PARALLEL INEXACT NEWTON-TYPE METHODS FOR THE SUPG/PSPG SOLUTION OF STEADY INCOMPRESSIBLE 3D NAVIER-STOKES EQUATIONS IN PC CLUSTERS

Renato N. Elias
Alvaro L. G. A. Coutinho
Marcos A. D. Martins
Rubens M. Sydenstricker
Center for Parallel Computations and Department of Civil Engineering
Federal University of Rio de Janeiro, P. O. Box 68506,
RJ 21945-970 – Rio de Janeiro, Brazil
{renato, alvaro, marcos, rubens}@nacad.ufrj.br

Abstract. The finite element discretization of the three-dimensional incompressible Navier-Stokes yields a non-linear problem due the convective terms in the momentum equation. In this work we describe a parallel MPI-based implementation for PC clusters of inexact Newton-type schemes associated to the SUPG/PSPG stabilized finite element formulation of steady incompressible flows. The resulting linear systems of equations are solved iteratively by an element-by-element preconditioned GMRES solver. Performance of the parallel iterative solver and the nonlinear strategy is evaluated by numerical tests in a benchmark problem.

Keywords. Parallel finite elements, Inexact Newton methods, Navier-Stokes equations.

1 INTRODUCTION

The finite element computation of incompressible Newtonian flows involves two sources of potential numerical instabilities associated with the Galerkin formulation of the problem. One source is due to the presence of convective terms in the governing equations. The other source is due to using inappropriate combinations of interpolation functions to represent the velocity and pressure fields. These instabilities are frequently prevented by addition of stabilization terms into the Galerkin formulation.

In this work we consider the stabilized finite element formulation proposed by Tezduyar (1991) applied to solve steady Newtonian incompressible flows. This formulation allows that equal-order-interpolation velocity-pressure elements are employed, circumventing the Babuska-Brezzi stability condition by introducing two stabilization terms. The first term is the Streamline Upwind Petrov-Galerkin (SUPG) introduced by Brooks and Hughes (1982) and the other one is the Pressure Stabilizing

Petrov Galerkin (PSPG) stabilization proposed initially by Hughes *et al* (1986) for Stokes flows and generalized by Tezduyar *et al* (1992) to high Reynolds number flows.

The discretization of the incompressible Navier-Stokes equations gives rise to a system of nonlinear algebraic equations due the presence of convective terms in the momentum equation. Among several strategies to solve nonlinear problems the Newton's methods is attractive because it converges rapidly from any sufficient good initial guess (Dembo *et al.*, 1982). However, the implementation of Newton's method involves some considerations: Newton's method requires the solution of linear systems at each stage and exact solutions can be too expensive if the number of unknowns is large. In addition, the computational effort spent to find exact solutions may not be justified when the nonlinear iterates are far from the solution. Therefore, it seems reasonable to use an iterative method, such GMRES, to solve these linear systems only approximately.

The inexact-Newton method associated with iterative Krylov solvers have been used to reduce computational efforts related to non-linearities in many problems of computational fluid dynamics, offering a trade-off between the accuracy and the amount of computational effort spent per iteration. According to Kelley (1995) its success depends on some factors, such as: quality of initial Newton step, robustness of Jacobian evaluation and proper forcing function choice.

Many authors have proposed combinations between stabilized finite element formulations and algorithms to solve the nonlinear problems arising from non-Newtonian incompressible flow simulations. Some of these strategies employ analytical or directional forms of Jacobians in the Newton method. The analytical derivative of the stabilization terms are often difficult to evaluate. In this work we have tested the performance of the approximate Jacobian form described by Tezduyar (1999). This numerically approximated Jacobian is based in Taylor's expansions of the nonlinear terms and presents an alternative and simple way to implement the tangent matrix employed by inexact Newton-type methods.

Three-dimensional non-linear finite element computations can yield linear systems of thousands and sometimes billions of equations. Several different approaches for designing and implementing parallel programs have been developed to solve such problems. However, two dominant alternatives have emerged: *message passing* and *multithreading*. According to Dongarra *et al* (2002) message passing is by far the most widely used approach to parallel computing, at least on large parallel systems. In the message-passing model, a computation comprises one or more processes that communicate by calling library routines to send and receive *messages*. There are several library-based implementations to provide these inter-processes communications, such as, *Message Passing Interface* (MPI) and *Parallel Virtual Machine* (PVM). In this work we have used the MPI standard. The message passing model has the advantages that programs become highly portable and the programmer has explicit control over the memory used by each process (Dongarra *et al* 2002). A crucial task when developing a message passing parallel program is to decompose the global domain into pieces, or partitions, which can be run in parallel with minimal communication and good load balance. Algorithms that find good partitionings over unstructured meshes are critical for efficient execution. In this work we have used the widely known Metis package (Karypis and Kumar 1998) to perform this task.

The present paper is organized as follows. Sections 2 and 3 present the governing equations and the SUPG/PSPG finite element formulation. Section 4 introduces the

inexact Newton-type schemes under consideration. Section 5 describes some issues related to our parallel implementation. Performance studies for the three-dimensional leaky driven cavity flow at Reynolds 400 test problem in different meshes and PC clusters are presented in Section 5 and the paper ends with a summary of our main conclusions.

2 GOVERNING EQUATIONS

Let $\Omega \subset \mathbb{R}^{n_{sd}}$ be the spatial domain, where n_{sd} is the number of space dimensions. Let Γ denote the boundary of Ω. We consider the following velocity-pressure formulation of the Navier-Stokes equations governing steady incompressible flows:

$$\rho\left(\mathbf{u}\cdot\nabla\mathbf{u}-\mathbf{f}\right)-\nabla\cdot\boldsymbol{\sigma}=\mathbf{0} \qquad \text{on } \Omega \tag{1}$$

$$\nabla\cdot\mathbf{u}=\mathbf{0} \qquad \text{on } \Omega \tag{2}$$

where ρ and $\mathbf{u}$ are the density and velocity, $\boldsymbol{\sigma}$ is the stress tensor given as

$$\boldsymbol{\sigma}\left(p,\mathbf{u}\right)=-p\mathbf{I}+\mathbf{T}, \tag{3}$$

where p is the hydrostatic pressure, $\mathbf{I}$ is the identity tensor and $\mathbf{T}$ is the deviatoric stress tensor.

