subdomain solves to yield a preconditioner which is actually of the form

$$M^{-1} = \sum_{i=1}^{p} R_i^T A_i^{-1} R_i \;. \tag{15}$$

Here, R_i and A_i differ from $\tilde{R}_i$ and $\tilde{A}_i$ in (14) since R_i now represents the projection from $\mathcal{W}$ to $\tilde{\mathcal{W}}_0 \cup \tilde{\mathcal{W}}_i$ for $i = 1, ..., p$. In addition to this, and following [5], both the 2-d and the 3-d parallel implementations in [2,7] use a restricted version of the AS preconditioner (15). This involves the following simplification:

$$M^{-1} = \sum_{i=0}^{p} D_i A_i^{-1} R_i \;, \tag{16}$$

where D_i is a diagonal matrix with entries of 1 for those rows corresponding to vertices inside subdomain i, 0 those rows corresponding to vertices outside subdomain i, and $\frac{1}{q}$ for those rows corresponding to vertices on the interface of subdomain i (shared with $q - 1$ neighbouring subdomains).

Results presented in [2,7] show that the preconditioner (16) outlined above appears to behave in an optimal manner when applied to a range of problems as the FE mesh is uniformly refined. That is, the number of iterations required to obtain a converged solution appears to be bounded independently of the mesh size h. In particular, in [7] it is demonstrated that when (16) is applied to solve three-dimensional convection-dominated problems using a stabilized (streamline-diffusion) FE discretization, the quality of the preconditioning is excellent.

2   Local Mesh Refinement

In our previous work in three dimensions (e.g. [6–8]) we have focused on applying the finite element method on sequences of uniformly refined meshes. For many practical problems however the solution exhibits local behaviour, such as

the existence of shocks or boundary layers, which lead to the requirement for meshes obtained via local, rather than global, refinement. Consider, for example, a convection-diffusion equation of the form

$$-\varepsilon \Delta u + \underline{b} \cdot \underline{\nabla} u = f \tag{17}$$

on some domain $\Omega \subset \Re^3$. When $|\underline{b}| \gg \varepsilon > 0$ it is common for the solution to involve boundary layers of width $O(\frac{\varepsilon}{|\underline{b}|})$, depending on the precise nature of the boundary conditions. If the solution is smooth away from these layers then it is most efficient, computationally, to concentrate the majority of the degrees of freedom in and around the layers. The following test problem is of the form (17), with a computational domain $\Omega = (0,2) \times (0,1) \times (0,1)$ and exact Dirichlet boundary conditions applied throughout $\partial\Omega$:

$$\left. \begin{array}{l} \underline{b} = (1,0,0)^T \,, \\ f = 2\varepsilon \left(x - \frac{2(1-e^{x/\varepsilon})}{(1-e^{2/\varepsilon})} \right) \left(y(1-y) + z(1-z) \right) + y(1-y)z(1-z) \,, \\ u = \left(x - \frac{2(1-e^{x/\varepsilon})}{(1-e^{2/\varepsilon})} \right) y(1-y)z(1-z) \,. \end{array} \right\} \tag{18}$$

The streamline-diffusion discretization of (17) on a mesh, $\mathcal{T}_J$, of tetrahedra seeks a piecewise linear solution of the form

$$u^h = \sum_{i=1}^{N+B} u_i N_i(\underline{x}) \tag{19}$$

where N_i are the usual linear basis functions, N is the number of nodes of $\mathcal{T}_J$ inside Ω and B is the number of nodes of $\mathcal{T}_J$ on $\partial\Omega$. The values of the unknowns u_i are determined so as to satisfy the finite element equations

$$\varepsilon \int_\Omega \underline{\nabla} u_h \cdot \underline{\nabla}(N_j + \alpha \underline{b} \cdot \underline{\nabla} N_j) \, d\underline{x} + \int_\Omega (\underline{b} \cdot \underline{\nabla} u_h)(N_j + \alpha \underline{b} \cdot \underline{\nabla} N_j) \, d\underline{x} =$$
$$\int_\Omega f(\underline{x})(N_j + \alpha \underline{b} \cdot \underline{\nabla} N_j) \, d\underline{x} \tag{20}$$

for $j = 1, ..., N$ (see, for example, [9] for further details). This yields an $N \times N$ non-symmetric linear system of the form

$$K \underline{u} = \underline{f} \,. \tag{21}$$

In [7] it is shown that when the above DD preconditioner, (16), is applied with an iterative method (GMRES, [11, 12]) to solve (21) on a sequence of uniformly refined meshes excellent iteration counts are obtained. Table 1, taken from [7], illustrates this for two choices of ε. The precise values for the iteration counts obtained using this preconditioner depend on the particular partition of the coarse mesh into subdomains (see (2), (3) and (4)) that is used. For all of the calculations presented in this paper we partition the domain Ω with cuts that are parallel to the convection direction $\underline{b}$ in (17). For the specific test problem,

Table 1. The performance of the preconditioner (16) on the system (21), the discretization of (17) and (18), using global mesh refinement: figures quoted represent the number of iterations required to reduce the initial residual by a factor of 10^5.

