Domain Decomposition Using a 2-Level Correction Scheme

R.H. Marsden, T.N. Croft, and C.-H. Lai

School of Computing and Mathematical Sciences, University of Greenwich,
Greenwich,
London SE10 9LS, U.K.
{R.H.Marsden, T.N.Croft, C.H.Lai}@gre.ac.uk
http://cms1.gre.ac.uk

Abstract. The PHYSICA software was developed to enable multiphysics modelling allowing for interaction between Computational Fluid Dynamics (CFD) and Computational Solid Mechanics (CSM) and Computational Aeroacoustics (CAA). PHYSICA uses the finite volume method with 3-D unstructured meshes to enable the modelling of complex geometries. Many engineering applications involve significant computational time which needs to be reduced by means of a faster solution method or parallel and high performance algorithms. It is well known that multigrid methods serve as a fast iterative scheme for linear and nonlinear diffusion problems. This papers attempts to address two major issues of this iterative solver, including parallelisation of multigrid methods and their applications to time dependent multiscale problems.

1 Introduction

The PHYSCIA software [6][15] was developed to enable multiphysics modelling allowing for interaction between Computational Fluid Dynamics (CFD) and Computational Solid Mechanics (CSM) and Computational Aeroacoustics (CAA). PHYSICA uses the finite volume method with 3-D unstructured meshes to enable the modelling of complex geometries. Many engineering applications involve significant computational time which needs to be reduced by means of a faster solution method or parallel and high performance algorithms.

It is well known that multigrid methods serve as a fast iterative scheme for linear and nonlinear diffusion problems. There are two major issues in this fast iterative solver. First, multigrid methods are usually very difficult to parallelise and the performance of the resulting algorithms are machine dependent. Early work in parallelisation of multigrid methods include Barkai and Brandt [2], Chan [5], Frederickson [10], Naik [14], etc. Methods developed by these authors concerned the load balancing between processors and the full use of all co-existing coarse level meshes in order to fit into the parallelism requirement. This paper intends to address these issues with particular attention being paid to linear and nonlinear diffusion type of problems in a distributed computing environment. The method is then extended to time-dependent and multiscale problems.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2330, pp. 480–489, 2002.

2 Classical Multigrid Methods for Linear Diffusion Problems

The first article which describes a 'true' multigrid technique was probably published by Fedorenko [9]. He formulated a multigrid algorithm for the standard five-point finite difference discretisation of the Poisson equation on a unit square and showed the computational work is of $O(N)$, where N is the number of unknowns. An extension of the concept to a general linear elliptic problem defined in a unit square was given by Bachvalov [1]. Results shown in these articles were not optimal and the method have not been adopted at that time. Practical multigrid algorithms were first published by Brandt [3] in 1973 and, later, in a revised article [4]. There are many excellent review articles and introductory lectures, for examples [17], [16]. The authors do not intend to produce an exhaust list, but related work can be easily found.

2.1 Defect Correction Principle

Suppose one wish to solve the elliptic problem as given below,

$$Lu = f \in \Omega \tag{1}$$

where L is a linear elliptic problem with suitable boundary conditions defined on $\partial\Omega$. The problem is usually discretised by means of a finite volume method, which leads to the discretised system of linear equations,

$$L_h u_h = f_h \in \Omega_h \tag{2}$$

where subscript h denotes a discretised approximation to the corresponding quantity. Here h denotes a typical mesh size being used in the discretisation. Usually smaller h leads to a larger system of linear equations and an iterative method may be used to solve the system. Suppose u_h^* is an approximation obtained by means of an iterative method to the linear system (2), it is possible to compute the defect or residual due to the approximation u_h^* as

$$R_h = f_h - L_h u_h^* \tag{3}$$

For linear operator such as L_h, one obtains the defect equation $L_h(u_h - u_h^*) \equiv L_h u_h - L_h u_h^* = f_h - L_h u_h^* \equiv R_h$, and an iterative scheme may be used to obtain an iterative refinement or a correction v_h, which may be used to correct the approximation u_h^* and compute a better approximation as $u_h^{**} = u_h^* + v_h$. However, such iterative refinement technique does not improve the convergence rate of the iterative method used in the numerical solution process. In essence the convergence rate deteriorates as h decreases.