The relationship between the stress tensor and deformation rate for Newtonian fluids is defined by a proportionality constant that represents the momentum diffusion experienced by the fluid. Therefore, the deviatoric stress tensor in equation (3) can be expressed by

$$\mathbf{T}=2\mu\,\boldsymbol{\varepsilon}\left(\mathbf{u}\right) \tag{4}$$

where μ is the proportionality constant known as dynamic viscosity and $\boldsymbol{\varepsilon}$ is deformation rate tensor or

$$\boldsymbol{\varepsilon}\left(\mathbf{u}\right)=\tfrac{1}{2}\left[\nabla\mathbf{u}+\left(\nabla\mathbf{u}\right)^{\mathrm{T}}\right] \tag{5}$$

The essential and natural boundary conditions associated with equations (1) and (2) can be imposed at different portions of the boundary Γ and represented by,

$$\mathbf{u}=\mathbf{g} \qquad \text{on } \Gamma_g \tag{6}$$

$$\mathbf{n}\cdot\boldsymbol{\sigma}=\mathbf{h} \qquad \text{on } \Gamma_h \tag{7}$$

where Γ_g and Γ_h are complementary subsets of Γ.

3 FINITE ELEMENT FORMULATION

Let us assume following Tezduyar (1991) that we have some suitably defined finite-dimensional trial solution and test function spaces for velocity and pressure, $S_{\mathbf{u}}^{h}$, $V_{\mathbf{u}}^{h}$, S_{p}^{h}

and $V_p^h = S_p^h$. The finite element formulation of equations (1) and (2) using SUPG and PSPG stabilizations for incompressible fluid flows can be written as follows: find $\mathbf{u}^h \in S_{\mathbf{u}}^h$ and $p^h \in S_p^h$ such that $\forall \mathbf{w}^h \in V_{\mathbf{u}}^h$ and $\forall q^h \in V_p^h$:

$$\begin{aligned}\int_\Omega \mathbf{w}^h \cdot \rho\left(\mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f}\right) d\Omega + \int_\Omega \boldsymbol{\varepsilon}\left(\mathbf{w}^h\right) : \boldsymbol{\sigma}\left(p^h, \mathbf{u}^h\right) d\Omega - \int_\Gamma \mathbf{w}^h \cdot h \, d\Gamma + \int_\Omega q^h \nabla \cdot \mathbf{u}^h d\Omega \\ + \sum_{e=1}^{n_{el}} \int_\Omega \tau_{SUPG} \mathbf{u}^h \cdot \nabla \mathbf{w}^h \cdot \left[\rho\left(\mathbf{u}^h \cdot \nabla \mathbf{u}^h\right) - \nabla \cdot \boldsymbol{\sigma}\left(p^h, \mathbf{u}^h\right) - \rho \mathbf{f}\right] d\Omega \\ + \sum_{e=1}^{n_{el}} \int_\Omega \frac{1}{\rho} \tau_{PSPG} \nabla q^h \cdot \left[\rho\left(\mathbf{u}^h \cdot \nabla \mathbf{u}^h\right) - \nabla \cdot \boldsymbol{\sigma}\left(p^h, \mathbf{u}^h\right) - \rho \mathbf{f}\right] d\Omega = 0\end{aligned} \tag{8}$$

In the above equation the first four integrals on the left hand side represent terms that appear in the Galerkin formulation of the problem (1)-(7), while the remaining integral expressions represent the additional terms which arise in the stabilized finite element formulation of the problem. Note that the stabilization terms are evaluated as the sum of element-wise integral expressions. The first summation corresponds to the SUPG (Streamline Upwind Petrov/Galerkin) term and the second correspond to the PSPG (Pressure Stabilization Petrov/Galerkin) term. We have calculated the stabilization parameters as described in Tezduyar (1991).

The spatial discretization of equation (8) leads to the following system of nonlinear equations,

$$\begin{aligned}\mathbf{N}(\mathbf{u}) + \mathbf{N}_\delta(\mathbf{u}) + \mathbf{K}\mathbf{u} - \left(\mathbf{G} + \mathbf{G}_\delta\right)\boldsymbol{p} &= \mathbf{f}_{\mathbf{u}} \\ \mathbf{G}^T\mathbf{u} + \mathbf{N}_\varphi(\mathbf{u}) + \mathbf{G}_\varphi \boldsymbol{p} &= \mathbf{f}_{\boldsymbol{p}}\end{aligned} \tag{9}$$

where $\mathbf{u}$ is the vector of unknown nodal values of $\mathbf{u}^h$ and $\boldsymbol{p}$ is the vector of unknown nodal values of $\boldsymbol{p}^h$. The non-linear vectors $\mathbf{N}(\mathbf{u})$, $\mathbf{N}_\delta(\mathbf{u})$, and $\mathbf{N}_\varphi(\mathbf{u})$ the matrices $\mathbf{K}$, $\mathbf{G}$, $\mathbf{G}_\delta$, and $\mathbf{G}_\varphi$ emanate, respectively, from the convective, viscous and pressure terms. The vectors $\mathbf{f}_{\mathbf{u}}$ and $\mathbf{f}_{\boldsymbol{p}}$ are due to the boundary conditions (6) and (7). The subscripts δ and φ identify the SUPG and PSPG contributions respectively. In order to simplify the notation we denote by $\mathbf{x} = (\mathbf{u}, \boldsymbol{p})$ a vector of nodal variables comprising both nodal velocities and pressures. Thus, equation (9) can be written as,

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \tag{10}$$

where $\mathbf{F}(\mathbf{x})$ represents a nonlinear vector function.

