MECHANIKA TEORETYCZNA
I STOSOWANA
Journal of Theoretical
and Applied Mechanics
2, 35, 1997

PARALLEL h-ADAPTIVE SIMULATIONS OF INVISCID FLOWS BY THE FINITE ELEMENT METHOD

KRZYSZTOF BANAŚ

JOANNA PŁAŻEK

Section of Applied Mathematics, Cracow University of Technology

e-mail: Krzysztof.Banas@pk.edu.pl

The construction and performance of a parallel algorithm for solving the Euler equations on unstructured grids with dynamically distributed data is presented. The algorithm uses a linear, implicit version of the well known Taylor-Galerkin time marching scheme for time discretization. Finite elements are employed for space discretization of one-step problems and an overlapping domain decomposition algorithm combined with the preconditioned GMRES method is used to solve iteratively in parallel the resulting system of linear equations. A new mesh partition algorithm based on the idea of advancing front is described and tested in practice.
The domain decomposition and the mesh partition methods are combined to form a simple and effective algorithm ensuring the optimal load balance for a multiprocessor system.
A general MIMD multiprocessor (multicomputer) system with distributed memory is assumed as the hardware setting for simulations and Parallel Virtual Machine package is used for message passing. The performance of the method is monitored for a well known transient benchmark problem of 2D inviscid flow simulations – the ramp problem.

Key words: parallel computations, finite element method, compressible fluid flow

1. Introduction

The successful application of Computational Fluid Dynamics in solving real life problems depends strongly on the computational power available, so the progress in computer technology in recent years have always stimulated the development of new computational techniques for making the best use of new hardware environments. Parallel computers form currently the most promising new architecture and many algorithms have been already developed or adapted for concurrent computing. In Computational Fluid Dynamics the simplest to parallelize are explicit methods for integration of the fluid flow equations, where computations are performed locally and parallelization consist mainly in a suitable domain decomposition which should guarantee the smallest intersubdomain boundary and therefore the minimal time spent on interprocessor communication. More difficult are implicit methods where parallelization concerns the solution of the system of linear equations, with the exchange of information across the whole computational domain.

In this paper we present a complete set of procedures for efficient parallel simulation of compressible fluid flows using the implicit adaptive finite element methods. A general multiprocessor system with distributed memory equipped with any software for message passing is assumed as the computational environment for developed programs. Such a setting can be realized on various hardware platforms, ranging from clusters of PC's or workstations to parallel supercomputers.

2. Algorithm for approximation of the Euler equations of inviscid compressible flows

The following variational formulation constitutes the basis for a finite element algorithm to solve inviscid compressible fluid flow problems (see Demkowicz and Banaś (1994) for details):

Find $\boldsymbol{U}^{n+1} \in [H^1(\Omega_C)]^m$ satisfying the suitable Dirichlet boundary conditions and such that for every test function $\boldsymbol{W}$ the following holds

$$\begin{aligned}\int\limits_{\Omega_C} \boldsymbol{W}^\top \boldsymbol{U}^{n+1}\, dV + \Delta t \int\limits_{\Omega_C} \boldsymbol{W}^\top_{,i} (\mathsf{K}^{ij})^n \boldsymbol{U}^{n+1}_{,j}\, dV = \\ = \int\limits_{\Omega_C} \boldsymbol{W}^\top \boldsymbol{U}^n\, dV + \Delta t \int\limits_{\Omega_C} \boldsymbol{W}^\top_{,i} (\boldsymbol{f}^i)^n\, dV - \Delta t \int\limits_{\partial\Omega_C} \boldsymbol{W}^\top (\boldsymbol{f}^i)^n \boldsymbol{n}_i\, dS\end{aligned} \tag{2.1}$$

where

Ω_C	–	computational domain, $\Omega_C \subset \mathbb{R}^l$, $l = 2$ or 3
$\boldsymbol{n}$	–	outward unit vector, normal to the boundary $\partial\Omega_C$

$\boldsymbol{U}$	–	vector of conservation variables $[\rho, \rho u_j, \rho e]^{\top}$, $j = 1,..,l$ (ρ, u_j and e are the density, the jth component of velocity and the specific total energy respectively)
$\boldsymbol{f}^i$	–	Eulerian fluxes, $\boldsymbol{f}^i = [\rho u_i, \rho u_i u_j + p\delta_{ij}, (\rho e + p)u_i]^{\top}$, $(i, j = 1, ..., l)$
p	–	pressure, $p = (\gamma - 1)(\rho e - \frac{1}{2}\rho u_i u_i)$ (γ – the ratio of specific heats, $\gamma = 1.4$ in all examples)
Δt	–	time step length, $\Delta t = t^{n+1} - t^n$
$\mathbf{K}^{ij}$	–	nonlinear matrix functions representing regularization terms and artificial viscosity.