One purpose of using the concept of multigrid is to avoid the deterioration of the convergence rate. A Fourier smoothing analysis was first introduced by Brandt [4], which explains the role of an iterative method in the context of the above defect correction principle. The smoothing analysis uses the amplification

factor, $g(\lambda_\alpha)$, which involves the ratio of the error at the n-th iterative approximation to the error at the $(n-1)$-th iterative approximation, in measuring the growth or decay of a Fourier error mode during an iteration. Here α is the Fourier mode and $\lambda_\alpha = \pi\alpha/m$ is the wave number, where m is the typical number of grid points along a characteristic length of Ω_h and $\alpha = -m+1, -m+2, \ldots, m$. When α increases the wave number, λ_α, increases and the amplication factor, $g(\lambda_\alpha)$, decreases. In other words Fourier error modes with long wavelengths (α close to 1) decay slowly and with short wavelengths decay rapidly. The smoothing factor can now be easily defined as

$$\rho = \max\{|g(\lambda_\alpha)| : \lambda_\alpha = \pi\alpha/m \ , \qquad \alpha = -m+1, -m+2, \ldots, m\} \tag{4}$$

and $\rho < 1$ denotes that the iterative method is a smoother, i.e. all Fourier error modes of short wave lengths have been damped out. The remaining part consists of a smoothed part, which is of long wave lengths, should be dealt with at a coarser mesh. This is because long wave lengths on a mesh of size h become relatively shorter wave lengths on a mesh of size $2h$.

Therefore based on Brandt's smoothing analysis, one would like to handle Fourier error modes of long wave lengths on a coarser mesh, say H. The above iterative refinement can then be implemented as the following 2-level algorithm.

Iterate to obtain an approximation u_h^*: $L_h u_h = f_h$;
Compute defect: $R_h := f_h - L_h u_h^*$;
$u_H := I_H^h u_h$; $f_H := I_H^h R_h$;
Solve u_H^*: $L_H u_H = f_H \in \Omega_H$ with suitable boundary conditions;
Apply correction: $u_h := u_h^* + I_h^H u_H^*$;

Usually H is taken as $2h$ and there exists a vast amount of literature in 2-level algorithms. These algorithms are being used recursively to form a multigrid algorithm.

2.2 Parallelisation Issues of Multigrid Methods

It is obvious that as the mesh size is doubled, the number of discrete unknowns decreased by a half. This leads to dummy computation when multigrid methods were implemented on SIMD or vector machines such as the DAP [11], Connection Machines and the CDC Cyber 205 [2]. Such dummy computation cannot be avoided if dynamic data structure is not allowed in the programming language, such as FORTRAN. An attempt to examine a possible parallel network to circumvent the disadvantage was presented by Chan et al [5]. However only theoretical performance analysis was given. The idea of using co-existing coarse grids, such that the union of these coarse grids automatically forms the next finer grid, was also discussed in [10]. Similar concept of using coarse grid correction in conjunction with the finest grid on a unigrid was also examined by McCormick and Ruge [13], and the method was shown to be equivalent to multigrid methods.

Many articles and research reports have been devoted to the implementation of multigrid methods, such as [14] to name just one reference, for a distributed

memory machine. The main techniques involve evenly distributing computational load and minimising communication costs. These techniques amount to a data partitioning of the finest level problem being distributed evenly across the processors in the computational system. The finest level problem must be pre-defined making adaptivity extremely difficult in parallel processing.

In the case of using co-existing coarse grids, which may be solved concurrently, may lead to heavy data traffic because these co-existing grids involves mesh points located further away. On the other hand, averaging procedure of all the coarse grids leading to the correction at the fine grid cannot be compared with the fast convergence rate of using the classical multigrid method.

