BLOCK ILU PRECONDITIONERS FOR PARALLEL AMR/C SIMULATIONS

Jose J. Camata
Alvaro L. G. A. Coutinho
Federal University of Rio de Janeiro, NACAD, COPPE
Department of Civil Engineering, Rio de Janeiro, Brazil
{*camata,alvaro*}*@nacad.ufrj.br*
Andrea M. P. Valli
Lucia Catabriga
Federal University of Espírito Santo, LCAD,
Computer Science Department, Vitória, ES, Brazil.
{*avalli,luciac*}*@inf.ufes.br*
Graham. F. Carey
The University of Texas at Austin, ICES, CFD Lab
Austin, Texas, USA
carey@cfdlab.ae.utexas.edu

Abstract. *In this work we study the behavior of Block ILU preconditioners in distribuited-memory parallel simulations. Of particulat interest is the resulting preconditioned iterative solver behavior when adaptive mesh refinement and coarsening (AMR/C) are utilized. Here, we use a domain decomposition scheme to partition the spatial domain as a collection of subdomains. These subdomains are repartitioned dynamically when the mesh adaptation proceeds. To build approximate preconditioners, we employ the Block-Jacobi and Additive Schwarz techniques. In both cases, incomplete LU factorizations (ILU) are applied as local preconditioners. In addition to this, the Reverse Cuthill-McKee (RCM) and Quotient Minimum Degree (QMD) reodering algorithms are applied in the local ILU preconditioners. Numerical studies are conducted using the object oriented AMR/C software system libMesh linked to the PETSc solver library in order to compare the efficiency of Block ILU preconditioners. Preliminary results suggest that Block-Jacobi preconditioners have lower computation cost than Additive Schwarz.*

Keywords: *Block ILU preconditioner, Krylov subspace methods, Additive Schwarz, Domain decomposition*

1. INTRODUCTION

In general, implicit time integration schemes are used for temporal discretization when multiple spatial and temporal scales are present. Although these schemes have better perfomance when compared with explicit schemes, they require a solution of large and sparse linear system of equations. Krylov subspace iterative methods are now extensively used in conjunction with preconditioning strategies for large-scale computational engineering and science applications. Solver libraries such as PETSc (2009) and Trilinos (2009) have been developed to facilitate and promote their utilization. Incomplete LU factorization (ILU) preconditioners are a standard option with such libraries. Hence techniques for enhancing the performance of this class of solvers are of general interest. Examples of Krylov subspace solvers include the Conjugate Gradient (CG) method for symmetric systems (Hestenes and Stiefel, 1952) and the Generalized Minimal Residual (GMRES) method (Saad and Schultz, 1986), the Bi-Conjugate Gradient Stabilized (Bi-CGSTAB) method (van der Vorst, 1986) and the Left Conjugate Direction (LCD) method (Dai and Yuan, 2004; Catabriga et al., 2006) for non-symmetric systems.

Here we are interested in the role of parallel techniques to further enhance preconditioned iterative solvers. Unfortunately, a limited amount of parallelism can be extracted from standard precontitioners. One of the powerful preconditioning methods in terms of reducing the number of iterations is the ILU (incomplete LU) factorization method. However, it is very difficult to parallelize the ILU factorization process due to the recursive nature of the computation. In order to make the ILU factorizations more suitable for parallel architectures, a number of alternative techniques were developed using block preconditioners, that permits combination with local ILU factorizations. The simplest approach is Block-Jacobi preconditioner. This method consists of block-diagonals of A were each block coincide with the division of variables over the processors. Although this method is easy to implement, it often presents slow convergence rate (Saad and van der Vorst, 2000). Another approach widely used are Overlapping Schwartz methods and its variants. It combines nice parallel properties with high convergence rates (Silva et al., 1997; Gropp et al., 2001).

This work deals on how the choice of block ILU preconditioners affects the simulation performance. Of particular interest is the iterative solver behavior when adaptive mesh refinement (AMR) is utilized for parallel simulations. Our numerical studies are conducted using the oriented AMR software system libMesh with the PETSc Library (B. S. Kirk and Carey, 2006; PETSc, 2009). We also present performance results for natural unknowns orderings compared to Reverse Cuthill-McKee (RCM) and Quotient Minimum Degree (QMD) reorderings for a representative scalar field problem when local ILU is used. Without loss of generality, we confine the simulations to finite element discretizations but it is clear that the ideas and conclusions apply equally to similar systems coming from finite difference and finite volume approximations.