In the notation it is assumed that indices i, j have always their range from 1 to l, that the outer superscripts of functions of $\boldsymbol{U}$ refer to their actual argument (e.g.: $(\mathbf{K}^{ij})^n = \mathbf{K}^{ij}(\boldsymbol{U}^n)$ or $(\boldsymbol{f}^i)^n = \boldsymbol{f}^i(\boldsymbol{U}^n)$), that the summation convention holds and differentiation is denoted by $(\cdot)_{,}$.

A sequence of solutions to the one-step problem (2.1) constitutes a time discrete approximate solution to the Euler equations. Since as regularization matrices we use the products

$$\frac{\Delta t}{2}\boldsymbol{f}^i_{,U}\boldsymbol{f}^j_{,U}$$

the described method belongs to the family of Taylor-Galerkin time marching schemes (cf Donea and Quartapelle (1992)).

The problem (2.1) is discretized in space using triangular finite elements with linear shape functions. Since each one-step computations create a separate problem they may be performed on a different mesh, taking the interpolated result of the previous time step as an initial condition. The strategy of refinements and unrefinements is used to adapt the mesh. As an element refinement or unrefinement indicator the term (cf Eriksson and Johnson (1993))

$$\mathsf{I}_{el} = h^2 \boldsymbol{f}^i_{,i}\mathbf{H}_{,UU}\boldsymbol{f}^j_{,j}$$

based on the residual of the steady state Euler equations $\boldsymbol{f}^i_{,i}$ is employed (h is a characteristic linear dimension of the element). One-irregular meshes are allowed (vertices of refined elements may lie in the middle of a side of bigger elements) and the methodology of constrained approximation (cf Demkowicz et al. (1989)) is applied. The meshes produced by the above procedure are highly nonuniform and unstructered. The existence of constrained nodes complicates even further the data structure used. All that creates important technical difficulties when designing an algorithm for partitioning the mesh into submeshes.

3. The preconditioned GMRES algorithm for solving systems of linear equations

The well known and widely used Generalized Minimal Residual (GMRES) method for solving nonsymmetric systems of linear equations is used to solve the system resulting from the finite element discretization of the problem (2.1). We focus only on several implementation issues connected with preconditioning and parallelization of the algorithm. We consider the original system of equations $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ (here $\mathbf{A}$ stands for the global stiffness matrix and $\boldsymbol{b}$ is the global load vector) and the preconditioned system $\mathbf{M}^{-1}\mathbf{A}\boldsymbol{x} = \mathbf{M}^{-1}\boldsymbol{b}$, where the matrix $\mathbf{M}^{-1}$ will be specified later. The matrix $\mathbf{M}^{-1}$ should be designed in such a way that the preconditioned system has better convergence properties than the original one. The GMRES algorithm, shortly characterized as a method for minimizing the residual for a given problem over the related Krylov space of dimension k, applied to the preconditioned system looks now as follows

```
set an initial guess $\boldsymbol{x}_0$
repeat until attained
  compute the residual of preconditioned system, $\boldsymbol{r}_0 = \mathbf{M}^{-1}(\mathbf{A}\boldsymbol{x}_0 - \boldsymbol{b})$ (*)
  normalize the residual, $\overline{\boldsymbol{r}}_0 = \boldsymbol{r}_0/||\boldsymbol{r}_0||$ (**)
  for $i = 1, 2, ..., k$
    compute matrix-vector product: $\hat{\boldsymbol{r}}_i = \mathbf{M}^{-1}\mathbf{A}\overline{\boldsymbol{r}}_{i-1}$ (*)
    orthonormalize $\hat{\boldsymbol{r}}_i$ with respect to all previous $\overline{\boldsymbol{r}}_j$, $j = 1, .., i-1$ by the
      modified Gram-Schmidt procedure obtaining $\overline{\boldsymbol{r}}_i$ (**)
  solve the GMRES minimization problem
  check convergence, if attained update the solution and leave GMRES
```