3 Domain Decomposition Methods

The idea of domain decomposition has a long evolving history. Many literatures may be found, and it is not intended to give a full list of these references, but one [7], in this paper. Domain decomposition involves the subdivision of a given problem into a number of subproblems. Each of these subproblems can be solved separately before being combined to give the global solution of the original problem. The subdivision can be done at either the physical problem level or the discretised problem level. At the discretised problem level, the resulting linear system of equations is rearranged as a collection of smaller linear systems which may be solved independently. At the physical problem level, regions governed by different mathematical models or different material properties are identified and decomposed into different subdomains resulting in a number of locally regular subproblems. It should be noted at this stage that the use of a distributed environment is highly suitable for this class of methods [12].

3.1 Block Iterative Methods

Figure 1 shows a rectangular domain which is subdivided into 16 subdomains. Assuming that the subdomains are nonoverlapped and done at the level of the physical problem, then $\bigcup \Omega_i = \Omega$ and $\bigcap \Omega_i = \emptyset$, and the interior boundary or interface, γ, is defined as $\gamma = (\bigcup \partial\Omega_i) \backslash \partial\Omega$. Let $L_i u_i = f_i$ be the subproblem defined in the subdomain $\Omega_i,\ i = 1, \ldots, N_s$. A simultaneous update to each of the subdomains may be achieved by means of the block Jacobi algorithm as given below.

loop
  for $i := 1 \ldots N_s$ do
  Solve $\mathrm{L}_i u_i = f_i$ subject to suitable boundary conditions along $\partial\Omega_i$;
  end-do
  Update: Interior boundary conditions along γ;
Until solution converged;

This block iterative method may be implemented in parallel but the convergence rate of this method is very slow. There is also a need to update the interior boundary conditions.

A similar technique may be applied to overlapped subdomains. Assuming each of the subdomains in Figure 1 is extended across the interior boundary into its neighbouring subdomains where an overlapped region of thickness of the mesh size h is included into the subdomain Ω_i. Suppose $N(i)$ denotes the numbering of the neighbouring subdomains of Ω_i. It is obvious that $\Omega_i \cap \Omega_{N(i)} \neq \emptyset$ and that $\partial\Omega_i$ lies inside $\Omega_{N(i)}$. Therefore an exchange of information between neighbouring subdomains is sufficient to act as an update to the interior boundaries.

loop
  for $i := 1 \ldots N_s$ do
  Solve $L_i u_i = f_i$ subject to suitable boundary conditions along $\partial\Omega_i$;
  end-do
  Exchange information: Interior boundary conditions along γ;
Until solution converged;

This block iterative method may be implemented in parallel but the convergence rate of this method is, again, very slow. The update of the interior boundary conditions is achieved by means of exchanging information in neighbouring subdomains.

3.2 A Parallel Multigrid Algorithm

Since the classical multigrid algorithm experiences partitioning issues for parallel or distributed computing environment, the aim here is to seek for an alternative. The concept here is to employ the above block iterative algorithms, either overlapped and nonoverlapped. For the present purpose, only the overlapped version has been studied because the implementation is straight forward. The algorithm sees the subdomains as shown in Figure 1 as a natural set up for the coarsest mesh H. Since each subdomain may be executed autonomously, one natural idea is to use the defect correction principle as discussed above with H not necessarily equal to $2h$. In fact it is purely up to individual subdomain to determine the finest mesh h.

The coarsest level using the classical multigrid method is equivlant to solve the system,

$$L_H u_H \;=\; f_H \equiv I_H^{H/2} \ldots I_{2h}^{h} R_h -$$

$$(I_H^{H/2} \ldots I_{4h}^{2h} L_{2h} u_{2h}^{*} + I_H^{H/2} \ldots I_{8h}^{4h} L_{4h} u_{4h}^{*} + \ldots + I_H^{H/2} L_{H/2} u_{H/2}^{*}) \qquad (5)$$

Note that in the present parallel multigrid algorithms it is not possible to evaluate the linear combination of the corrections at all intermediate level. However, it should be clear that the dominant source of error comes from the finest level which is projected onto the coarsest level according to $I_H^{H/2} \ldots I_{2h}^{h} R_h$. Here the projection can be done by means of a sequence of linear interpolation.