For Reynolds numbers much greater than unity, the nonlinear character of the equations becomes dominant, making the choice of the solution algorithm, especially with respect to its convergence and efficiency a key issue. The search for a suitable nonlinear solution method is complicated by the existence of several procedures and their variants. In the following section we present the nonlinear solution strategies based on the Newton-type methods evaluated in this work.

4 NONLINEAR SOLUTION PROCEDURES

Consider the nonlinear problem arising from the discretization of the fluid flow equations described by equation (10). We assume that $\mathbf{F}$ is continuously differentiable in $\mathbb{R}^{n_{sd}}$ and denote its Jacobian matrix by $\mathbf{F}' \in \mathbb{R}^{n_{sd}}$. The Newton's method is a classical algorithm for solving equation (10) and can be enunciated as: given an initial guess $\mathbf{x}_0$, we compute a sequence of steps $\mathbf{s}_k$ and iterates $\mathbf{x}_k$ as follows:

ALGORITHM N

`for` $k = 0$ `step` 1 `until convergence do`

$$\texttt{solve } \mathbf{F}'(\mathbf{x}_k)\mathbf{s}_k = -\mathbf{F}(\mathbf{x}_k) \tag{11}$$

$$\texttt{set } \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$$

Newton's method is attractive because it converges rapidly from any sufficiently good initial guess (see Dembo 1982). However, one drawback of Newton's method is the need to solve the Newton equations (11) at each stage. Computing the exact solution using a direct method can be expensive if the number of unknowns is large and may not be justified when $\mathbf{x}_k$ is far from a solution. Thus, one might prefer to compute some approximate solution, leading to the following algorithm:

ALGORITHM IN

`for` $k = 0$ `step` 1 `until convergence do`

`find some` $\eta_k \in [0,1)$ `AND` $\mathbf{s}_k$ `that satisfy`

$$\left\|\mathbf{F}(\mathbf{x}_k) + \mathbf{F}'(\mathbf{x}_k)\mathbf{s}_k\right\| \le \eta_k \left\|\mathbf{F}(\mathbf{x}_k)\right\| \tag{12}$$

$$\texttt{set } \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$$

For some $\eta_k \in [0,1)$, where $\|\bullet\|$ is a norm of choice. This formulation naturally allows the use of an iterative solver: one first chooses η_k and then applies the iterative solver to (11) until a $\mathbf{s}_k$ is determined for which the residual norm satisfies (12). In this context η_k is often called a forcing term, since its role is to force the residual of (11) to be suitably small. This term can be specified in several ways and we have followed the approach described in Eisenstat and Walker 1996 to enhance efficiency and convergence. We have adopted $\eta_{max} = 0.99$ arbitrarily in our tests. We have used a parallel nodal block-diagonal preconditioned element-by-element (EBE) GMRES method to compute $\mathbf{s}_k$ such as equation (11) holds. A particularly simple scheme for solving the nonlinear system of equations (10) is a fixed point iteration procedure known as successive substitution (SS) (also known as Picard iteration, functional iteration or successive iteration). Note in the algorithms above that, if we do not build the Jacobian matrix in equations (11) and (12), and the solution of previous iterations were used, we have a successive substitution (SS) method. In this work, we have evaluated the efficiency of Newton and successive substitution methods in their inexact versions. We may also define a mixed strategy combining SS and N (or ISS and IN) iterations, to improve performance, as described in Elias *et al* (2003).

To form the Jacobian $\mathbf{F}'$ required by Newton-type methods we use a numerical approximation described in Tezduyar (1999). Consider the following Taylor expansion for the nonlinear convective term emanating from the Galerkin formulation:

$$\mathbf{N}(\mathbf{u}+\Delta\mathbf{u}) = \mathbf{N}(\mathbf{u}) + \frac{\partial \mathbf{N}}{\partial \mathbf{u}}\Delta\mathbf{u} + \ldots \tag{13}$$

where $\Delta\mathbf{u}$ is the velocity increment. Discarding the high order terms and omitting the integral symbols we arrive to the following approximation,

$$\begin{aligned}\rho(\mathbf{u}+\Delta\mathbf{u})\cdot\nabla(\mathbf{u}+\Delta\mathbf{u}) \cong\ & \rho(\mathbf{u}\cdot\nabla)\mathbf{u} \\ & +\rho(\mathbf{u}\cdot\nabla)\Delta\mathbf{u} + \rho(\Delta\mathbf{u}\cdot\nabla)\mathbf{u}\end{aligned} \tag{14}$$

Note that the first term in the right hand side of equation (11) is the corresponding residual vector and the remaining terms represent the numerical approximation of $\partial\mathbf{N}/\partial\mathbf{u}$. If we apply similar derivations to $\mathbf{N}_\delta(\mathbf{u})$ and $\mathbf{N}_\varphi(\mathbf{u})$ we arrive to the SUPG and PSPG contributions to the residual vector and to the approximations of $\partial\mathbf{N}_\delta/\partial\mathbf{u}$ and $\partial\mathbf{N}_\varphi/\partial\mathbf{u}$.

5 PARALLEL IMPLEMENTATION

The formulation presented in the previous sections was implemented using the message passing parallel model with the MPI standard. Thus, we need to partition the global mesh into minor pieces or non-overlapped sub-domains. The data communication must be restricted to only the equations associated with interface nodes, as shown in Figure 1. Thus, it is desirable an algorithm able to perform mesh partitioning minimizing the number of interface nodes and making a good load balance among processes with partitions distributed equally or according to the computational system. In this work we have used the METIS_PartMeshDual routine provided by Metis package (Karypis and Kumar 1998) to accomplish this task.