Preconditioning of the GMRES algorithm can be efficiently achieved by using any of so called basic iterative methods such as the standard Jacobi, Gauss-Seidel or block Jacobi, block Gauss-Seidel methods (cf Golub and Ortega (1993)). To show this we first decompose the vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ (having dimension N) into nonoverlapping parts $\boldsymbol{x}_i$ and $\boldsymbol{b}_i$, $i = 1, .., N_{bl}$ (N_{bl} – the total number of blocks) which induces the decomposition of the matrix $\mathbf{A}$ into nonoverlapping blocks $\mathbf{A}_{ij}$, $i, j = 1, ..., N_{bl}$ (in the case of standard methods the size of blocks is 1×1).

We denote by $\mathbf{R}_i$ the restriction matrix (of dimension $N_i \times N$, where N_i is the dimension of ith block) which selects $\boldsymbol{x}_i$ from all the components of the vector $\boldsymbol{x}$, $\boldsymbol{x}_i = \mathbf{R}_i\boldsymbol{x}$. Then $\mathbf{R}_i^T$ is the matrix extending $\boldsymbol{x}_i$ to the full dimension. Since at the kth step of any basic iterative method we solve for each $\boldsymbol{x}_i$ the problem

$$\mathsf{A}_{ii}\boldsymbol{x}_i^{k,i} = \boldsymbol{b}_i - \sum_{j\neq i} \mathsf{A}_{ij}\widehat{\boldsymbol{x}}_j^k \tag{3.1}$$

(the choice of $\widehat{\boldsymbol{x}}^k$ depends on the method and is explained below) and additionally

$$\mathsf{R}_i\mathsf{A}\widehat{\boldsymbol{x}}^k := \mathsf{A}_{ii}\widehat{\boldsymbol{x}}_i^k + \sum_{j\neq i} \mathsf{A}_{ij}\widehat{\boldsymbol{x}}_j^k$$

then the operation carried out with the ith block A_{ii} can be written as

$$\boldsymbol{x}^{k,i} = \widehat{\boldsymbol{x}}^k - \mathsf{R}_i^\top \mathsf{A}_{ii}^{-1}\mathsf{R}_i(\mathsf{A}\widehat{\boldsymbol{x}}^k - \boldsymbol{b})$$

($\boldsymbol{x}^{k,i}$ above denotes $\boldsymbol{x}^k$ after performing operations with the ith block). In the block Jacobi algorithm we update $\boldsymbol{x}^k$ after all block operations so $\widehat{\boldsymbol{x}}^k = \boldsymbol{x}^{k-1}$, while in the block Gauss-Seidel method we update it after each block operation and $\widehat{\boldsymbol{x}}^k = \boldsymbol{x}^{k,i-1}$. For both cases the full iteration of the method can be expressed as (cf LeTallec (1994))

$$\boldsymbol{x}^k = \boldsymbol{x}^{k-1} - \mathsf{M}^{-1}(\mathsf{A}\boldsymbol{x}^{k-1} - \boldsymbol{b}) \tag{3.2}$$

where for the block Jacobi method

$$\mathsf{M}^{-1} = \mathsf{M}_J^{-1} := \sum_{i=1}^{N_{bl}} \mathsf{R}_i^\top \mathsf{A}_{ii}^{-1}\mathsf{R}_i$$

and for the block Gauss-Seidel method

$$\mathsf{M}^{-1} = \mathsf{M}_{GS}^{-1} := \left[\mathsf{I} - \prod_{i=1}^{N_{bl}}(\mathsf{I} - \mathsf{R}_i^\top \mathsf{A}_{ii}^{-1}\mathsf{R}_i\mathsf{A})\right]\mathsf{A}^{-1}$$

Thus the two marked with (*) most time consuming steps of the GMRES iteration involving matrix vector products can be performed by means of one of basic iterative methods: the computation of initial residual of the preconditioned system by performing one iteration (3.2) and then subtracting $\boldsymbol{x}^1 - \boldsymbol{x}^0$ and the multiplication $\mathsf{M}^{-1}\mathsf{A}\overline{\boldsymbol{r}}_{i-1}$ by iteration (3.2) with no right-hand side vector $\boldsymbol{b}$ and the vector $\overline{\boldsymbol{r}}_{i-1}$ as an initial guess ($\boldsymbol{x}^0 = \overline{\boldsymbol{r}}_{i-1}$), followed by the subsequent subtraction $\widehat{\boldsymbol{r}}_i = \boldsymbol{x}^0 - \boldsymbol{x}^1$.