4 Numerical Examples

The numerical example is to find $u(x, y)$ such that $\nabla^2 u(x, y) = 0 \in (0, 20) \times (0, 20)$ subject to $u(x, 0) = 0$, $u(x, 10) = 0$, $u(0, y) = 0$, and $u(20, y) = 100$. A cell centre finite volume technique is used to discretise the problem, which is implemented in PHYSICA. Numerical results are obtained using 2-level correction schemes, one based on classical 2-level algorithm and the other based on the parallel multigrid algorithm. The next coarser level for the classical 2-level algorithm is chosen to be $2h$. The coarsest level of the parallel multigrid algorithm is chosen to be $H = 1.25$ in the new parallel algorithm. For demonstration purposes, a V-cycle multigrid iteration is adopted such that the number of iterations on the coarsest level is chosen to be 6 and that on the finest level is chosen to be 3.

By using a 2-level correction scheme with domain decompsotion method, the present parallel multigrid method, it shows similar computational work to the sequential 2-level multigrid method. When the finest mesh becomes increasingly smaller in its mesh, the computational work seems to be smaller as compare to the sequential 2-level multigrid method. An addition property of the present method is that it is intrinsic parallel, in which each subdomain may be computed simultaneously. Table 1 shows a comparison of the total computational work for the sequential and the parallel multigrid. It should be note that as the parallel multigrid is to be run on distributed computing environment, the projected parallel computational work when all the 16 subdomains are computed simultaneously. The dramatic decrease in the timing can be easily observed.

Two-Level Correction Scheme: Comparison of Computational Work.				
Finest mesh	16×16	32×32	64×64	128×128
	$h = 1.25$	$h = 0.625$	$h = 0.3125$	$h = 0.15625$
Sequential 2-Level Multigrid	48	151.5	484.5	1533
Mesh	8×8	16×16	32×32	64×64
Parallel: Sequential run	61.5	120.8	314.5	933.8
Multigrid: Parallel run	10.8	11.2	22	60
Mesh	4×4	4×4	4×4	4×4

Table 1. Comparison of computational work units. 1 Computational work unit is the computational work required to perform 1 iteration on the finest level.

5 An Extension to Multi-Scale Problems

The concept of defect correction is being extended to the problem of sound generation due to fluid motion. This is a multi-scale problem not relating to the size of the subdomains but to the size of the flow variables. The aim here is to

solve the non-linear equation

$$\frac{\partial U}{\partial t} + \mathcal{L}\{U\}U \equiv \frac{\partial(\bar{u}+u)}{\partial t} + \mathcal{L}\{\bar{u}+u\}(\bar{u}+u) := 0 \ , \tag{6}$$

where $\mathcal{L}\{U\}$ is a non-linear operator depending on U. Here $\mathcal{L}\{U\}$ is the Navier-Stokes operator and u is certain noise signal such that $u \ll \bar{u}$. For a 2-D problem,

$$\bar{u} = \begin{bmatrix} \bar{\rho} \\ \bar{v}_1 \\ \bar{v}_2 \end{bmatrix} \qquad u = \begin{bmatrix} \rho \\ v_1 \\ v_2 \end{bmatrix} \ ,$$

where ρ is the density of fluid and v_1 and v_2 are the velocity components along the two spatial axes. Using the summation notation of subscripts, the 2-D Navier-Stokes problem $\frac{\partial u}{\partial t} + \mathcal{L}\{u\}u = 0$ may be written as

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v_j)}{\partial x_j} = 0,$$

$$\frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} + \frac{1}{\rho}\frac{\partial P}{\partial x_i} - \frac{\mu}{\rho}\nabla^2 v_i = 0,$$