Elements/Procs.	$\varepsilon = 10^{-2}$				$\varepsilon = 10^{-3}$			
	2	4	8	16	2	4	8	16
6144	2	3	3	3	2	3	3	3
49152	3	3	4	4	3	4	4	4
393216	3	4	4	5	3	4	4	4
3145728	4	5	6	6	3	4	4	4

Fig. 1. An illustration of the partitioning strategy, based upon cuts parallel to $\underline{b} = (1, 0, 0)^T$, used to obtain 2, 4, 8 and 16 subdomains, where $\Omega = (0, 2) \times (0, 1) \times (0, 1)$.
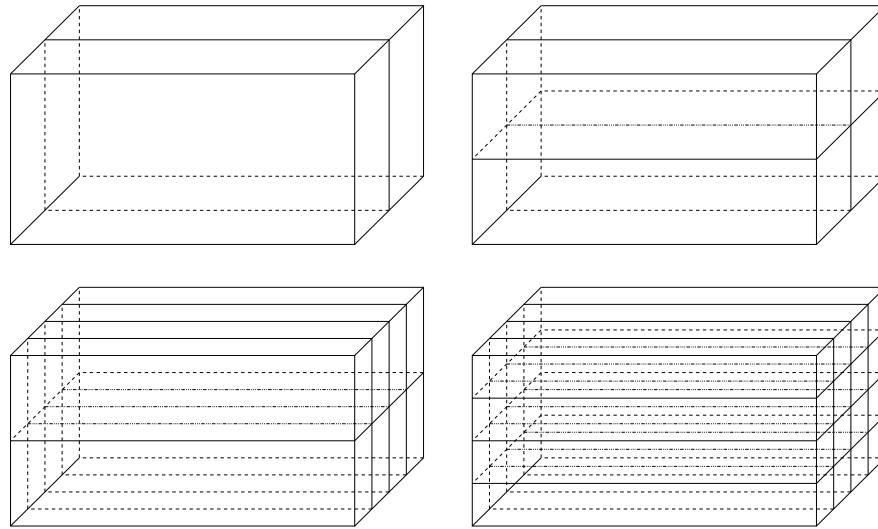
(18), considered here this means that all cuts are parallel to the x-axis, as illustrated in Fig. 1. This strategy is shown in [10] to yield convergence in fewer iterations than is possible with more isotropic partitions when (17) is convection-dominated. Furthermore, in the following section on parallel results, we are able to ensure good load balancing when using this strategy, even in the presence of local refinement.

We now consider the use of a simple *a priori* local mesh refinement strategy for the stabilized finite element solution of (17), (18). The purpose of this is to illustrate the potential for the DD preconditioner (16) to be used successfully within an adaptive FE framework, a discussion of which is included in Section 4. For this particular test problem however we make use of the fact that the only boundary layer in the solution is known to be next to the domain boundary $x = 2$, and that the solution is smooth elsewhere. Hence, beginning with a

Table 2. The performance of the preconditioner (16) on the system (21), the discretization of (17) and (18), using local mesh refinement: figures quoted represent the number of iterations required to reduce the initial residual by a factor of 10^5.

Elements/Procs.	$\varepsilon = 10^{-2}$				$\varepsilon = 10^{-3}$			
	2	4	8	16	2	4	8	16
2560	4	5	5	6	5	6	6	6
9728	5	5	6	6	5	6	6	6
38400	5	6	6	7	5	6	7	7
153088	6	8	8	8	6	7	7	7

mesh of 768 elements, we are able to get results of a similar accuracy to those obtained using uniform mesh refinement by applying local mesh refinement to the same number of levels: only refining elements in the neighbourhood of $x = 2$ at each level. The iteration counts obtained using the DD preconditioner with this mesh refinement strategy are shown in table 2. Although slightly higher than the corresponding numbers of iterations shown in table 1, we again see that the results appear to be bounded independently of both h and p.

