INFORMATICA, 2004, Vol. 15, No. 3, 363–378
© 2004 *Institute of Mathematics and Informatics, Vilnius*

Parallel FEM Software for CFD Problems

Arnas KAČENIAUSKAS
Parallel Computing Laboratory, Vilnius Gediminas Technical University
Saulėtekio 11, 2040 Vilius, Lithuania
e-mail: arnka@fm.vtu.lt

Peter RUTSCHMANN
Institute of Hydraulic Engineering, Innsbruck University
Technikerstr. 13, 6020 Innsbruck, Austria
e-mail: peter.rutschmann@uibk.ac.at

Received: January 2004

Abstract. The present paper describes the development and the performance of parallel FEM software for solving various CFD problems. Domain decomposition strategy and parallel iterative GMRES solver have been adapted to the universal space-time FEM code FEMTOOL, which allows implementation of any partial differential equation with minor expenses. The developed data structures, the static load balancing and the inter-processor communication algorithms have been particularly suited for homogeneous distributed memory PC clusters. The universality of the considered parallel algorithms has been validated solving applications described by the Poisson equation, by the convective transport equation and by the Navier–Stokes equations. Three typical benchmark problems have been solved in order to perform the efficiency study. The performance of parallel computations, the speed-up and the efficiency have been measured on three BEOWULF PC clusters as well as on the cluster of IBM RISC workstations and on the IBM SP2 supercomputer.

Key words: parallel computing, domain decomposition, homogeneous Beowulf PC clusters, GMRES solver, finite element method.

1. Introduction

Computational fluid dynamics (CFD) plays an increasingly important role in the design of modern industrial components. Various computational methods are developed for discretising the governing equations and spatial domains. The finite element method (FEM) (Baker, 1983; Zienkiewicz *et al.*, 1991) has the capability of accepting complex geometries in an integrated fashion, making it particularly interesting to the designers of complex mechanisms. However, it has always been the problem of the FEM that larger computational times have been associated with it. Parallel computing is thus perceived as a promising avenue for future advances in this applied area of science.

Domain decomposition (Smith *et al.*, 1996) is the most efficient parallelisation technique used for finite element algorithms. The basic idea of this technique is the partitioning of the computational domain into sub-domains, each being assigned to a processor.

Non-overlapping domain decomposition (Le Tallec *et al.*, 1997) of the grid is generated with processor interfaces coincident with a subset of element surfaces. This implies that adjacent processors share nodes on the sub-domain interfaces. The sub-domains exchange data with each other through their boundaries. If localised data dependency is not destroyed, a solution on the interior elements of a particular sub-domain can be sought entirely in parallel. Then all required data will be local to the processor in control of that specific sub-domain. Inter-processor communication is necessary only when the solution on boundary elements of a sub-domain is sought. Such computation and communication arrangement enables data parallelism to be executed efficiently and is particularly suitable for platforms of distributed memory computers. A parallel computer is used efficiently if load-balancing (Mandel, 1993) is performed and none of the processors have to wait for information they need from other processors. In recent years efficient parallel methods have been introduced. Amongst them, the Shur domain decomposition method (Farhat *et al.*, 1994) proposes a dual approach of the Shur formulation and overlapping Schwarz splitting (Tang, 1992) provides an alternative to non-overlapping domain decomposition methods.

The most time consuming parts of FEM software are the assembling of finite element coefficient matrices and solving large systems of linear equations. The domain decomposition strategy significantly reduces the first time consuming part, because coefficient matrices are assembled locally on every processor. Iterative Krylov subspace methods are believed to be very fast in solving large linear equation systems. The generalised minimal residual solver (GMRES) (Saad *et al.*, 1986) is a sophisticated iterative solver, which cannot break down even for indefinite matrices. This solver employs the Arnoldi process to construct an orthogonal basis of Krylov sub-spaces. In order to avoid excessive storage requirements and computational costs for the orthogonalisation, GMRES is usually restarted after each m iteration steps. This algorithm is referred to as GMRES(m). The parallel implementation of GMRES(m) on distributed memory computers and the performance study is presented in (De Sturler and Van der Vorst, 1995). The basic approach involves parallelisation of the computationally intensive matrix-vector products and dot products of the solver across a partition to sub-domains.

Preconditioning is a very important device for the efficiency and robustness of GMRES solver, particularly, of ill-conditioned matrices. However, despite this, there is little theory available to guide the design of efficient and parallel preconditioners for the various types of matrices encountered in CFD applications. The diagonal (Haroutunian *et al.*, 1993) and block-diagonal (Klawonn, 1998) preconditioners based on the simple iteration of Jacobi type are believed to be the simplest and very well parallelised preconditioners. There are many kinds of other preconditioners proposed in the literature, among which ILU type preconditioners (Meijerink *et al.*, 1977) based on incomplete Cholesky decomposition are most widely used. However, the inherent sequentiality of these preconditioners precludes efficient implementation on distributed memory computers. (De Sturler, 1995; Pavarino, 2000) investigated indefinite overlapping Schwarz splittings combined with local ILUT preconditioning. Several sophisticated preconditioners suitable for non-overlapping domain decomposition strategy have emerged in the last decade: coarse grid preconditioner (Mandel, 1990), wire basket techniques (Smith, 1992) and multilevel preconditioner (Zhang, 2001). However, failure rates of preconditioned iterative solvers are still too high for them to be used as black box library software for solving general sparse matrices of practical CFD interests.

In the present paper, parallel FEM software based on the domain decomposition strategy and iterative GMRES solver is developed and investigated. Mathematical models of investigated problems are outlined in the Section 2. Section 3 introduces FEM formulations and stabilisation techniques applied in this work. Section 4 discusses implementation of parallel algorithms in the general FEM code developed for solving various problems. In Section 5, numerical results are presented and efficiency of the code is discussed. Concluding remarks are included in Section 6.