Other important optimizations are avoiding indirect memory addressing and IF clauses. This can be achieved by performing a mesh reordering after the domain decomposition. We have done these tasks in a preprocessing level where nodes are renumbered according to its partitions and the number of interface nodes, and elements are renumbered by their partitions. The main tasks developed in the preprocessor level are summarized in Figure 1.

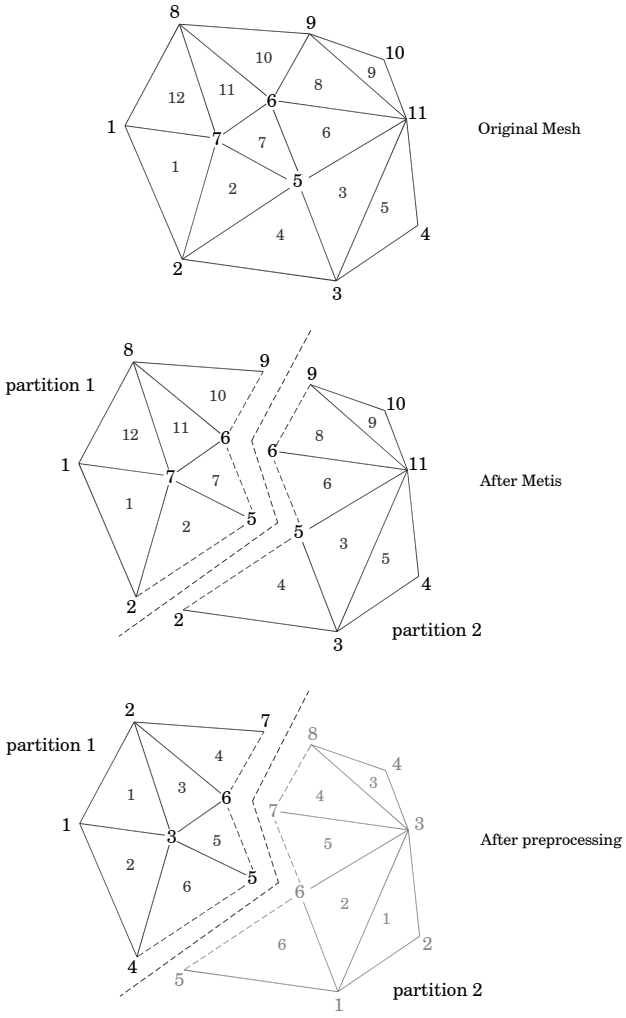

Figure 1 – Domain decomposition and mesh reordering.

Note in Figure 1 that elements, internal and interface nodes are re-defined into sequential ranges at each partition. Elements are reordered according to the number of nodes at the interfaces in a hierarchical manner. Thus, any data can be easily distributed and managed over distributed systems such as PC clusters, avoiding indirect addressing and IF sentences. Data distribution can also take advantage of element-by-element (EBE) storage scheme, where the unassembled stiffness matrix is stored at element level.

Most of the computational effort spent in iterative solution of linear systems is due to evaluations of matrix-vector products, or simply *matvec* operations. In our tests *matvec* operations achieved 92% of the total computational costs. In EBE storage this task is parallelizable by performing *matvec* operations at each partition level, then collecting the contribution of interface equations by calling MPI_AllReduce routine. The P-EBE MAT-VEC (*Parallel Element by Element Matrix-Vector*) scheme between two processes is shown in Figure 2.

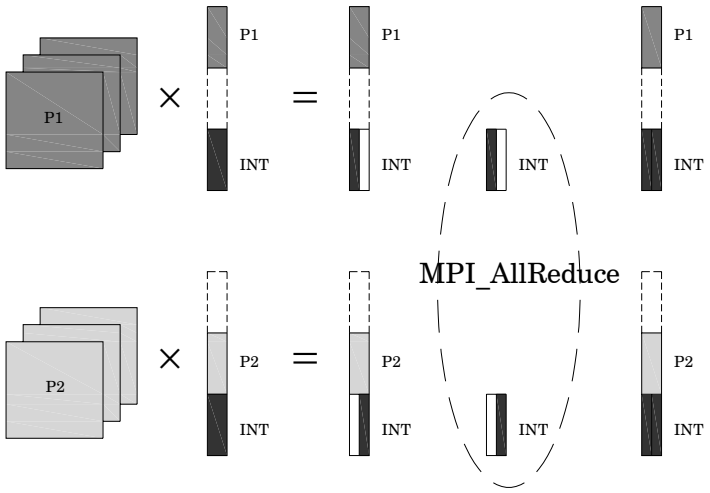

Figure 2 – MPI Parallel Element-by-Element Matrix-Vector operation.

Note in Figure 2 that the communication region is restricted to only the interface equations. Other vector-vector computations needed by the iterative linear solver are accomplished with BLAS level 1 routines slightly modified to take into account message passing parallelism. Figure 3 shows an example of BLAS `ddot` routine modified to support parallelism with message passing. Note that others BLAS level 1 routines such as `dscal`, `daxpy`, `dcopy` and `dclear` do not require any communication. These routines are performed at partition and interface levels individually.