4. Overlapping domain decomposition

Overlapping domain decomposition methods for solving partial differential equations can be roughly outlined as the methods where the original problem posed for some domain is solved iteratively: we divide the domain into a number of overlapping subdomains, for each subdomain we solve the original problem setting the Dirichlet boundary conditions on the intersubdomain boundaries and then repeat the procedure until convergence is achieved. Not going into details of these methods (see e.g. Quarteroni and Valli (1994) and citations there) we mention only that in the context of the finite element method basic iterative algorithms can be themselves interpreted as special cases of the overlapping domain decomposition methods with one-element overlap between subdomains. Solving Eq (3.1) corresponds in this interpretation to the solution to the original problem for a small subdomain, composed of all elements sharing the nodes, degrees of freedom of which constitute $\boldsymbol{x}_i$ (the, so called, patch of elements).

5. Parallel implementation of GMRES

In the actual implementation of GMRES into the FEM code we introduce a two level domain decomposition. At the first level subdomains consist of all elements sharing a given common vertex and correspond to the blocks $\mathbf{A}_{ij}$ of the matrix $\mathbf{A}$ with the size of only four entries. This allows one for a quick and efficient solution to problems (3.1) by a direct linear equations solver. The second level decomposition results from the partition of the whole finite element mesh with the number of created subdomains equal to the number of processors (workstations) used. Each big subdomain (from now on called simply subdomain) contains many small subdomains (patches of elements) and has one-element overlap with other subdomains (see a sample mesh in Fig.1). Such construction implies that each node belongs to the interior of only one subdomain and the data structure for a problem is naturally distributed among subdomains (processors).

The sequence of procedures for solving each one step looks as follows. First, a finite element mesh is decomposed into a specified number of subdomains equal to the number of processors. Then the element stiffness matrices and patch matrices are generated in parallel and the system of linear equations is solved by the GMRES without forming explicitly the global stiffness matrix.

Steps marked with (*) in the GMRES diagram are performed using the block Gauss-Seidel algorithm in each subdomain followed by the exchange of data between subdomains. Since the exchange of data takes place after considering all patches of a given subdomain the method becomes a mixed method, subdomain level – Jacobi, block level – Gauss-Seidel. Thus the convergence properties of the parallel method become worse than those of the serial algorithm with the GMRES preconditioned by the block Gauss-Seidel method.

The steps marked with (**) require synchronized interprocessor communication due to such operations as computing scalar products of global vectors (for nodes from all subdomains) or their norms. This fact can influence the efficiency of the algorithm for networks with low data transfer speed.

6. Mesh partition algorithm

At subsequent time steps partitioning of the mesh is performed only after mesh adaptation. For transient problems this means frequent partitions, after each few steps, while for steady state problems only several partitions are required in the whole solution procedure. These changes may imply that different mesh partition algorithms should be used for these two types of problems. The optimal partition of the mesh should lead to the minimal execution time of the whole problem solved by the finite element method. The partition determines two important factors: the convergence properties of an iterative method used to solve the system of linear equations (cf Rachowicz (1995)) and the amount of data exchanged between processors during the solution procedure. The influence of the partition on the execution time depends also upon the multicomputer architecture which determines the speed of transferring the data and the type of the problem solved (as the abovementioned difference between transient and stationary problems).

We have developed an algorithm where the principle of adding the nearest neighbor to the created subdomain is combined with an intermediate step of creating a front (a group) of nodes being the candidates for adding to the subdomain (cf Banaś and Płażek (1996)). Subdomains are created sequentially, each one by starting with some chosen node, by adding to the subdomain all elements to which this node belongs together with all their nodes and then, if certain conditions are satisfied, adding the latter nodes to the front. Next step in the procedure begins with adding to the subdomain a node from the front according to some assigned weight. Then the mechanism of adding elements and nodes to the subdomain as well as adding nodes to the front and chosing consecutive nodes according to the weights continues until the number of nodes in the subdomain does not reach some explicitly specified limit.