2. Mathematical and Numerical Models

CFD investigates a wide range of practical problems described by a large set of partial differential equations (PDEs). The main PDEs solved in CFD areas are considered in this work. The Poisson equation is the elliptic differential equation describing potential flows. The convective transport equation governs non-steady sediment transport and thermal processes. The Navier–Stokes equations form the basis for non-linear mathematical models of viscous flows. All above mentioned PDEs discretised by various numerical schemes are implemented in the general FEM software. The solved PDEs and the employed FEM formulation are outlined in the following sub-sections.

2.1. *The Poisson Equation*

The Poisson equation is one of the simplest equations used in CFD areas:

$$\frac{\partial}{\partial x_j}\left(k_j\frac{\partial\phi}{\partial x_j}\right)=q, \tag{1}$$

here, ϕ is an unknown function; k_j is the coefficient and q is a source term. In this paper, all governing equations are provided in a Cartesian reference frame $(x_j,\ j=1,2,3)$ using index notation and usual summation convention.

The standard Galerkin method is applied for the development of the finite element formulation. The state variable ϕ within an element is approximated by shape functions $\boldsymbol{N}^T$ that are equal to weighting functions $\boldsymbol{N}$. The weak form of the weighted residual statement is obtained after applying the Green–Gauss theorem to the left side of (1):

$$-\int_S\frac{\partial\boldsymbol{N}}{\partial x_j}\left(k_j\frac{\partial\boldsymbol{N}^T}{\partial x_j}\right)\mathrm{d}S\cdot\boldsymbol{\Phi}=\int_S\boldsymbol{N}q\,\mathrm{d}S-\int_\Gamma k_j\frac{\partial\phi}{\partial x_j}n_j\,\mathrm{d}\Gamma. \tag{2}$$

The boundary term Γ incorporates Newmann boundary conditions. The Galerkin FEM discretisation (2) of Poisson equation yields a symmetric and positive defined matrix. Such systems of equations are efficiently solved by iterative solvers.

2.2. *The Convective Transport Equation*

Time-dependent problems of convective transport are governed by hyperbolic PDE:

$$\frac{\partial \psi}{\partial t} + u_j \frac{\partial \psi}{\partial x_j} = 0, \tag{3}$$

here ψ is transported variable and u_j is the j-th component of the velocity vector.

The standard Galerkin method yields oscillatory solutions when it is applied to convection dominant problems in conjunction with classical time-stepping algorithms. The Galerkin Least Squares Method (GLS) (Hughes *et al.*, 1989) belongs to the family of the stabilised methods based on adding a stabilisation term to the Galerkin method. This stabilisation term is the least square form of the residual of (3) evaluated elementwise and multiplied by a stabilisation parameter. GLS method is naturally used together with space-time approach (Tezduyar *et al.*, 1992), and the temporal derivatives are treated like the first spatial derivatives:

$$\begin{aligned}&\int_{Q_n} \boldsymbol{N} \left(\frac{\partial \boldsymbol{N}^T}{\partial t} + u_j \frac{\partial \boldsymbol{N}^T}{\partial x_j} \right) \mathrm{d}Q \cdot \Psi \\ &\quad + \sum_{e=1}^{N} \tau_e \int_{Q_n^e} \left(\frac{\partial \boldsymbol{N}}{\partial t} + u_i \frac{\partial \boldsymbol{N}}{\partial x_i} \right) \left(\frac{\partial \boldsymbol{N}^T}{\partial t} + u_j \frac{\partial \boldsymbol{N}^T}{\partial x_j} \right) \mathrm{d}Q \cdot \Psi = 0. \end{aligned} \tag{4}$$

The time derivatives are computed using central weighted space-time finite elements in every space-time slab Q_n. The stabilisation parameter τ_e is calculated in every finite element Q_n^e:

$$\tau_e = \frac{h_e}{2\,|u|_e}; \tag{5}$$

here, h_e is a measure of the finite element length; $|u|_e$ is a length of velocity vector in finite element e $(e = 1, \ldots, N)$. The implicit numerical schema (4–5) is unconditionally stable. The resulting coefficient matrices are unsymmetrical in spite of the symmetry of the stabilising term.

2.3. *The Navier–Stokes Equations*

The laminar and Newtonian flow of viscous and incompressible fluid is described by the Navier–Stokes equations:

$$\rho \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = \rho F_i - \frac{\partial p}{\partial x_i} + \mu \Delta u_i, \tag{6}$$

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{7}$$

where the pressure p and the velocity components u_i are primary variables. ρ is the density, μ is the dynamic viscosity coefficient and F_i are the gravity force components.

In recent years, the space-time GLS finite element method has been developed as a general-purpose computational approach to solve a wide variety of incompressible flow problems (Tezduyar *et al.*, 2000). The corresponding variational formulation employs the time-discontinuous Galerkin method and includes least-squares terms to stabilise the formulation. The stabilisation nature of the formulation prevents numerical oscillation for incompressible flows, using equal-order interpolation functions for velocity and pressure and preserves the consistency of the standard Galerkin method when adapting remeshing is performed.

The variational FEM formulation for the space-time GLS can be written in a matrix form:

$$\Big(\int_{Q_n}\hat{\boldsymbol{N}}\cdot\hat{\boldsymbol{R}}dQ+\sum_{e=1}^{N}\int_{Q_n^e}\hat{\boldsymbol{W}}\cdot\hat{\boldsymbol{R}}\,\mathrm{d}Q\Big)\boldsymbol{X}=\int_{Q_n}\boldsymbol{F}\,\mathrm{d}Q+\sum_{e=1}^{N}\int_{Q_n^e}\hat{\boldsymbol{W}}\cdot\boldsymbol{F}\,\mathrm{d}Q; \quad (8)$$

here, $\boldsymbol{X}$ is the vector of unknown variables in 2D case; $\boldsymbol{F}$ is the right hand side vector:

$$\boldsymbol{X}=\begin{Bmatrix}u_1\\u_2\\p\end{Bmatrix},\quad \boldsymbol{F}=\begin{Bmatrix}\rho F_1\\\rho F_2\\0\end{Bmatrix}; \quad (9)$$