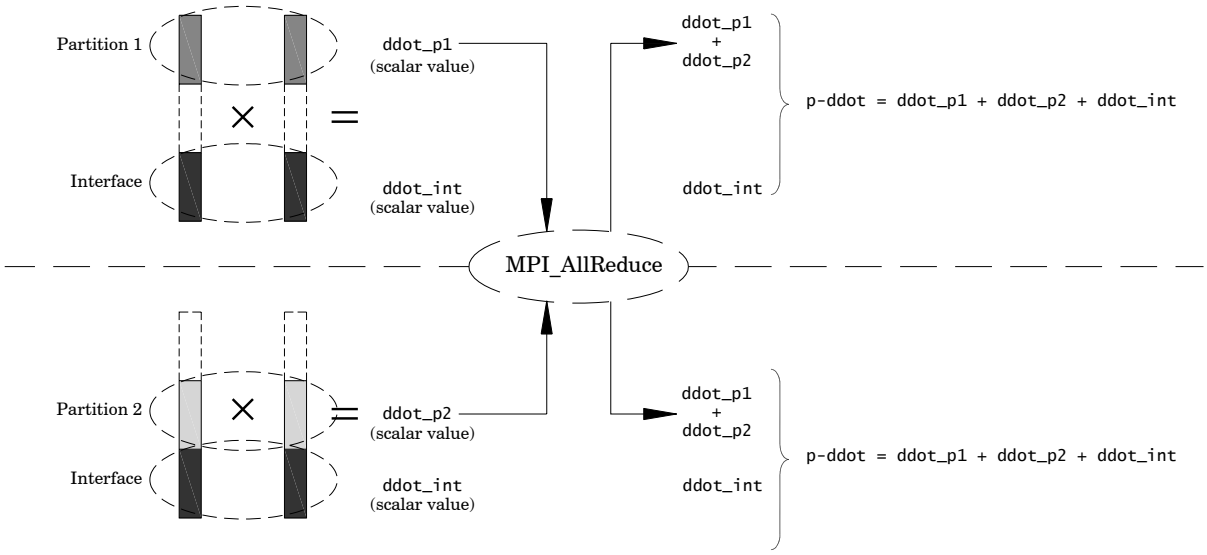

Figure 3 – `p-ddot` – Parallel dot product with message passing and BLAS level 1.

Note in Figure 3 that in `p-ddot` routine the communication is restricted to only scalar numbers due the partial dot products performed at each partition with the conventional BLAS `ddot` routine.

6 RESULTS

In this work we employed the three-dimensional leaky lid driven cavity flow as benchmark. The numerical procedure considers a coupled **u**-*p* version for the stabilized formulation using linear tetrahedron elements. The parallel-GMRES(25) linear solver with nodal block diagonal preconditioner is employed as the inner iterative driver and the computations were performed until the maximum residual and relative error decreased 3 orders of magnitude.

The computations have been performed on the Infoserver Itautec PC Cluster (16 nodes dual Intel Pentium III - 1 GHz, 512 Mb of memory per node, interconnected by Gigabit network, Intel Fortran compiler, LAM-MPI and platform Red Hat Linux 7.3). Some tests of code portability were performed on a mini-cluster fast-ethernet/wireless composed by 4 laptop nodes with Intel Centrino processors and running Microsoft Windows 2000/XP platform. All hardware is located at Center for Parallel Computations of the COPPE/UFRJ.

The classical problem of the closed cavity driven by the motion of a leaky lid has been used rather extensively as a validation test case by many authors (see Ghia *et al*, 1982 for details). In this problem a unit velocity is specified along the entire top surface and zero velocity on the other surfaces as shown in Figure 4.

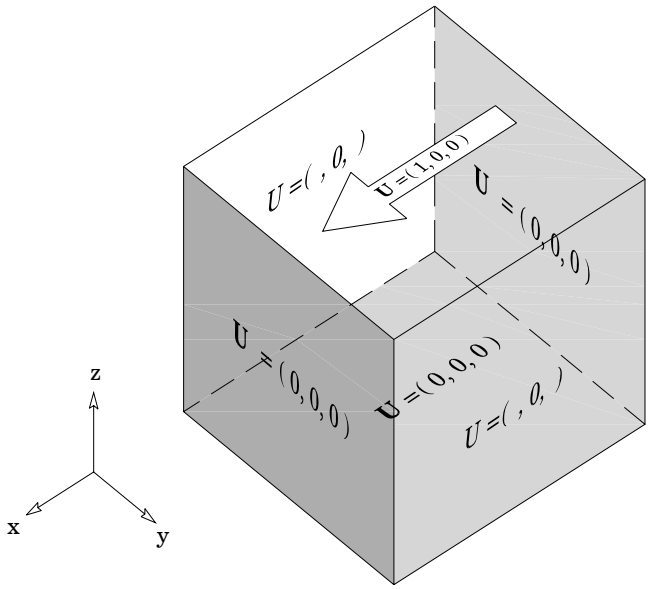

Figure 4 – Flow in a leaky lid-driven cavity. Geometry and boundary conditions.

Tests were performed considering two meshes with 31 and 51 divisions along the edges of the cavity. Figure 5 shows examples of these meshes decomposed by Metis. Reynolds number of 400 was considered for all tests and details about the meshes can be accessed in Table 1.

Table 1. Problem dimensions

	31x31x31	51x51x51
Elements	148,955	663,255
Nodes	32,768	140,608
Equations	117,367	525,556