The organization of the paper is as follows: we first briefly present the test case for the subsequent numerical studies, indicating the discretization approach and commenting on algebraic system properties. This is followed by a summary of the block preconditioning strategies to be applied. Next, results of experiments with adaptive meshes on parallel systems are presented and discussed. Some concluding remarks on the performance and other points of interest complete the work.

2. GOVERNING EQUATION AND DISCRETIZATION

The system of equations considered is the general scalar transient transport equation:

$$\frac{du}{dt} + \mathbf{v} \cdot \nabla u - \nabla \cdot (D \nabla u) = f \quad \text{em } \Omega \tag{1}$$

$$u = g \quad \text{em } \partial\Omega_1 \tag{2}$$

$$-D\nabla u \cdot \mathbf{n} = h \quad \text{em } \partial\Omega_2 \tag{3}$$

where $D = D(u, \nabla u)$ in the nonlinear case or more simply D is a function of position (D is taken as constant in later numerical studies), f, g and h are known functions. To prevent spurious oscillation generated by the dominance of the convection term in the differential equation, we use a SUPG (Streamline-Upwind/Petrov-Galerkin) stabilization technique (Brooks and Hughes, 1982; Codina et al., 1992). Introducing a finite element discretization and corresponding basis functions to define the approximation space V^h, the SUPG weighted residual formulation for u_h is:

$$\begin{aligned} &\int_{\Omega_h} \left(\frac{du_h}{dt} w_h + (\mathbf{v_h} \cdot \nabla u_h) w_h + D \nabla u_h \cdot \nabla w_h \right) d\Omega \\ + &\sum_{e=1}^{E} \int_{\Omega_e} \tau \frac{\mathbf{v_h} \cdot \nabla w_h}{||\mathbf{v_h}||} \left(\frac{du}{dt} + \mathbf{v_h} \cdot \nabla u_h - \nabla \cdot (D \nabla u_h) - f \right) d\Omega = \\ &\int_{\Omega_h} f w_h d\Omega + \int_{\partial\Omega_2} h w_h d\Omega \end{aligned} \tag{4}$$

where the first and RHS integrals correspond to the standard Galerkin form and the second integral is the dissipative SUPG term added to the variational formulation to prevent numerical oscillations on coarse meshes. The parameter τ is computed as suggested by in Codina et al. (1992). Introducing a finite element approximations for u_h into Equation (5), we obtain a semi-discrete ODE system which is integrated using a standard θ-method. In the experiments, the Crank-Nicolson ($\theta = 1/2$) method is used. If D is a function of position or constant, as in the later numerical cases, then the resulting nonlinear systems simplify to linear nonsymmetric, of the form $Ax = b$, for the vector of nodal unknowns at the grid points defining the discretization.

3. PARALLEL AMR/C SIMULATIONS

Adaptive mesh refinement has been used for some time now to enhance grids using local element subdivision or basis enrichment and thereby resolve different scales such as those of boundary and interior layers. For serial computations, when the AMR scheme is invoked, the size and structure of the discrete system changes dynamically at each mesh adaptation stage. In the case of parallel computations a partitioner embedded with the solver library is typically applied to distribute the computations across processors and to avoid the development of a significant load imbalance during the AMR process. In turn this implies recomputing the corresponding global ILU at each repartitioning step, however, this process is very complex to implement in parallel systems.

A more interesting and perhaps natural situation arises in the parallel case if block local ILU preconditioning is applied instead of their global counterparts. Of course the action of this block ILU preconditioning will be different than in the previous global factorizations and parallel construction of the ILU factors are simpler and more efficient. With AMR, the mesh is changing dynamically. Repartitioning will be required when sufficient imbalance occurs. Since a different element group now resides on each processor after a repartition, a new local reordering and local factorization for that partition is needed. This problem is a challenge and domain decomposition preconditioners can be a very good answer to this.

3.1 Domain Decomposition Preconditioners

Domain decomposition methods have been studied extensively in the past few years. The increasing success of these methods is due to the fact that they provide a high level of concurrency and are simple to implement on most modern parallel computers (Gropp et al., 2001). Although they can be used directly as iterative methods, in this work we consider two domain decomposition methods as preconditioners for standard Krylov subspace iterative solvers.

Additive Schwarz preconditioner. Each processor solves a local subsystem including bordering variables which belong to other processors. The choice of how to exchange these bordering variables will define the Schwarz method used: multiplicative or additive. In the first one, each subdomain uses the solution recently evaluated as boundary values. In essence, it is a sequential method and to obtain parallelism in this version, is necessary to use some subdomain coloring scheme. On the other hand, in the additive scheme, all subdomains uses the last solution computed as boundary values. Thus, this schemes presents better parallelism properties than multiplicative schemes (Gropp et al., 2001).