$\hat{\boldsymbol{R}}$ is the residual matrix of the equations (6–7):

$$\hat{\boldsymbol{R}}=\begin{bmatrix}\rho\left(\frac{\partial\boldsymbol{N}^T}{\partial t}+u_j\frac{\partial\boldsymbol{N}^T}{\partial x_j}\right)+\mu\Delta\boldsymbol{N}^T & 0 & \frac{\partial\boldsymbol{N}^T}{\partial x_1}\\ 0 & \rho\left(\frac{\partial\boldsymbol{N}^T}{\partial t}+u_j\frac{\partial\boldsymbol{N}^T}{\partial x_j}\right)+\mu\Delta\boldsymbol{N}^T & \frac{\partial\boldsymbol{N}^T}{\partial x_2}\\ \frac{\partial\boldsymbol{N}^T}{\partial x_1} & \frac{\partial\boldsymbol{N}^T}{\partial x_2} & 0\end{bmatrix}; \quad (10)$$

$\hat{\boldsymbol{N}}$ is the matrix formed by standard Galerkin shape functions:

$$\hat{\boldsymbol{N}}=\begin{bmatrix}\boldsymbol{N}&0&0\\0&\boldsymbol{N}&0\\0&0&\boldsymbol{N}\end{bmatrix}; \quad (11)$$

$\hat{\boldsymbol{W}}$ is the matrix of GLS weighting functions:

$$\hat{\boldsymbol{W}}=\begin{bmatrix}\tau_{SUPG}\left(\frac{\partial\boldsymbol{N}}{\partial t}+u_j\frac{\partial\boldsymbol{N}}{\partial x_j}\right) & 0 & \tau_{CONT}\frac{\partial\boldsymbol{N}}{\partial x_1}\rho\\ 0 & \tau_{SUPG}\left(\frac{\partial\boldsymbol{N}}{\partial t}+u_j\frac{\partial\boldsymbol{N}}{\partial x_j}\right) & \tau_{CONT}\frac{\partial\boldsymbol{N}}{\partial x_2}\rho\\ \tau_{PSPG}\frac{\partial\boldsymbol{N}}{\partial x_1}\frac{1}{\rho} & \tau_{PSPG}\frac{\partial\boldsymbol{N}}{\partial x_2}\frac{1}{\rho} & 0\end{bmatrix}; \quad (12)$$

here, τ_{SUPG}, τ_{CONT} and τ_{PSPG} are stabilisation parameters of non-linear convection terms, pressure terms and continuity equation correspondingly. Tezduyar proposed to use different values for the outlined stabilisation parameters (Tezduyar *et al.*, 2000). In this work,

$$\tau_{SUPG} = \tau_{PSPG} = \tau_{CONT} = \tau_e = \frac{\alpha h_e}{2\left|u\right|_e}, \tag{13}$$

here,

$$\alpha = \coth\left(\frac{2\mu}{\rho\left|u\right|_e h_e}\right) - \frac{\rho\left|u\right|_e h_e}{2\mu}. \tag{14}$$

The Green–Gauss theorem is applied to the Laplasian appearing in the first term $\hat{\boldsymbol{N}}\cdot\hat{\boldsymbol{R}}$ of (8). In the stabilisation terms $\hat{\boldsymbol{W}}\cdot\hat{\boldsymbol{R}}$ of (8), the Laplasian is not evaluated, because the first order shape functions are preferable. The time derivatives in the GLS stabilisation terms can be evaluated or, alternatively, can be omitted in order to save computational resources. The numerical experiments performed in this work showed that the inclusion of time derivatives in $\hat{\boldsymbol{R}}$ and $\hat{\boldsymbol{W}}$ stabilisation matrices makes solution insignificantly more accurate. The absence of the time derivatives in both matrices slightly increases the number of nonlinear iterations (4% per 50 time steps), but significantly improves the convergence of GMRES solver (40% per 50 time steps). The employed stabilisation parameters add symmetric stabilisation matrix including bigger pressure stabilisation terms therefore obtained coefficient matrices are better proceeded by the iterative GMRES solver.

3. Parallel Algorithms

Parallel algorithms were implemented in the FEMTOOL program (Rutschmann, 1995) created in the Swiss Federal Institute of Technology and developed in Innsbruck University and Vilnius Gediminas Technical University. FEMTOOL is a Finite Element Method Toolbox, which allows implementation of any partial differential equation with minor expenses. Time dependent problems are solved using space-time finite elements. The order of shape functions is determined by input and is limited neither in space nor in time. A given transient problem can be solved in several implicit time steps from one time level to the other or in one single implicit step for all time levels. Space-time finite element integration in time and the high order shape functions generated automatically make FEMTOOL applicable to various CFD problems.

Parallelisation of the FEMTOOL is based on the domain decomposition strategy. Finite element meshes are partitioned into sub-domains by METIS or ParMETIS libraries (Karypis *et al.*, 1998). The multilevel graph partitioning schemes and parallel multilevel k-way graph partitioning algorithms are employed for creating finite element mesh partitions of high quality (Fig. 1). Partitions of roughly equal size ensure the static load balancing on the homogeneous parallel machines. The fill-reducing orderings of sparse matrices computed by METIS significantly reduce the amount of communication in parallel sparse matrix-vector multiplication. The node-based nested dissection algorithms

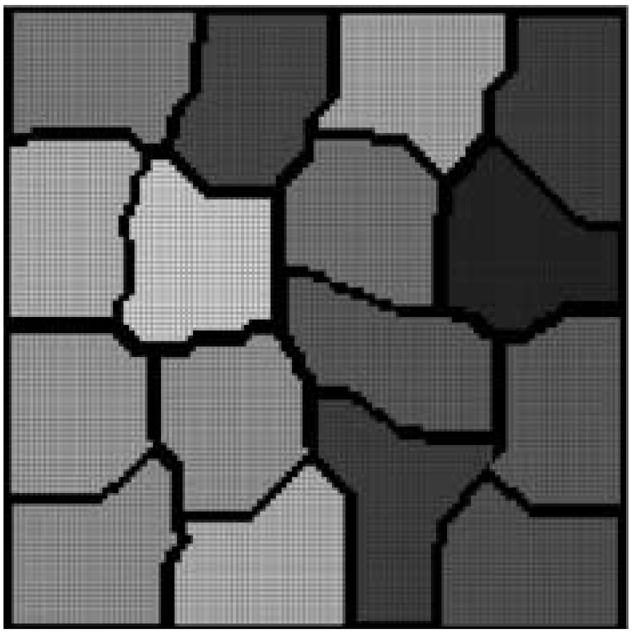

Fig. 1. The partitioned finite element mesh (10000 elements, 16 sub-domains).

outperform other popular reordering algorithms. METIS routines are incorporated in the pre-processor of FEMTOOL.