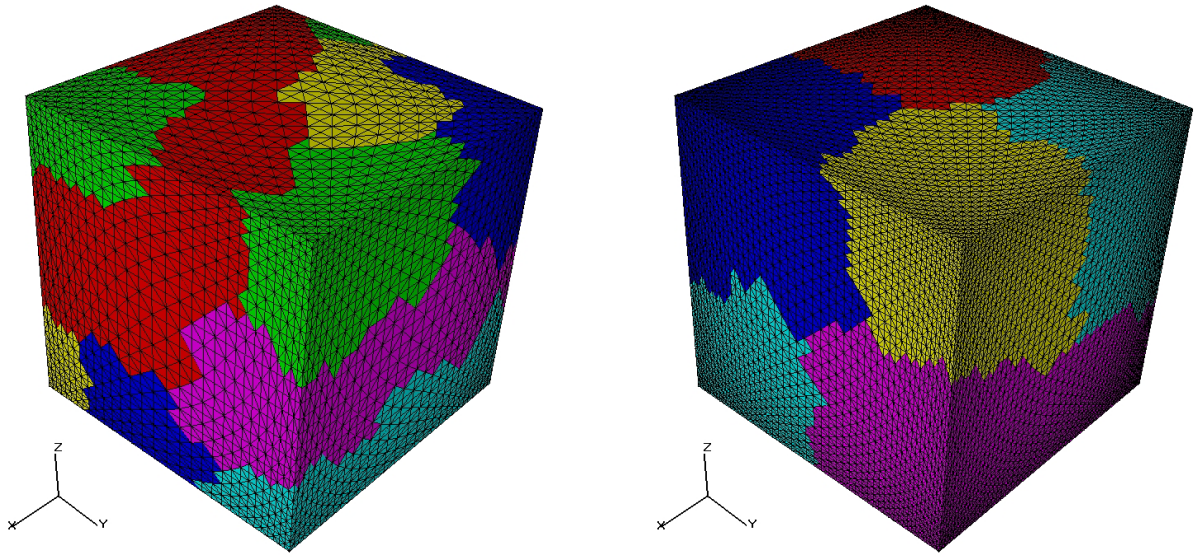

Figure 5 – Meshes partitioned by Metis (Left) 31x31x31 with 16 partitions and (right) 51x51x51 with 8 partitions.

Figure 6 shows the velocity and pressure fields. We can note the vortex formation and the pressure singularities at the cavity corners, typical of this problem. In Figures 7 and 8 the inexact Newton-type methods are compared for the meshes employed. Figure 7 (left) shows the adapted linear tolerances selected by the inexact solver as described in Section 4.

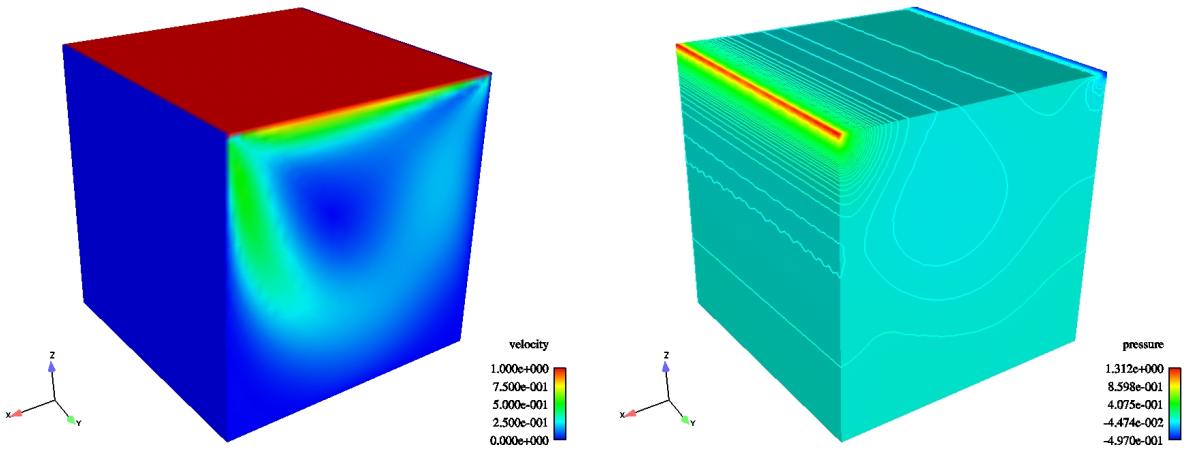

Figure 6 – Results for the leaky lid cavity flow Reynolds 400
(left) Velocity (right) pressure fields

Figure 7 (right) shows the cumulative time spent at the nonlinear solvers as the nonlinear iterations progress. In Figure 8 we have a comparison between the nonlinear inexact methods studied.

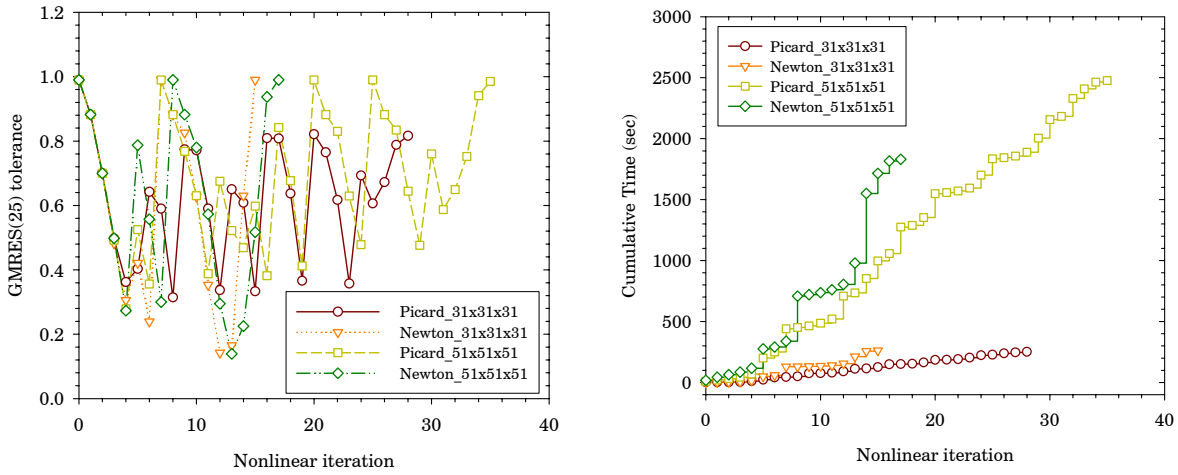

Figure 7 – (left) Adaptative GMRES(25) tolerance due Inexact nonlinear solution – (right) Cumulative time solution.

Note in Figures 7 (left) and 8 that both nonlinear inexact methods were able to achieve convergent solutions and that was not necessary linear tolerances bellow 0.16. Figure 7 (right) shows that the inexact Newton method was faster than Picard iterations only for the most refined mesh, but in all tests the inexact Newton solution required less nonlinear iterations. We can note comparing Figures 7 (left and right) and 8 that the inexact nonlinear methods try to adjust the inner linear solver tolerance to the nonlinear convergence. Thus, the linear tolerance tightening is directly related to the relative nonlinear residual decreasing.