To describe the additive Schwarz algorithm, we define a graph $G = (W, E)$, where the set of vertices $W = 1...n$ represents the n unknowns and the edge set $E = (i, j)|a_{i,j} \neq 0$ represents the pairs of vertices that are coupled by a nonzero element in *A*. Assume that graph *G* was decomposed into *N* nonoverlapping sets of vertices W_i^0 whose union is W. Let W_i^1 be the one overlap partition of W, where $W_i^1 \supset W_i^0$ is obtained by including all the immediate neighbouring vertices of the vertices in W_i^0 . Recursively, we can define a δ overlap partition of W, that is,

$$W = \bigcup_{i=1}^{N} W_i^{\delta} \tag{5}$$

where $W_i^{\delta} \supset W_i^0$ with δ levels of overlap with its neighbouring subdomains. Let us define a restriction matrix R_i that returns the coefficients belonging to W_i^{δ}. With this we define the local matrices A_i by

$$A_i = (R_i^{\delta})^T A R_i^{\delta} \tag{6}$$

Thus, the Additive Schwarz preconditioner (ASM) can be defined by

$$M_{AS}^{-1} = \sum (R_i^{\delta})^T A_i^{-1} R_i^{\delta} \tag{7}$$

Block-Jacobi preconditioner. The Block-Jacobi is the simplest domain decomposition method. They can be regarded as a zero-overlap form of Additve Schwarz. This method consists of block-diagonals of A were each block coincide with the division of variables over the processors. Early experiences with this idea can be found in Kalro and Tezduyar (1998).

In both methods, the incomplete LU factorizations can be applied in the local block. Considering Block Jacobi, a non-overlapping incomplete factorization is performed over the main diagonal block of the local part of the matrix *A*, thus avoiding extra communication. On the other hand, when the ASM preconditioner is used, each processor block overlaps to the neighbouring domain block by the amount δ levels. An ILU is performed over each processor block. A greater overlap level means more communication is necessary between the processors.

3.2 Implementation Issues

Here we use the open-source, C++ finite element library, libMesh (B. S. Kirk and Carey, 2006) in our tests. A major goal of libMesh is to provide a platform for parallel, adaptive, multiphysics finite element simulations. libMesh users can focus on the specifics of a given application without having to develop the additional complexities for parallel adaptive mesh computing. In this way libMesh has proved a valuable testbed for a wide range of physical applications. On the other hand, libMesh uses the well-known PETSc library (PETSc, 2009) to solve the linear equation systems. PETSc implements several Krylov interactive solvers and preconditioners, including Block-Jacobi and one-level Addtive Schwarz. PETSc permits that we use a local preconditioner on each subdomain (processor). In this paper, we use ILU factorizations.

4. NUMERICAL RESULTS

We chose the three-dimensional deformation problem that has been used to investigate mass (volume) loss issues associated with level set methods (Osher and Fedkiw, 2000). It superposes deformation in the $x - y$ plane with deformation in the $x - z$ plane. In this case, a sphere is entrained by vortices and stretched out very thinly, before the flow times return the sphere to its original form. Following LeVeque (1996), the velocity field is given by

$$\begin{aligned} u_x &= 2sin^2(\pi x)\, sin(2\pi y)\, sin(2\pi z)\; g(t) \\ u_y &= -sin(2\pi x)\, sin^2(\pi y)\, sin(2\pi z)\; g(t) \\ u_z &= -sin(2\pi x)\, sin(2\pi y)\, sin^2(\pi z)\; g(t) \end{aligned} \tag{8}$$

The time dependent function $g(t)$ is used to reverse the flow field at time $T/2$ so that the initial data should be recovered at time T. The problem comprises a sphere of radius 0.15 centered at (0.35, 0.35, 0.35) in a unit domain. In this test a unstructured mesh with 1,130,949 linear tetrahedra is used. The refine and coarse fractions are 0.9 and 0.1, respectively. In addition, the maximum level that elements can be divided is 4. The AMR procedure is employed every 5 time steps.

The numerical tests were carried out on the SGI Altix ICE 8200 with 32 nodes. Each node has two Intel Xeon quad-core (2.66GHz / 4MB L2) and 8GB of memory. The nodes are connected by a InfiniBand network. The MPI over InfiniBand is OpenMPI version 1.2.8. The libMesh and PETSc were compiled with Intel compilers version 10.1. In all cases, the linear solver is the GMRES(5) and the linear tolerance is 10^{-6}. PETSc provides parallel preconditioners that can be used with incomplete LU factorizations on each subdomain (on each processor). They are Block Jacobi and one-level overlapping Additive Schwartz (ASM). In the experiments we used both preconditioners and compare solutions with natural, RCM and QMD orderings in local ILU(0) and ILU(1) preconditioners.