Parallel sparse matrix-vector operations need information about the degrees of freedom processed on neighbouring sub-domains. Every processor creates local arrays for communication between neighbouring processors. Fig. 2 shows data structure used for inter-processor communication. The data structure illustrates how the processor 1 addresses and stores the local data necessary for neighbouring processors 0, 2 and 3. Local variable nr_of_nb stores the number of neighbouring processors (sub-domains) for a considered processor. The array ***nb2proc***(nr_of_nb) contains identification numbers of neighbouring processors. The variable tot_bnd_var stores a local number of degrees of freedom that are also processed on the other processors. Only these unknowns are needed for inter-processor communication in parallel FEM codes. The array ***nb2var_ptr***$(nr\_of\_nb + 1)$ contains the pointers to the array ***nb2var***(tot_bnd_var).

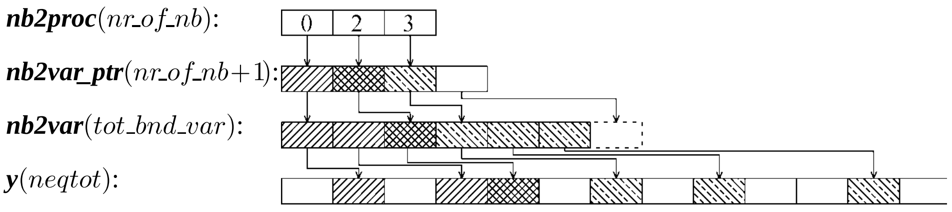

Fig. 2. Data structures for inter-processor communication.

These pointers are used for convenient access of data belonging to any processor. The last cell in array ***nb2var_ptr*** is used for identifying the end of the array ***nb2var***. The main integer array ***nb2var***(tot_bnd_var) defines the position of a considered variable in assembled coefficient matrix or vector of unknowns $\boldsymbol{y}(neqtot)$. The variable *neqtot* stores the local number of degrees of freedom. The algorithm in Fig. 3 shows how the necessary data is addressed in the local vector $\boldsymbol{y}$ and is sent to the neighbouring processors. The universal MPI routine MPI_SEND might be changed by more efficient and specific routines like non-blocking MPI_ISEND, depending on the particular algorithm. In the presented algorithm, the consistency of local and global ordering of the nodes ensures that values received from the neighbouring processor can be directly included into the assembling matrix or vector using addresses contained in ***nb2var***. Subroutines of matrix-vector operations use the described integer data arrays for processing double precision arrays ***invar***(tot_bnd_var) and ***outvar***(tot_bnd_var). MPI routines send (receive) information stored in ***outvar***(***invar***) for the neighbouring processors. The variable *proc* stores the global number of the neighbouring processor, while the variables *first* and *last* are used for convenient addressing of the array ***nb2var*** in the loop and help to estimate the *count* of sent data. A described algorithm performs inter-processor communication required for the matrix and vector operations in the GMRES solver.

The GMRES (Saad *et al.*, 1986) is a popular method for iterative solution of linear equation systems with non-symmetric coefficient matrices. The GMRES(m) solver (De Sturler and Van der Vorst, 1995) is based on the matrix and vector operations, therefore, it was parallelised using the algorithm described above (Figs. 2–3). The iteration number of the GMRES(m) solver depends on a preconditioner. The simple diagonal preconditioner (Haroutunian *et al.*, 1993) is implemented in the software. The lack of robustness and universality of the diagonal preconditioner is compensated by the trivial parallel implementation and the negligible amount of inter-processor communication. The need for better parallel preconditioner which is robust and dramatically reduce the number of iterations while not significantly degrading scalability and parallel performance is a key open problem for the most PDEs applications. The most appropriate preconditioning strategies based on the approximate Shur complement (Farhat *et al.*, 1994), overlapped Schwarcz splitting (De Sturler, 1995; Pavarino, 2000) or multilevel approaches (Zhang, 2001) require significant changes in the data structures used in the standard FEM software. The local ILUT preconditioning (Saad *et al.*, 1999) combined with non-overlapped domain decomposition is implemented and tested in this work. The sequential version of the ILUT preconditioner is very efficient solving the Navier–Stokes equations, but the parallel implementation on the distributed memory computers is very complex. Slightly overlapping sub-domains combined with local preconditioning produce large increase of the iteration number and huge inter-processor communication overhead. This unsuccessful attempt only confirms the fact, that design of universal and efficient parallel preconditioner for ill-conditioned matrices reveals a great challenge.

```
do $nb = 1,$ *nr_of_nb*
 $proc =$ ***nb2proc***$(nb)$
 $first =$ ***nb2var_ptr***$(nb)$
 $last =$ ***nb2var_ptr***$(nb+1) - 1$
 do $i = first, last$
         ***outvar***$(i) = \boldsymbol{y}(\boldsymbol{nb2var}(i))$
 end do
 $count = last - first + 1$
 call MPI_SEND(***outvar***$(first), count,$ MPI_DOUBLE_PRECISION,
                     $proc, mat\_exch\_tag, comm, ierror)$
end do
```

Fig. 3. The algorithm for inter-processor communication.