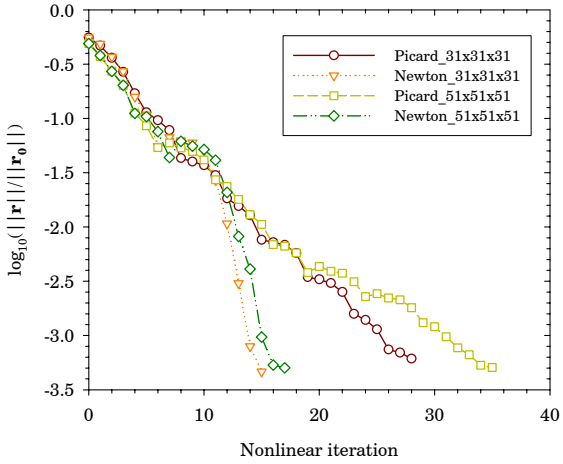

Figure 8 – Relative nonlinear residual – convergence of inexact Newton and Picard methods

Figures 9 and 10 show the parallel performance results. In Figure 9 (left and right) are shown the time spent for the Inexact Picard solution and the ratio of interface equations to the total number of equations for both meshes. In Figure 10 (left and right) are presented the parallel speedups and memory requirements.

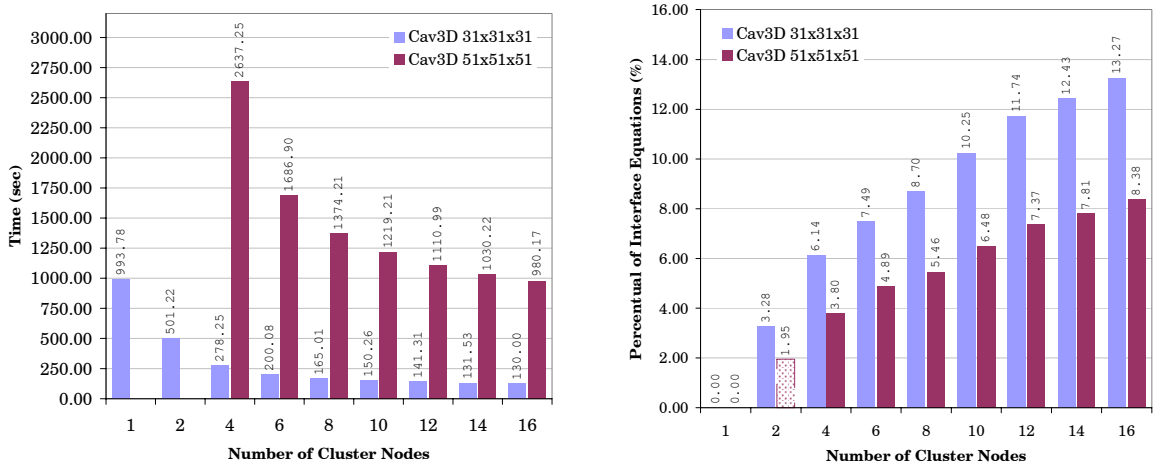

Figure 9 – (left) Solution time and (right) Percentage of interface equations

We can note comparing Figure 9 left and right that increasing the number of processing nodes the solution was faster. The speedup curve in Figure 10 shows that the larger model (51x51x51) took more advantage of the parallel inexact algorithm than the smaller one (31x31x31). The dashed column in Figures 9 and 10 (right) means that the problem could not be fitted in the memory available in each cluster node (512 Mb). Thus, these cases were not run. Most of the memory spent by the solution algorithm is due to the EBE matrices storage, but the parallel data distribution diminishes this memory overhead. The parallel data distribution associated with matrix-free techniques, such as described in Knoll and Keyes (2004), could make possible to solve large scale problems even with few processing nodes and less memory requirements.

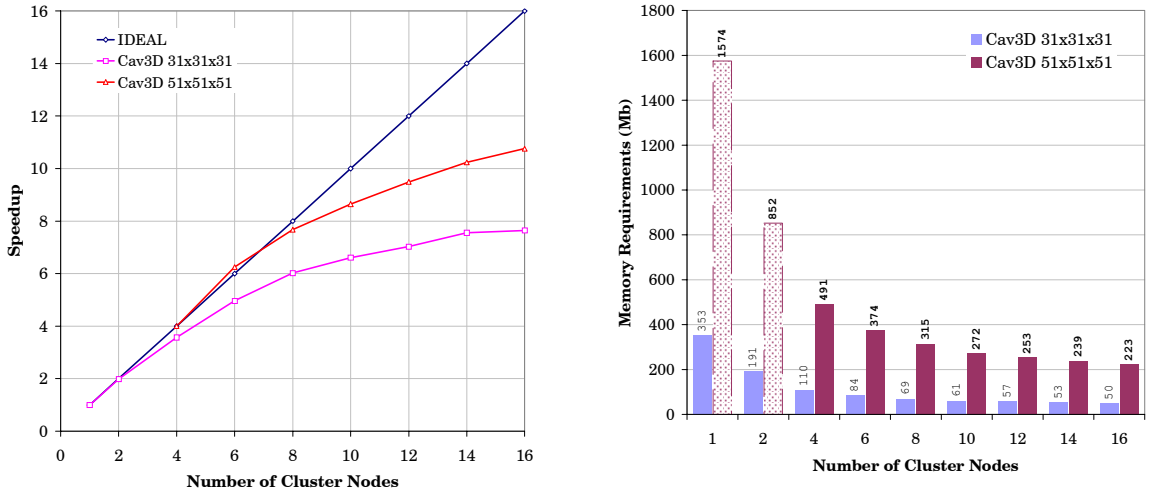

Figure 10 – (left) Speedup and (right) Memory requirements.