Expanding $\frac{\partial(\bar{u}+u)}{\partial t} + \mathcal{L}\{\bar{u}+u\}(\bar{u}+u)$ and re-arranging the resulting terms, one obtains

$$\frac{\partial \rho}{\partial t} + \bar{v}_j \frac{\partial \rho}{\partial x_j} + \bar{\rho}\frac{\partial v_j}{\partial x_j} + [v_j \frac{\partial(\bar{\rho}+\rho)}{\partial x_j} + \rho \frac{\partial(\bar{v}_j + v_j)}{\partial x_j}] = -[\frac{\partial \bar{\rho}}{\partial t} + \bar{v}_j \frac{\partial \bar{\rho}}{\partial x_j} + \bar{\rho}\frac{\partial \bar{v}_j}{\partial x_j}],$$

and

$$\frac{\partial v_i}{\partial t} + \bar{v}_j \frac{\partial v_i}{\partial x_j} + \frac{1}{\bar{\rho}}\frac{\partial P}{\partial x_i} - \frac{\mu}{\bar{\rho}}\nabla^2 v_i \tag{7}$$

$$+[\frac{\rho}{\bar{\rho}}\frac{\partial(\bar{v}_i + v_i)}{\partial t} + (v_j + \frac{\rho}{\bar{\rho}}(\bar{v}_j + v_j))\frac{\partial(\bar{v}_i + v_i)}{\partial x_j}] = -[\frac{\partial \bar{v}_i}{\partial t} + \bar{v}_j \frac{\partial \bar{v}_i}{\partial x_j} + \frac{1}{\bar{\rho}}\frac{\partial \bar{P}}{\partial x_i} - \frac{\mu \nabla^2 \bar{v}_i}{\bar{\rho}}].$$

It can be seen that (6) may be written as

$$\frac{\partial(\bar{u}+u)}{\partial t} + \mathcal{L}\{\bar{u}+u\}(\bar{u}+u) \equiv \frac{\partial \bar{u}}{\partial t} + \mathcal{L}\{\bar{u}\}\bar{u} + \frac{\partial u}{\partial t} + E\{\bar{u}\}u + K[\partial_t, \bar{u}, u], \tag{8}$$

where $\mathcal{L}\{\bar{u}\}$ and $E\{\bar{u}\}$ are operators depending on the knowledge of $\bar{u}$ and $K[\partial_t, \bar{u}, u]$ is a functional depending on the knowledge of both $\bar{u}$ and its derivative and u. Here

$$E\{\bar{u}\}u = \begin{bmatrix} \bar{v}_j \frac{\partial \rho}{\partial x_j} + \bar{\rho}\frac{\partial v_j}{\partial x_j} \\ \bar{v}_j \frac{\partial v_i}{\partial x_j} + \frac{1}{\bar{\rho}}\frac{\partial P}{\partial x_i} - \frac{\mu}{\bar{\rho}}\nabla^2 v_i \end{bmatrix}, \tag{9}$$

$$K[\partial_t, \bar{u}, u] = \begin{bmatrix} v_j \frac{\partial(\bar{\rho}+\rho)}{\partial x_j} + \rho \frac{\partial(\bar{v}_j + v_j)}{\partial x_j} \\ \frac{\rho}{\bar{\rho}}\frac{\partial(\bar{v}_i + v_i)}{\partial t} + (v_j + \frac{\rho}{\bar{\rho}}(\bar{v}_j + v_j))\frac{\partial(\bar{v}_i + v_i)}{\partial x_j} \end{bmatrix}. \tag{10}$$

In order to simulate accurately the approximate solution, $\bar{u}$, to the original problem,

$$\frac{\partial(\bar{u}+u)}{\partial t} + \mathcal{L}\{\bar{u}+u\}(\bar{u}+u) = 0 \ ,$$