4. Numerical Results

The parallel FEMTOOL software presented in this paper has been tested on several parallel architectures. The main tests have been performed on three BEOWULF PC clusters: AMD Athlon BEOWULF cluster (OS Linux, 6 processors, 600 MHz, 256 MB RAM for a processor, 100 Mbit/s network), PC cluster VILKAS (OS Linux, 20 Intel Pentium III processors, 1.4 GHz, 0.5 GB RAM for a processor, 1 Gbit/s network) and Transtec PC cluster (OS Linux, 32 Intel Xeon processors, 2.2 GHz, 0.5 GB RAM for a processor, 1 Gbit/s network). Each node of clusters is equipped with two processors. Nodes are interconnected with a hub running 100 Mbit or 1 Gbit Ethernet. The performance achieved on the PC clusters is compared with that obtained on the cluster of IBM RISC workstations (OS AIX, 6 processors, 166 MHz, 96 MB RAM for a processor, 100 Mbit/s network) and IBM SP2 supercomputer (OS AIX, 4 processors, 120 MHz, 128 MB RAM for a processor, 360 Mbit/s network). The parallel performance of the developed code is judged by measurements of speed-up S_p and the efficiency E_p:

$$S_p = \frac{t_1}{t_p}, \quad E_p = \frac{S_p}{p}; \tag{15}$$

here, t_1 is the program execution time for a single processor; t_p is the wall clock time for a given job to execute on p processors. Parallel efficiency is measured by fixing the problem size and increasing the number of processors used. In practice, perfect efficiency is not naturally attained because of an inherent serial part of the algorithm, parallel communication overhead and load imbalance.

To demonstrate the capability of the implemented parallel algorithms to handle various CFD applications, three typical benchmark problems have been considered. These test examples are described by different PDEs. Various numerical schemes have been employed and very different numerical models have been obtained solving the considered test problems in order to test a wide range of actual CFD applications. The first benchmark problem includes exact solution simulation of the simplified Poisson equation $(k_j = 1,\ q = 0)$ or the Laplace equation. The computational domain is the unit square $[0.0; 1.0] \times [0.0; 1.0]$ discretised by three meshes of 40000, 90000 and 160000 bilinear finite elements (39999, 89999 and 159999 degrees of freedom, respectively). The Dirichlet boundary conditions are prescribed on the boundaries $y = 0.0,\ \varphi = -10.0$ and $y = 1.0,\ \varphi = 10.0$. The zero Newmann boundary conditions are prescribed on the boundaries $x = 0.0$ and $x = 1.0$. The analytical solution of a considered problem linearly varies from $\varphi = -10.0$ to $\varphi = 10.0$ in the y direction.

All test runs of the Laplace equation have been performed on the AMD Athlon BEOWULF cluster and the cluster of IBM RISC workstations. The program execution time, the speed-up gained relative to a serial run as a function of the number of processors and the parallel efficiency are shown in Fig. 4. The unexpected position of efficiency curves (90000 and 160000 models) on experimental AMD Athlon cluster shows that parallel processes are better managed on IBM RISC cluster. The AMD Athlon processors are newer and faster, therefore, the program execution time is significantly shorter. When the number of processors is small, the speed-up is close to linear. The reduction of the efficiency owing to communication overhead is obtained for a larger number of processors. The communication cost is quite small for relatively large problems (meshes of 90000 and 160000 elements), where a large number of finite elements are used per processor. In case of benchmark problem with 160000 finite elements, even the super-linear speed-up was obtained on the IBM RISC cluster. It is caused by occurred advantageous cashing and by the lack of RAM on a single workstation.

The rotating cone problem is chosen as the second benchmark problem widely used to illustrate the effectiveness of the algorithms in case of convection dominated flows. The 2D square solution domain $[-0.5; 0.5] \times [-0.5; 0.5]$ is discretised by different structural finite element meshes of 10000, 40000, 90000 and 160000 finite elements (10201, 40401, 90601 and 160801 degrees of freedom, respectively). The concentration cone of radius 0.15 is positioned at ($-0.25;\ 0.0$). The concentration maximum equals 1.0 in the centre of the cone and decreases to zero as a sinusoidal curve. The velocity field $u = -y,\ v = x$ corresponds to a rotational flow with a nature of a solid body. The problem is numerically difficult to solve not only because of the pure advection but also because of the numerical diffusion attributed to the Cartesian grid discretising the rotational flow field.

The performed investigation has shown that the space-time GLS method achieves very accurate results in the wide range of Courant numbers. The accuracy and the efficiency of the developed implementation of the GLS method significantly outperform those of other investigated methods (Kačeniauskas, 2003). A position of the rotating cone after 2π period of time is shown in Fig. 5. The parallel tests have been carried out on the Transtec PC cluster in Innsbruck, PC cluster VILKAS and IBM SP2 supercomputer in the Laboratory of Parallel Computing in Vilnius Gediminas Technical University. Fig. 6 shows the results of the speed-up study on the PC clusters. The processors of Transtec PC cluster are faster, but the speed of the network is the same as that of the cluster VILKAS. This is the reason why Transtec PC cluster solves the small problem of 10000 elements with lower efficiency. The implemented domain decomposition strategy and parallel GMRES solver are well designed for solving convective transport problems. Fig. 7 illustrates the superiority of the speed-up achieved solving the convection transport equation in com-