3   Parallel Results

The parallel implementation of our preconditioner (16) is described in detail in two and three dimensions respectively in [2, 7]. The use of local mesh refinement does not alter this implementation in any way however it does require extra attention to be paid to the issue of load balancing. This issue is of fundamental importance in the parallel adaptive FE solution of PDEs and is beyond the scope of this paper: see, for example, [14]. For the test problems described in this section all refinement is applied in a neighbourhood of the domain face at $x = 2$ and the partitions illustrated in Fig. 1 are used. Whilst the overlapping nature of the preconditioner leads to a load balance that is far from perfect in the cases $p = 8$ and $p = 16$, it is sufficiently good to yield quite respectable parallel speed-ups, as demonstrated below.

In addition to the potential load imbalance there are at least two more significant factors that affect the performance of our parallel implementation of (16). The first of these is the quality of (16) as a preconditioner, and the second is the parallel overhead associated with inter-processor communications. The numerical tests described in this work were all undertaken on a tightly coupled parallel computer, the SG Origin2000, and so communication overheads are quite moderate in the results presented. The quality of (16) as a preconditioner is considerably more important however. In all of the tables of timings we include not only the parallel solution times but also the sequential solution times for different choices of p. As can be seen, these times vary enormously and are generally significantly slower than the fastest sequential solution time that we are able to obtain (using [11]). For this reason we present two speed-up rows

Table 3. Timings for the parallel DD solver applied to problem (17), (18) with $\varepsilon = 10^{-2}$ using four levels of local mesh refinement.

Processors/Subdomains	1	2	4	8	16
Parallel Time	26.20	17.54	10.54	7.17	5.35
Speed-up	–	1.5	2.9	3.7	4.9
Sequential Time	–	34.64	40.54	50.35	66.96
Parallel Speed-up	–	2.0	3.8	7.0	12.5

Table 4. Timings for the parallel DD solver applied to problem (17), (18) with $\varepsilon = 10^{-3}$ using four levels of local mesh refinement.

Processors/Subdomains	1	2	4	8	16
Parallel Time	20.07	14.02	8.62	6.02	4.87
Speed-up	–	1.4	2.3	3.3	4.1
Sequential Time	–	27.64	33.12	42.05	58.88
Parallel Speed-up	–	2.0	3.8	7.0	12.1

in each table: a regular speed-up which contrasts the parallel solution time with the best sequential solution time, and a parallel speed-up which contrasts the parallel solution time with the sequential time of the p subdomain DD solver.

Tables 3 and 4 present parallel timings for the entire solution algorithm when solving problem (17), (18) on the final mesh obtained using the *a priori* adaptive algorithm (i.e. containing 153088 elements). These timings are for $\varepsilon = 10^{-2}$ and 10^{-3} respectively. It should be noted that equivalent results are presented for the first of these cases in [7] using global mesh refinement to a similar resolution. In that paper parallel solution times of between 505.7 seconds ($p = 2$) and 144.7 seconds ($p = 16$) are reported. The use of local refinement clearly leads to a significant improvement on these times therefore.

An initial assessment of the figures presented in Tables 3 and 4 shows that a speed-up of about 5 has been achieved on 16 processors. Whilst this may be a little disappointing, inspection of the parallel speed-ups achieved gives much greater grounds for optimism. The major cause of loss of efficiency in the overall parallel solution appears to be due to the growth in the solution time of the sequential version of the p subdomain algorithm as p increases. If this can be overcome, through the use of a sequential multigrid solver (e.g. [4]) for each subdomain solve for example, then the parallel speed-ups suggest that there is significant scope for the use of this solution algorithm.

It is also apparent from the results in Tables 3 and 4 that the second most important factor in determining parallel performance is the quality of the load balance across the processors. When $p = 2$ and $p = 4$, as may be observed from Fig. 1, each subdomain has the same boundary area and will therefore have the same number of overlapping elements into neighbouring subdomains. When

$p = 8$ or $p = 16$ this is not the case however: with some subdomains (in the centre) having a larger overlap than others. In these latter cases the individual subdomain problems are of a different size and so the parallel efficiency can be expected to deteriorate. This is clearly observed in Tables 3 and 4 with a parallel speed-up of 3.8 on 4 processors compared to a parallel speed-up of 7.0 on 8 processors and just 12.1/12.5 on 16 processors. The communication overhead associated with the parallel implementation would appear to be the least significant of the three main factors identified above as leading to losses in parallel efficiency.