Let h denote the mesh size and δ_t be a difference approximation to $\frac{\partial}{\partial t}$ being used in the Reynolds averaged Navier-Stokes solver within PHYSICA. Instead of evaluating $\bar{u}^{(n)}$, one would solve the discretised approximation

$$\delta_t \bar{u}_h^{(n)} + \mathcal{L}_h\{\bar{u}_h^{(n)}\}\bar{u}_h^{(n)} = 0$$

to obtain $\bar{u}_h^*$. The residue, R_h, on the fine mesh h may be computed by using a higher order approximation [8] to $-[\delta_t \bar{u}_h^* + \mathcal{L}_h\{\bar{u}_h^*\}\bar{u}_h^*]$. Let H denote the mesh size for the linearised Euler equations solver, where the linearised Euler operator is given by (9), and is chosen to include as many sound signals with a specified long wavelength as possible. Hence $H \neq 2h$. Again instead of evaluating u, one would solve the discretised approximation

$$\delta_t u_H^{(n)} + E_H\{\bar{u}_H^{(n)}\}u_H^{(n)} = R_H^{(n)}$$

to obtain $u_H^{(n)}$. Here $R_H^{(n)}$ is the projection of R onto the mesh H. Let $I_{\{h,H\}}$ be a restriction operator to restrict the residue computed on the fine mesh h to the coarser mesh H. The restricted residue can then be used in the numerical solutions of linearised Euler equations. Therefore the two-level numerical scheme is (for non-resonance problems):

$n := 0$;
Do $n := n + 1$
  Solve $\delta_t \bar{u}_h^{(n)} + \mathcal{L}_h\{\bar{u}_h^{(n)}\}\bar{u}_h^{(n)} = 0$
  $R_H^{(n)} := -I_{\{h,H\}}[\delta_t \bar{u}_h^* + \mathcal{L}\{\bar{u}_h^*\}\bar{u}_h^*$
  $\bar{u}_H^{(n)} := I_{\{h,H\}}\bar{u}_h^{(n)}$
  Solve $\delta_t \bar{u}_h^{(n)} + E_H\{\bar{u}_H^{(n)}\}u_H^{(n)} = R_H^{(n)}$
  $U_H^{(n)} := \bar{u}_H^{(n)} + u_H^{(n)}$(Corrected results do not need to be used in $u_h^{(n+1)}$.)
Until $n = n_{max}$

Here $U_H^{(n)}$ denotes the discretised approximation of the resultant solution on mesh H. Note that $R_H^{(n)}$ cannot be computed as $\delta_t \bar{u}_h^{(n)} + \mathcal{L}\{\bar{u}_h^{(n)}\}I_{\{h,H\}}\bar{u}_h^{(n)}$ because $\mathcal{L}$ is a non-linear operator. Note also that $\delta_t u_H^{(n)}$ involves a number of smaller time steps, each of $\triangle T$, starting from $u_H^{(n-1)}$ such that $u_H^{(n)}$ defines at the same time level as $\bar{u}_h^{(n)}$.

6 Conclusion

This paper provides some earlier experiments of a parallel multigrid algorithm based on the combination of domain decomposition methods and a coarse level

correction technique. The convergence of a block Jacobi iterative method in the classical Schwarz iterative scheme is greatly accelerated with the idea of a coarse grid correction. Furthermore, it is found that the coarse grid with mesh size H does not require to be the next coarser level of the finest grid. Due to this property, the finest level in different subdomains may be different from each other. This would enable local refinement to be done in an efficient manner and also enable subproblems to be solved in machines located geographically apart. The technique is also extended to handle a multi-scale problem invovling sound signal propagation.