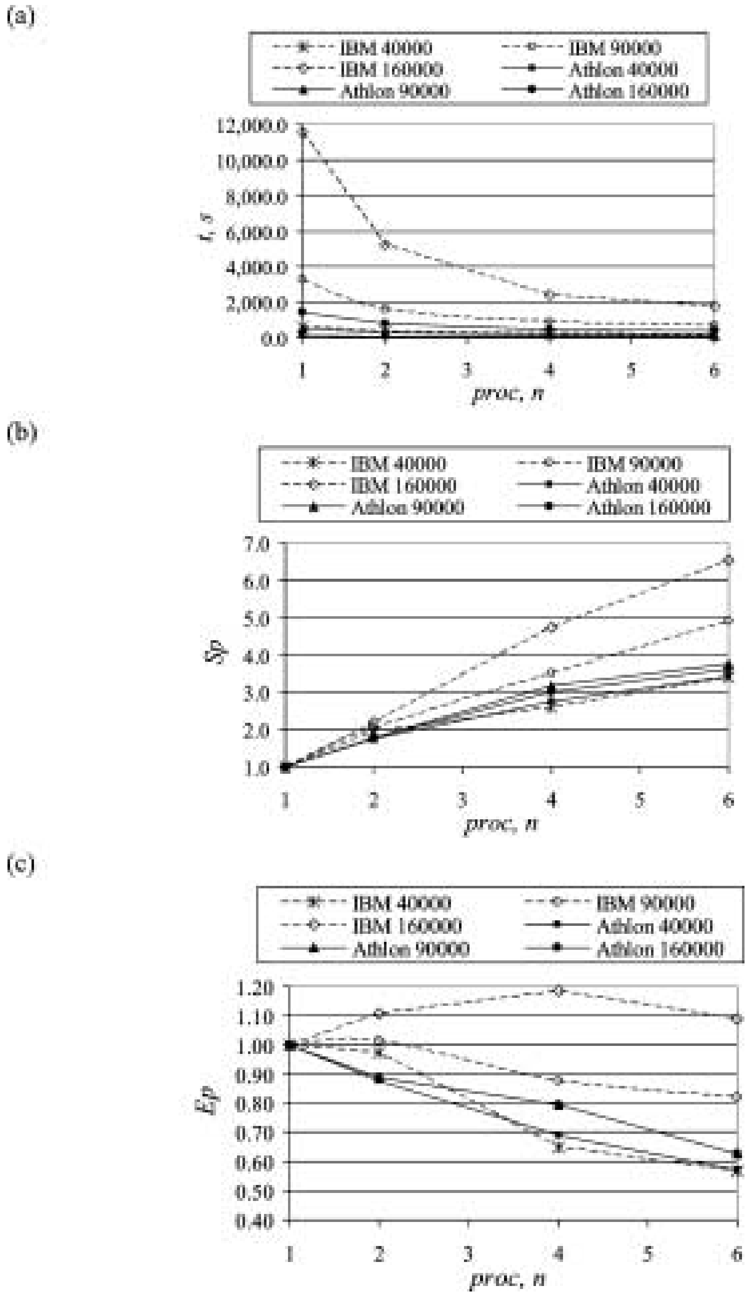

Fig. 4. Performance tests solving the Laplace equation on the AMD Athlon BEOWULF cluster and the cluster of IBM RISC workstations: (a) the run time, (b) the speed-up, (c) the efficiency.

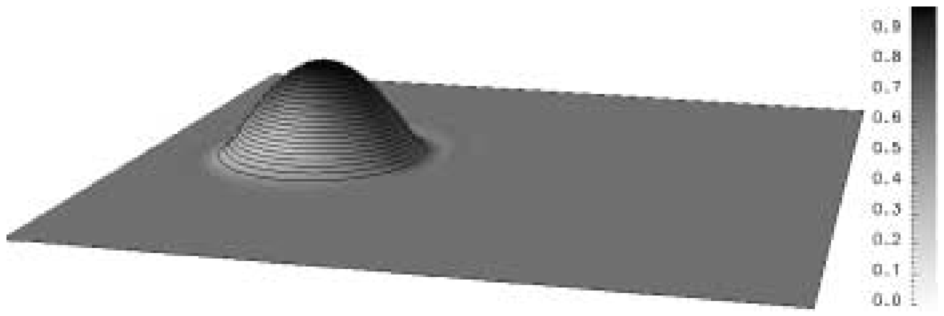

Fig. 5. The position of the rotating cone after 2π period of time.

parison with that obtained by solving the problems described by the coupled formulation of the Navier–Stokes equations.

The fully developed laminar viscous flow in a 2D rectangular channel has been solved as the third test problem. The considered flow is described by the Navier–Stokes equations using coupled formulation and the space-time GLS method (8–14). The solution domain $[0.0; 3.0] \times [0.0; 1.0]$ is discretised by finite element meshes of 3300, 13200, 22000 and 30000 elements (9800, 39400, 65750 and 89700 degrees of freedom, respectively). The no-slip boundary conditions for the velocity are prescribed on the walls of the channel. The parabolic velocity profile is defined at the inflow and zero pressure is fixed at the outflow. The influence of the gravity force is neglected. For this example, the exact analytical solution for the axial velocity is parabolic throughout the domain. The axial pressure gradient is constant and given in the reference (Lewis *et al.*, 1995).

The efficiency tests carried out on the IBM SP2 supercomputer and the PC cluster VILKAS are presented in Figs. 7 and 8. The curves show that the advantages obtained by parallelism diminish rapidly as the number of processors exceeds some threshold value. This behaviour is explained by the fact that the parallel efficiency is largely determined by the ratio of local computations over interprocessor communication. As the number of processors increases, for a fixed problem size, the communication cost will eventually become dominant over the local computation cost after a certain stage. This high ratio of communication to computation makes the influence of a further reduction in the local computation very small on the overall cost of running the application. It is evident that the discussed threshold value for solving the coupled Navier–Stokes equations is smaller than that of other problems. It can be explained by the fact that the resulting finite element coefficient matrices are not well proceeded by iterative solvers and GMRES solver needs a large number of iterations to converge. Thus communication overhead occurred solving linear equation systems increases and the solution time becomes long in comparison with the time used for parallel and independent assembling of finite element coefficient matrices. It dramatically reduces the efficiency measured on a fixed size problem. The last benchmark problem has been the most complicated one. IBM SP2 supercomputer has managed to solve this problem in fast and efficient way despite its slow processors.