The introduction of the front allows one for efficient formation of subdomains with one-element overlap required by the parallel solver described previously. The explicit specification of the limiting number of nodes for a given subdomain is used in a load balancing strategy to create subdomains of different sizes, i.e. with the different number of nodes.

The scheme is fast and due to the presence of weights at nodal points allows one for using different mesh partition strategies. These strategies usually aim at creating the subdomains with desired geometrical features (shape, alignment of intersubdomain boundaries) so the question of optimality of the mesh partition is posed only indirectly.

7. Dynamic load balancing

The domain decomposition and the mesh partition methods are combined into a simple and effective algorithm to ensure the optimal load balance for a multiprocessor system. The idea consist in increasing the efficiency of the whole simulation procedure by minimizing the idle time of processors. Since in the presented GMRES implementation the work load for a given processor is proportional to the number of nodes in the corresponding subdomain the load balance can be achieved by ascribing to each processor a subdomain with the number of nodes proportional to the processor computational power. As an estimate for the processor speed the inverse of the average time for computing patch vectors $\mathbf{A}_{ii}^{-1}\boldsymbol{b}_i$ and matrices $\mathbf{A}_{ii}^{-1}\mathbf{A}_{ij}$ is taken. The information on computer speeds is sent as the input to the procedure which specifies the numbers of nodes for particular subdomains. These numbers form the input for the mesh partition algorithm.

8. Numerical examples

As a test case for showing the performance of the developed parallel algorithm we have chosen a well known transient benchmark problem of inviscid fluid flow – the ramp problem (cf Woodward and Colella (1984)). A Mach 10 shock traveling along a channel and perpendicular to its walls meets at time $t = 0$ s a ramp, making an angle of 60 degrees with the channel walls.

A complicated flow pattern develops with double Mach reflection and a jet of denser fluid along the ramp behind the shock (see Fig.1).