Figure 11 shows the results of tests performed on a mini-cluster formed by 4 laptops and a wireless/fast-ethernet network (2 Intel Centrino 1.6 GHz/512Mb, 1 Intel Centrino 1.3/512Mb GHz and 1 Intel Pentium 4 2.4 GHz/512Mb interconnected by a Linksys Wireless-B Hub, IEEE 802.11b/2.4 GHz/11Mbps or Fast-Ethernet 10/100Mbps network). These tests show the versatility and portability that message passing codes can offer, making possible the solution of even large scale problems employing modest machines.

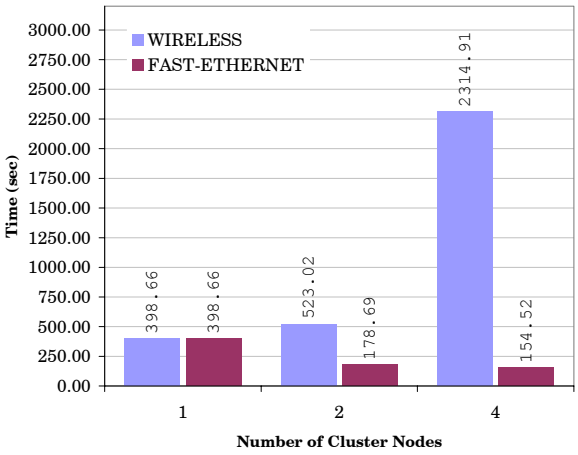

Figure 11 – (Left) Minicluster mobile wireless/fast-ethernet, (Right) Performance comparison between wireless and fast-ethernet networks.

Figure 11 (right) shows that the wireless technology employed (wireless-b) was not able to deliver the bandwidth required to reach a desirable speedup in this irregular MPI parallel computation. However, with the increasing bandwidth in wireless technology mobile-parallel computations will be a reality in the near future. The low speedups achieved in the minicluster with fast-ethernet network was also due to the small problem size, as occurred in the case shown in Figure 10 (left) for the Infoserver Itautec PC cluster.

7 CONCLUSIONS

We have tested the performance of inexact Newton-type algorithms to solve nonlinear systems of equations arising from the SUPG/PSPG finite element formulation of steady incompressible flows. We employed a numerically approximated Jacobian based on Taylor's expansion of the nonlinear convective terms emanating from the Galerkin and stabilization terms. We also introduced an inexact Picard scheme. Numerical tests in a 3D benchmark problem have shown the good parallel performance of the inexact Newton-type methods. The code is portable across different computer platforms, ranging from a mini-cluster with a fast-ethernet/wireless network composed by 4 laptop nodes to the Infoserver Itautec PC Cluster with 16 nodes dual Intel Pentium III - 1 GHz, 512 Mb of memory per node, interconnected by Gigabit network.

Acknowledgements

The authors would like to thank the financial support of the Petroleum National Agency (ANP, Brazil) and MCT/CNPq, The Brazilian Council for Scientific Research. The Center for Parallel Computations (NACAD) and Laboratory of Computational Methods in Engineering (LAMCE) at the Federal University of Rio de Janeiro provided the computational resources for this research.

REFERENCES

Brooks, A. N. and Hughes, T. J. R., 1982, *Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equation*, Comput. Methods Appl. Mech. Engrg. 32:

199-259.

Dembo, R. S., Eisenstat, S. C. and Steihaug, T., 1982, *Inexact Newton Methods*, SIAM J. Numer. Anal., 19: 400-408.

Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L. and White, A., 2002, *Sourcebook of Parallel Computing*, Morgan Kaufmann, San Francisco, USA.

Eisenstat, S. C. and Walker, H. F., 1996, *Choosing the Forcing Terms in Inexact Newton Method*, SIAM. J. Sci. Comput. 17–1: 16–32.

Elias, R. N., Coutinho, 2003, *Inexact Newton-Type Methods for Non-Linear Problems Arising from the SUPG/PSPG Solution of Steady Incompressible Navier-Stokes Equations*, M. Sc. thesis, Federal University of Rio de Janeiro, Brazil.
(http://www.nacad.ufrj.br/~rnelias)

Ghia, U., Ghia, K. N. and Shin, C. T., 1982, *High-Resolution for Incompressible Flow Using the Navier-Stokes Equations and Multigrid Method*. J. Comp. Phys., 48: 387-411.

Hughes, T. J. R., Franca, L. P. and Balestra, M., 1986, *A New Finite Element Formulation for Computational Fluid Dynamics: V. Circumventing the Babuška-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accommodating Equal-Order Interpolations*, Comput. Methods Appl. Mech. Engrg., 59: 85-99.

Karypis G. and Kumar V., 1998, *Metis 4.0: Unstructured Graph Partitioning and Sparse Matrix Ordering System*. Technical report, Department of Computer Science, University of Minnesota, Minneapolis;
(http://www.users.cs.umn.edu/~karypis/metis).

Kelley, C. T., 1995, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in applied mathematics, SIAM, Philadelphia.

Knoll, D. A. and Keyes, D. E., 2004, *Jacobian-free Newton-Krylov Methods: a Survey of Approaches and Applications*, Journal o f Computation Physics 193: 357-397.

Tezduyar, T. E., 1991, *Stabilized Finite Element Formulations for Incompressible Flow Computations*, Advances in Applied Mechanics, 28: 1-44.

Tezduyar, T. E., 1999, *Finite Elements in Fluids: Lecture Notes of the Short Course on Finite Elements in Fluids*, Computational Mechanics Division – Vol. 99-77, Japan Society of Mechanical Engineers, Tokyo, Japan.

Tezduyar, T. E., Mittal S., Ray S. E. and Shin R., 1992, *Incompressible Flow Computations with Stabilized Bilinear and Linear Equal-Order-interpolation Velocity-pressure Elements*, Comput. Methods Appl. Mech. Engrg. 95: 221-242.