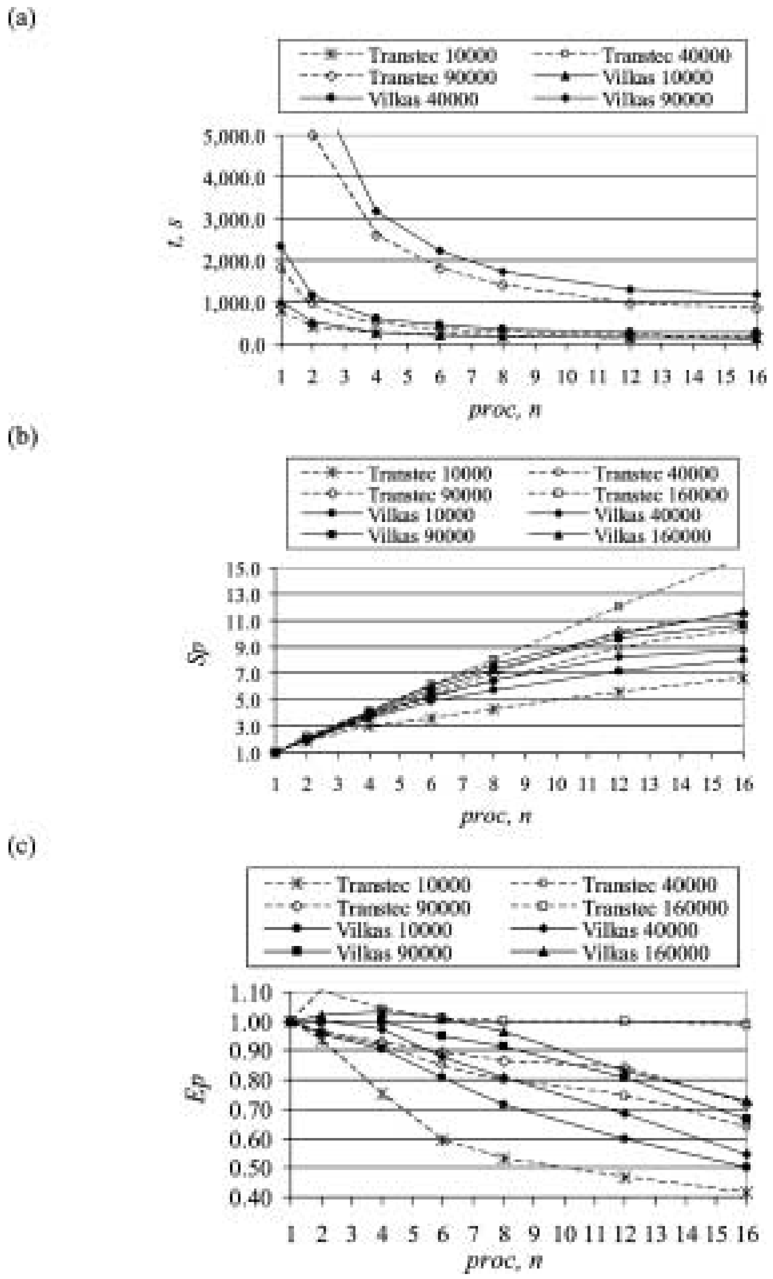

Fig. 6. Performance tests solving the convection transport equation on the Transtec PC cluster and the PC cluster VILKAS: (a) the run time, (b) the speed-up, (c) the efficiency.

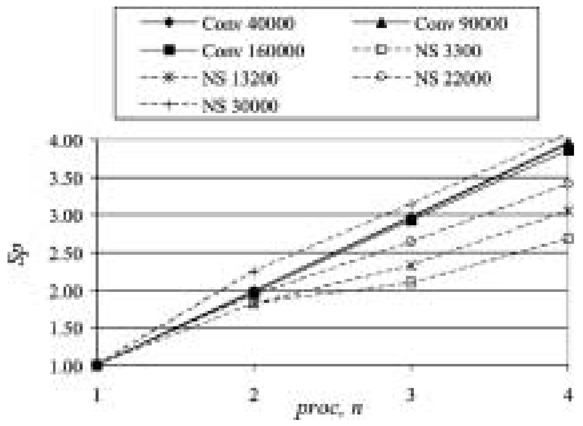

Fig. 7. Speed-up tests solving the convection transport equation and the Navier–Stokes equations on the IBM SP2 supercomputer.

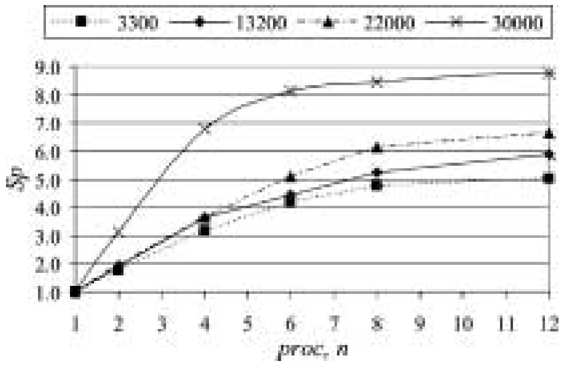

Fig. 8. Speed-up tests solving the Navier–Stokes equations on the PC cluster VILKAS.

5. Conclusions

In this paper, the development of parallel FEM software based on the domain decomposition strategy and the iterative GMRES solver has been described. The parallel and universal FEMTOOL code has been sought to be applicable to various CFD problems of interest. The universal domain decomposition strategy has been successfully applied to all types of considered boundary value problems in spite of different numerical models used. The static load balancing on the homogeneous parallel machines has been ensured by mesh partitioning code METIS incorporated in the pre-processor of the software. The iterative GMRES solver has been successfully parallelised developing the data structures and the communication algorithm particularly suited for distributed memory computers. The solution of three benchmark problems has illustrated high efficiency of the parallel algorithms. The performed speed-up analysis has shown that the best results are achieved solving convective transport problem. This has occurred due to the long time necessary for assembling GLS finite element coefficient matrices. The implemented domain decomposition has perfectly parallelised this part of computations without any inter-processor communications. The solution of non-linear equation systems requires extensive communications among processors, which automatically decrease the desirable speed-up. The favourable time ratio of assembling finite element coefficient matrices to solving linear equation systems has allowed the achieving of the best efficiency on the distributed memory BEOWULF clusters.

Acknowledgement

The authors would like to thank Prof. Eric De Sturler for helpful suggestions and useful discussions related with some important topics of this work.

References

Baker, A.J. (1983). *Finite Element Computational Fluid Mechanics*. McGraw–Hill, New–York.

Farhat, C., and F.X. Roux (1994). Implicit parallel processing in structural mechanics. *Computational Mechanics Advances*, **2**(1), 15–26.

Haroutunian, V., M.S. Engelman and I. Hasbani (1993). Segregated finite element algorithms for the numerical solution of large-scale incompressible flow problems. *Int.J.Numerical Methods in Engineering*,**17**(2), 323–348.