4   Discussion

The results presented in the previous section demonstrate that the modified additive Schwarz preconditioner, that was successfully applied in parallel to convection-dominated problems in three dimensions using global mesh refinement in [7, 8], may also be successfully applied in parallel to the same class of problem using local mesh refinement. The algorithm is shown to parallelize very efficiently, although there is an additional complexity in ensuring the load balance of the overlapping grids on each subdomain when local refinement is used. Other than this, the major constraint on the efficiency of the algorithm comes from the observed growth in the sequential solution time as p increases. From Table 2 it is clear that the number of iterations taken does not grow with p, so it is apparent that the work per iteration must be growing as the number of subdomains is increased. This is inevitable in the sense that the total size of the overlap region increases with p however there may be scope for improving this aspect of the solution algorithm through the use of a sequential multigrid solver for the subdomain problems.

Whilst the results described in this paper demonstrate that the proposed preconditioner may be applied as part of a parallel adaptive finite element algorithm, we have not explicitly considered such an algorithm here. The examples used in the previous section are based upon a mesh refinement strategy that is known *a priori* and therefore allows a partition of Ω to be made that is guaranteed to lead to reasonably well load-balanced subdomain problems once refinement has taken place. Developing a strategy for incorporating this DD solution method within a robust adaptive mesh refinement procedure, based upon *a posteriori* error estimation for example, is a current area of research. One approach that has been proposed, that involves partitioning the coarse mesh, $\mathcal{T}_0$, based upon an initial *a posteriori* error estimate, is described in [1]. It is our intention to use this technique as a framework for the implementation of a prototype adaptive algorithm that further exploits the DD solution strategy described here.

Acknowledgments

References

1. R.E. Bank and M. Holst, *"A New Paradigm for Parallel Adaptive Meshing Algorithms"*, SIAM J. on Scientific Computing, 22, 1411-1443, 2000.
2. R.E. Bank and P.K. Jimack, *"A New Parallel Domain Decomposition Method for the Adaptive Finite Element Solution of Elliptic Partial Differential Equations"*, Concurrency and Computation: Practice and Experience, 13, 327–350, 2001.
3. R.E. Bank, P.K. Jimack, S.A. Nadeem and S.V. Nepomnyaschikh, *"A Weakly Overlapping Domain Decomposition Method for the Finite Element Solution of Elliptic Partial Differential Equations"*, to appear in SIAM J. on Scientific Computing, 2002.
4. R.E. Bank and J. Xu, *"A Hierarchical Basis Multigrid Method for Unstructured Meshes"*, in Tenth GAMM-Seminar Kiel on Fast Solvers for Flow Problems (W. Hackbusch and G. Wittum, eds.), Vieweg-Verlag, Braunschweig, 1995.
5. X.-C. Cai and M. Sarkis, *"An Restricted Additive Schwarz Preconditioner for General Sparse Linear Systems"*, SIAM J. on Scientific Computing, 21, 792–797, 1999.
6. P.K. Jimack and S.A. Nadeem, *"A Weakly Overlapping Parallel Domain Decomposition Preconditioner for the Finite Element Solution of Elliptic Problems in Three Dimensions*, in Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Volume III, ed. H.R. Arabnia (CSREA Press, USA), pp.1517–1523, 2000.
7. P.K. Jimack and S.A. Nadeem, *"Parallel Application of a Novel Domain Decomposition Preconditioner for the Stable Finite Element Solution of Three-Dimensional Convection-Dominated PDEs*, in Euro-Par 2001 Parallel Processing: 7th International Euro-Par Conference Manchester, UK, August 2001 Proceedings, ed. R. Sakellariou et al. (Lecture Notes in Computer Science 2150, Springer), pp.592–601, 2001.
8. P.K. Jimack and S.A. Nadeem, *" A Weakly Overlapping Parallel Domain Decomposition Preconditioner for the Finite Element Solution of Convection-Dominated Problems in Three-Dimensions*, to appear in proceedings of Parallel CFD 2001, Egmond aan Zee, NL, 21-23 May, 2001.
9. C. Johnson *"Numerical Solution of Partial Differential Equations by the Finite Element Method"*, Cambridge University Press, 1987.
10. S.A. Nadeem *"Parallel Domain Decomposition Preconditioning for the Adaptive Finite Element Solution of Elliptic Problems in Three Dimensions"*, Ph.D. Thesis, University of Leeds, 2001.
11. 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).
12. Y. Saad, and M. Schultz, *"GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems"*, SIAM J. on Sci. Comp. 7, 856–869, 1986.
13. B. Smith, P. Bjorstad and W. Gropp, *"Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations"*, Cambridge University Press, 1996.
14. N. Touheed, P. Selwood, P.K. Jimack and M. Berzins, *"A Comparison of Some Dynamic Load-Balancing Algorithms for a Parallel Adaptive Flow Solver"*, Parallel Computing, 26, 1535–1554, 2000.