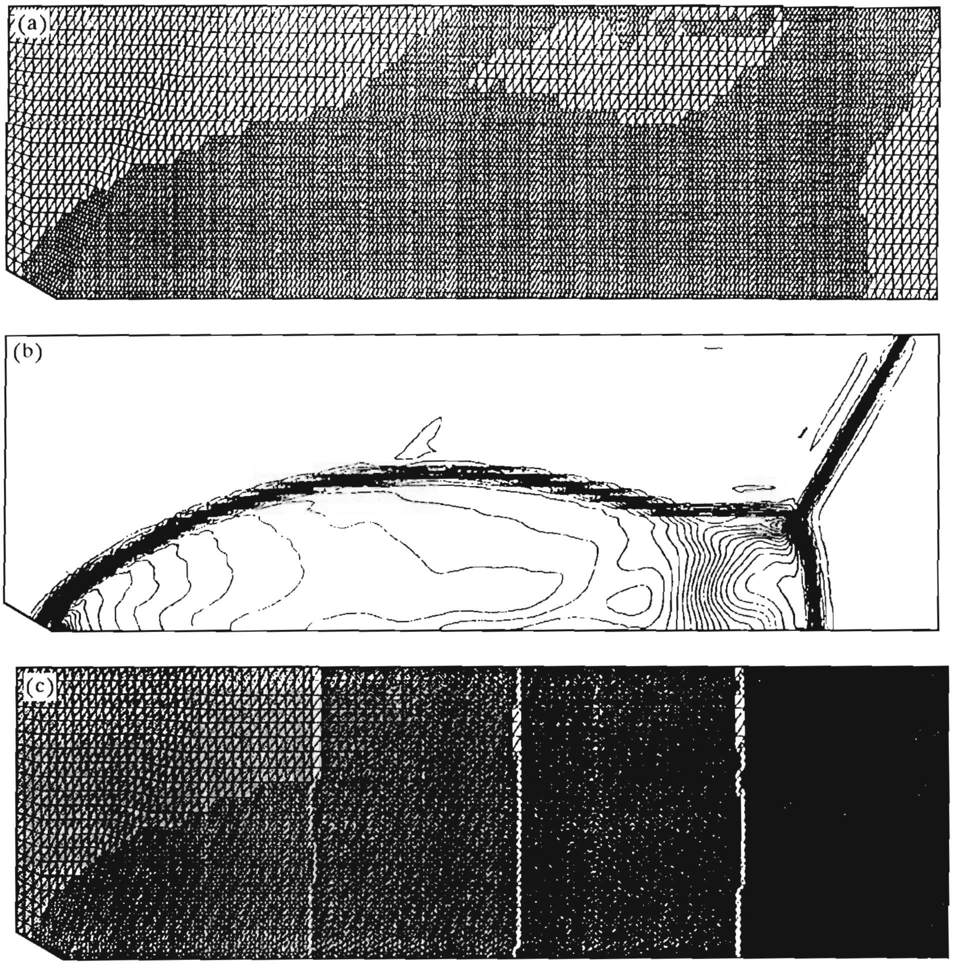

Fig. 1. The ramp problem; (a) – mesh at time $t = 2$s, (b) – solution (density contours with $\Delta\rho = 0.4$), (c) – division of the mesh into 4 subdomains

The results of test runs are presented for four different hardware configurations – 1, 2, 3 or 4 IBM RISC workstations connected by the standard Ethernet network. All figures and tables refer to a sample one-step problem at time $t = 2s$ chosen as a representative of the whole simulation. Fig.1a shows the mesh (12470 nodes), which initially was generated as structured and then refined after each time step. Fig.1b shows the density contours ($\Delta\rho = 0.4$)

and Fig.1c shows the division of the mesh into 4 submeshes. The weights in the mesh partition algorithm were so chosen as to align the intersubdomain boundaries along vertical lines.

Table 1. Performance of the preconditioned GMRES algorithm for the ramp problem – test run for the one-step problem at time $t = 2$s, the mesh with 12470 nodes

Configuration	Part. time [s]	Send. time [s]	Comput. time [s]	Speedup	Efficiency	Residual
1 × RISC IBM	–	–	46.96	1	100%	1.87
2 × RISC IBM	5.34	2.02	25.15	1.87	94%	2.05
3 × RISC IBM	4.44	2.23	17.82	2.64	88%	2.12
4 × RISC IBM	4.16	2.50	14.49	3.24	81%	2.83

The results of test runs are presented in Table 1 which shows times for sequential (performed on the master processor) partition of the mesh, for sending data to processors and for sequential or parallel execution of the example one-step problem. The latter figures include times for the calculation of element stiffness matrices and patch matrices as well as times for the solution of the system of linear equations.

Since, as it was already mentioned, the parallel version of the algorithm reveals the mathematical properties differing from those of the sequential version (resulting in slower convergence rates) we separated two issues – the convergence of the GMRES and the numerical efficiency of parallelization. The times reported in tables always refer to two restarts of the GMRES each with ten base vectors in the Krylov space. Additionally, the L_2 norm of residual after these iterations (divided by 10^{-7}) is quoted for each test run to compare the speed of convergence for different numbers of subdomains (processors).

As a measure of the quality of parallelization two standard quantities are used: the speed-up being the ratio of the sequential execution time to the parallel execution time and the efficiency (percentage) being the ratio of speed-up to the number of processors.

We also present an example of applying the load balancing strategy. For the same test case we divide the computational domain into four submeshes of equal size (i.e. with the same number of nodes). We solve several consecutive one-step problems and use only three computers (so the computations for two subdomains are performed on one computer), thus imitating the case of two fast and two slow computers. Table 2 shows how the load balancing strategy changes the sizes of subdomains in order to equate the times for formation of patch matrices (characteristic times) and how that increases the overall efficiency of the solution procedure. Due to the nonlinearity of the process the acceptable load balance is achieved only after two (in practice this number ranges from one to three) time steps, when the proportion of subdomain sizes stabilizes.

Table 2. Achieving the load balance at three consecutive time steps

Time step	Number of nodes in subdomains	Characteristic times	Computation time
1	1667	16.05	29.49
	1667	15.84	
	1667	8.15	
	1569	7.49	
2	1261	11.55	20.83
	1273	11.65	
	2013	9.38	
	2021	9.41	
3	1151	10.39	19.92
	1157	10.57	
	2126	10.45	
	2129	10.41	

9. Conclusions

We have shown a complete set of procedures for parallel finite element simulations of compressible fluid flow. Starting with the Taylor-Galerkin method for space and time discretization, through the parallel version of the GMRES preconditioned by basic iterative methods, till the mesh partition algorithm based on the idea of creating an advancing front from which the nodes are added to subdomains according to specifically designed weights.