Hughes, T.J.R., L.P. Franca, and G.M. Hulbert (1989). A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations. *Computer Methods in Applied Mechanics and Engineering*, **73**(2), 173–189.

Karypis, G., and V. Kumar (1998). A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *J. Parallel and Distributed Computing*, **48**(1), 71–95.

Kačeniauskas, A. (2003). Parallel solution of convective transport equation by GLS stabilised space-time FEM. *Mechanika*, **4**(42), 61–66.

Klawonn, A. (1998). Block-triangular preconditioners for saddle point problems with a penalty term. *SIAM J. Scientific Computing*, **19**(1), 172–184.

Lewis, R.W., K. Ravindran and A.S. Usmani (1995). Finite element solution of incompressible flows using an explicit segregated approach. *Archives of Computational Methods in Engineering*, **2**(4), 69–93.

Mandel, J. (1990). Two level domain decomposition preconditioning for the p-version finite element method in three dimensions. *Int. J. Numerical Methods in Engineering*, **29**(4), 1095–1108.

Mandel, J. (1993). Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, **9**(2), 233–241.

Meijerink, J.A., and H.A. Van der Vorst (1977). An iterative solution method for linear equation systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computations*, **31**(2), 148–162.

Pavarino, L.F. (2000). Indefinite overlapping Schwarz methods for time-dependent stokes problems. *Computer Methods in Applied Mechanics and Engineering*, **187**(1–2), 35–51.

Rutschmann, P. (1995). *FE Solver with 4D Finite Elements in Space and Time*. Report, Swiss Federal Institute of Technology.

Saad, Y., and M. Schultz (1986). GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Scientific Computing*, **7**(3–4), 856–859.

Saad, Y., and J. Zhang (1999). BILUTM: a domain-based multilevel block ILUT preconditioner for general sparse matrices. *SIAM J. Matrix Analysis in Applications*, **21**(1), 279–299.

Smith, B. (1992). An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems. *SIAM J. Scientific Computing*, **13**(2), 364–378.

Smith, B., P. Bjørstad and W. Gropp (1996). *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, Cambridge.

De Sturler, E. (1995). Incomplete block LU preconditioners on slightly overlapping subdomains for a massively parallel computer. *Applied Numerical Mathematics*, **19**(1–2), 129–146.

De Sturler, E., and H.A. Van der Vorst (1995). Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. *Applied Numerical Mathematics*, **18**(4), 441–459.

Le Tallec, P., and A. Patra (1997). Non-overlapping domain decomposition methods for adaptive *hp* approximations of the Stokes problem with discontinuous pressure fields. *Computer Methods in Applied Mechanics and Engineering*, **145**(3–4), 361–379.

Tang, W.P. (1992). Generalized Schwarz splittings. *SIAM J. Scientific Computing*, **13**(4), 573–595.

Tezduyar, T.E., and S. Aliabadi (2000). EDICT for 3D computation of two-fluid interfaces. *Computer Methods in Applied Mechanics and Engineering*, **190**(3–4), 403–410.

Tezduyar, T.E., J. Liou and M. Behr (1992). A new strategy for finite element computations involving moving boundaries and interfaces – the DSD/ST procedure: I. The concept and the preliminary numerical tests. *Computer Methods in Applied Mechanics and Engineering*, **94**(3), 339–351.

Zhang, J. (2001). A multilevel dual reordering strategy for robust incomplete LU factorization of indefinite matrices. *SIAM J. Matrix Analysis in Applications*, **22**(3), 925–947.

Zienkiewicz, O.C., and R.L. Taylor (1991). *The Finite Element Method*, Vol. 1–2, 4th ed., McGraw–Hill, London.

A. Kačeniauskas graduated in applied mathematics from the Vilnius University, Lithuania. In 1996 he received the MS degree in computer science from the Vilnius Gediminas Technical University (VGTU). In 2000 he defended his doctor thesis "Modelling of viscous incompressible flows and free surfaces by the finite element method" and received PhD degree from the VGTU. A. Kačeniauskas currently works as a principal researcher in the Parallel Computing Laboratory of VGTU. His research interests include parallel computing, computational fluid dynamics, the finite element method, the Navier–Stokes equations, free surfaces and moving interfaces.

P. Rutschmann graduated in 1979 in civil engineering at the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland. In 1988 he received his PhD degree from ETH Zurich with an experimental investigation on high-speed two phase flows. He then became chief-assistant at the chair of hydraulic engineering at ETH Zurich. Since 2002 he started working as a full professor in the Innsbruck University, Austria. Prof P. Rutschmann now is head of the Institute of Hydraulic Engineering at Innsbruck University. His research interests include hydraulics, hydrology, computational fluid dynamics and the space-time finite element method.

Lygiagreti baigtinių elementų metodo programinė įranga skaičiuojamosios skysčių dinamikos uždaviniams spręsti

Arnas KAČENIAUSKAS, Peter RUTSCHMANN

Straipsnyje nagrinėjamas lygiagrečios BEM programinės įrangos įvairiems skaičiuojamosios skysčių dinamikos uždaviniams spręsti kūrimas ir tobulinimas. Srities padalinimo koncepcija ir lygiagretus iteracinis tiesinių lygčių sistemų sprendėjas GMRES pritaikyti universaliam erdvės ir laiko BEM programų paketui FEMTOOL, kuriame diferencialinės lygtys su dalinėmis išvestinėmis realizuojamos minimaliomis pastangomis. Sukurtos duomenų struktūros, statinis apkrovos balansas ir komunikacijų tarp procesorių algoritmai puikiai tinka homogeniniams paskirstytos atminties PK klasteriams. Nagrinėjamų lygiagrečiųjų algoritmų universalumas patikrintas sprendžiant uždavinius, nusakomus Puasono, konvekcinio transporto bei Navje ir Stokso lygtimis. Atliekant algoritmų efektyvumo analizę buvo išspręsti trys testiniai uždaviniai. Lygiagrečiųjų skaičiavimų pagreitėjimas ir efektyvumas buvo išmatuoti trijuose PK klasteriuose, IBM RISC darbo stočių klasteryje bei IBM SP2 superkompiuteryje.