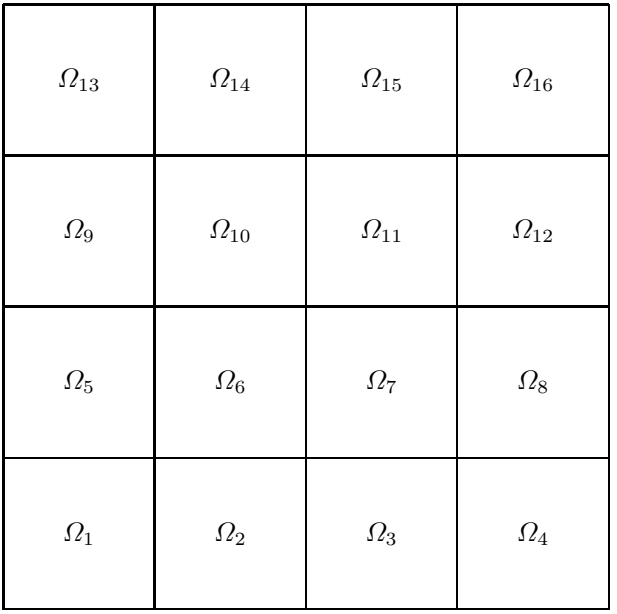

Figure 1: Illustration of a nonoverlapped domain decomposition, which consists of 16 subdomains. Each subdomain is denoted as Ω_i where $i = 1, \ldots, 16$.

References

1. Bachvalov, N.S.: On the Convergence of a Relaxation Method with Natural Constraints on the Elliptic Operator. USSR Comp. Math. and Math. Phys., **6** (1966) 101–135
2. Barkai, D., Brandt, A.: Vectorised Multigrid Poisson solver for the CDC Cyber 205. Applied Mathematics and Computation, **13** (1983) 215–227
3. Brandt, A.: Multi-level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems. Lecture Notes in Physics, **18** (1973) 82-89
4. Brandt, A.: Multi-level adaptive solutions to boundary value problems. Math. Comp., **31** (1977) 333–390
5. Chan, T.F., Schreiber, R.: Parallel Networks for Multi-Grid Algorithms - Architecture and Complexity. SIAM J. Sci. Stat. Comput., **6** (1985) 698–711

6. Croft, T.N., Pericleous, K.A., Cross, M.: PHYSICA: A Multiphysics environment for complex flow processes. Numerical Methods for Laminar and Turbulent Flow (IX/2), C. Taylor et al. (Eds), Pineridge Press, U.K. (1995)
7. Proceedings of International Conference on Domain Decomposition Methods for Science and Engineering. Vol 9, 11, and 12, DDM.Org.
8. Djambazov, G.S.: Numerical Techniques for Computational Aeroacoustics. Ph.D. Thesis, University of Greenwich (1998)
9. Fedorenko, R.P.: The Speed of Convergence of One Iterative Process. USSR Comp. Math. and Math. Phys., **4** (1964) 227–235
10. Frederickson, P.O., McByran, O.A.: Parallel Superconvergent Multigrid. Cornell Theory Centre Report CTC87TR12 (1987)
11. Lai, C.-H.: Non-linear Multigrid Methods for TSP Equations on the ICL DAP. Annual Research Report, Queen Mary, University of London, (1984)
12. Lai, C.-H.: Domain Decomposition Algorithms for Parallel Computers. High Performance Computing in Engineering - Volume 1: Introduction and Algorithms, H. Power and C.A. Brebbia (Eds), Computational Mechanics Publication, Southampton, (1995) 153–188
13. McCormick, S.F. Ruge, J.W.: Unigrid for Multigrid Simulation. Math. Comp., **41** (1983) 43–62
14. Naik, V.K., Ta'asan, S.: Implementation of Multigrid Methods for Solving Navier-Stokes Equations on a Multiprocessor System. ICASE Report 87-37 (1987)
15. PHYSICA on-line Menu: Three-dimensional Unstructured Mesh Multi-physics Computational Mechanics Computational Modellings. http://physica.gre.ac.uk/physica.html (1999)
16. Trottenberg, U., Oosterlee, C., Schuller, A.: Multigrid. Academic Press, New York (2001)
17. Introduction to Multigrid Methods. ICASE Report 95-11 (1995)