Figure 1 illustrates the solution in three different times. To verify the solution at 3 time units, we compared its volume with the initial volume. Volumes are obtained by a Paraview filter. At the last time step volume loss is 14.5%. More accurate solutions can be obtained using enhancing the formulation as shown in Elias and Coutinho (2007) and/or adding more refinement/coarsening levels. In our experiments we use the standard SUPG formulation and adaptive meshes generated by the AMR process to reduce the volume loss.

In the series of parallel investigations that follows we compare the solver performance using Block-Jacobi and Additive Schwarz with ILU(0) and ILU(1) preconditioners and different reordering schemes. Table 1 shows the CPU time for runs with 4, 8, 16, 32 and 64 nodes. The symbol † means that convergence failed. We can observe that ASM gives the best CPU

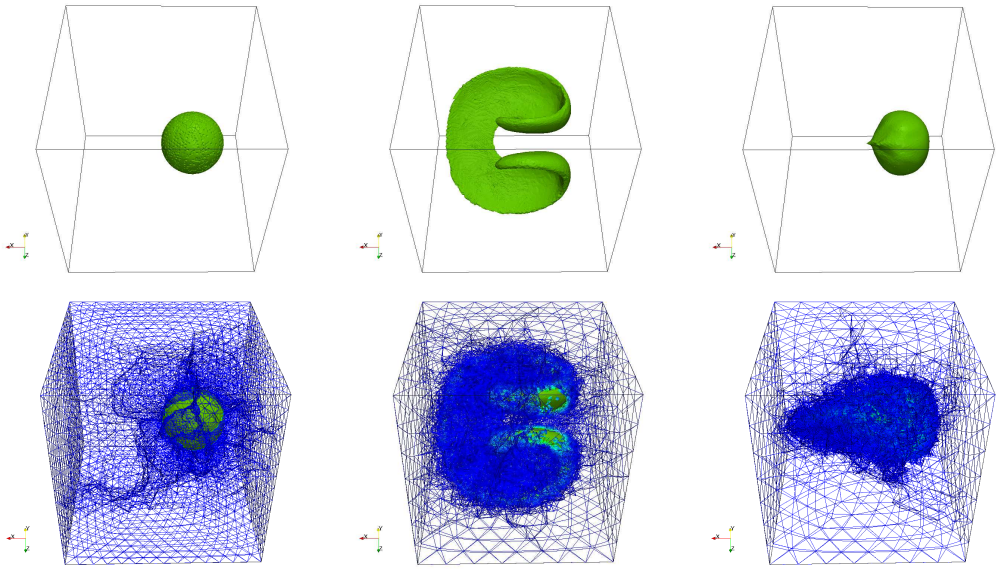


Figure 1: Solution and mesh configuration of the 3D deformation problem at three times, $T = 0$, $T = 1.5$, $T = 3.s$.

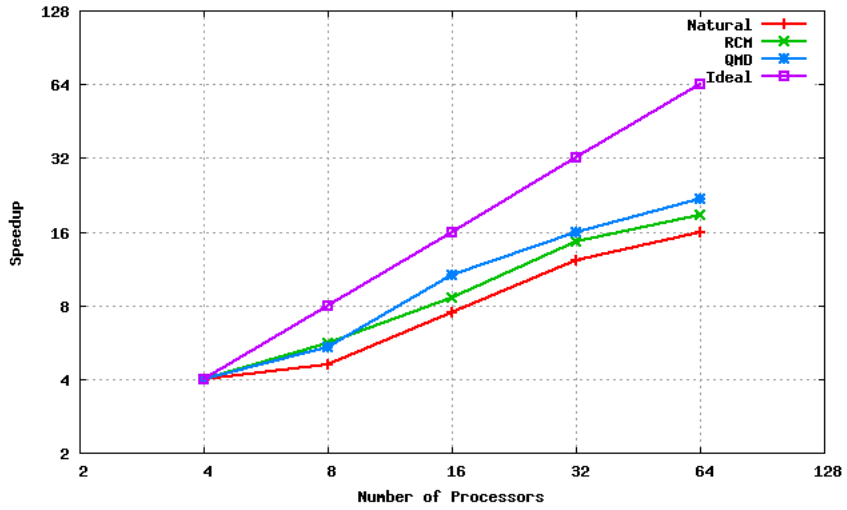
times when up to 16 processors are used. Increasing the number of processors, the ASM's communication cost affects its overall performance. Thereby, the Block-Jacobi has smaller CPU time for high processors counts. Comparing the ordering schemes, RCM presents the best CPU times using the ILU(0) preconditioner. This demonstrates that the preconditioners are very sensitive to the orderings.