The presented GMRES algorithm allows for efficient parallel solution of nonsymmetric systems of linear equations resulting from implicit finite element space discretizations of problems appearing in Computational Fluid Dynamics. It can be combined with many existing time discretization schemes for fluid flow equations. In described examples it displayed good speed-up and efficiency.

The mesh partition algorithm, thanks to the use of the front of nodes, reveals great flexibility, potentially allowing of creating subdomains with different overlap sizes and optimal shapes. We will report on the further development of the described algorithms in forthcoming papers.

Acknowledgement

The support of this work by the State Committee for Scientific Research (KBN) under grant 7T07A 008 08 is gratefully acknowledged. We also would like to thank the Laboratory of Parallel Computing of the Institute of Computer Science University of Mining and Metallurgy for the computer time.

References

1. DEMKOWICZ L., BANAŚ K., 1994, Entropy Stable Gas Dynamics Simulations by adaptive Finite Elements, in *Proceedings of the Second European Fluid Dynamics Conference ECCOMAS 94*, 5-8 September 1994, Stuttgart, Germany, edit. S.Wagner, E.H.Hirchel, J.Periaux and R.Piva, volume II, 97–104, Chichester, Wiley
2. DEMKOWICZ L., ODEN J.T., RACHOWICZ W., HARDY O., 1989, Towards a Universal h-p Adaptive Finite Element Strategy, Part.1 Constrained Approximation and Data Structure, *Computer Methods in Applied Mechanics and Engineering*, **77**, 79-112
3. DONEA J., QUARTAPELLE L., 1992, An Introduction to Finite Element Methods for Transient Advection Problems, *Computer Methods in Applied Mechanics and Engineering*, **95**, 169-203
4. ERIKSSON K., JOHNSON C., 1993, Adaptive Streamline Diffusion Finite Element Methods for Stationary Convection Diffusion Problems, *Mathematics of Computation*, **60**, 167-188
5. GOLUB G., ORTEGA J.M., 1993, *Scientific Computing with Introduction to Parallel Computing*, Academic Press, San Diego
6. LETALLEC P., 1994, *Domain Decomposition Method in Computational Mechanics*, Computational Mechanics Advances, J.T.Oden edit., North Holland, Amsterdam
7. QUARTERONI A., VALLI A., 1994, *Numerical Approximation of Partial Differential Equations*, Springer-Verlag, Berlin
8. RACHOWICZ W., 1995, An Overlapping Domain Decomposition Preconditioner for an Anisotropic h-Adaptive Finite Element Method, *Computer Methods in Applied Mechanics and Engineering*, **127**, 269-292
9. BANAŚ K., PŁAŻEK J., 1996, Dynamic Load Balancing for the Preconditioned GMRES Solver in a Parallel, Adaptive Finite Element Euler Code, in *Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference*, 9-13 September 1996 Paris, France, edit. J.-A. Désidéri, C.Hirsch, P.Le Tallec, M.Pandolfi, J.Périaux, 1025–1031, Chichester, Wiley
10. WOODWARD P., COLELLA P., 1984, The Numerical Simulation of Two Dimensional Fluid Flow with Strong Shocks, *Journal of Computational Physics*, **54**, 115-173

Współbieżne symulacje przepływów nieściśliwych za pomocą h-adaptacyjnej metody elementów skończonych

Streszczenie

Przedstawiono algorytm do współbieżnej symulacji nielepkich przepływów gazów ściśliwych na niestrukturalnych siatkach i z wykorzystaniem rozproszonej struktury danych. Algorytm wykorzystuje metodę Taylora-Galerkina do aproksymacji czasowej i adaptacyjną metodę elementów skończonych do dyskretyzacji przestrzennej uzyskanych problemów jednokrokowych. Powstający układ równań liniowych jest rozwiązywany metodą GMRES z wykorzystaniem podstawowych metod iteracyjnych do poprawy uwarunkowania macierzy sztywności. Implementacja współbieżna oparta jest o podział obszaru obliczeniowego na podobszary. Zaprezentowano nowy algorytm podziału obszaru (siatki elementów skończonych) wykorzystujący ideę postępującego frontu węzłów.

Algorytm przetestowano na przykładowym problemie interakcji fali uderzeniowej z klinem, uzyskując dobre wyniki przyspieszenia obliczeń dla użytej konfiguracji kilku stacji roboczych połączonych standardową siecią.

Manuscript received October 30, 1996; accepted for print November 29, 1996