Table 1: The Solver CPU time for the 3D deformation problem.

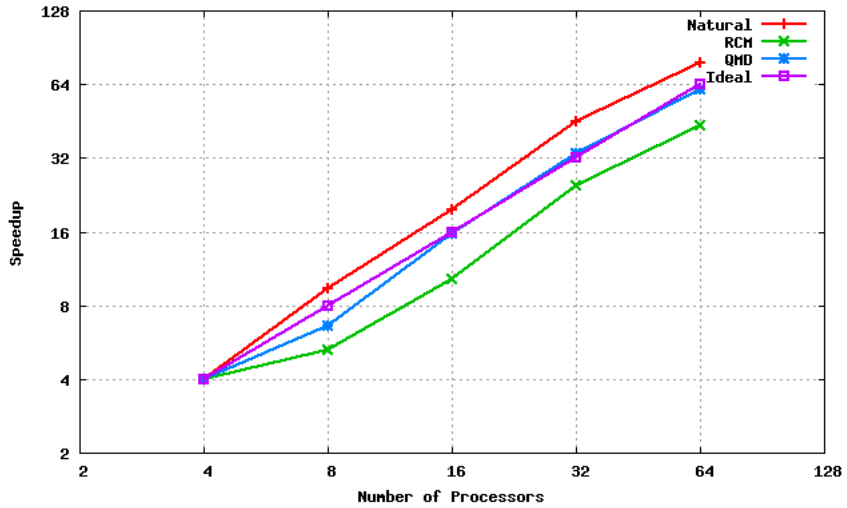
Additive-Schartz						
	Natural		RCM		QMD	
Processors	ILU(0)	ILU(1)	ILU(0)	ILU(1)	ILU(0)	ILU(1)
4	282.15	†	321.18	385.31	402.31	538.88
8	247.50	894.02	228.95	286.97	296.59	325.56
16	150.07	546.30	148.72	184.00	150.60	189.45
32	92.95	328.49	88.41	112.64	101.52	114.46
64	70.69	210.82	68.62	82.57	73.73	81.629
Block-Jacobi						
	Natural		RCM		QMD	
Processors	ILU(0)	ILU(1)	ILU(0)	ILU(1)	ILU(0)	ILU(1)
4	782.63	†	405.70	657.61	619.30	†
8	334.12	1091.60	309.53	370.37	376.36	378.62
16	158.04	573.65	158.50	190.19	158.71	181.65
32	69.33	279.19	65.79	85.18	74.19	81.28
64	39.78	129.08	37.44	43.54	40.56	43.66

Figure 2 shows the strong scalability for Additive Schwarz and Block-Jacobi, respectively. We can observe that Block-Jacobi scales better than additive Schwarz methods. In addition, although RCM ordering gives the best CPU times, the other ordering techniques are more scalable when Block-Jacobi and ILU are used. In this case, we have super linear speedup for the

natural ordering parallel computations in any number of processors. One possible reason for this the fact that no additional calculations are needed when reordering is not applied to local subdomains for block ILU preconditioning. We can also note that QMD scales better than RCM with a speedup closer to the ideal for 16, 32 and 64 processors.



(a) Additive-Schwatz



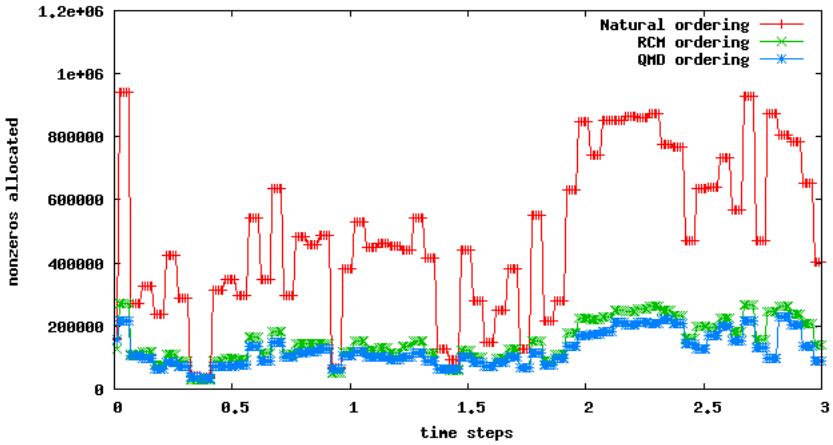
(b) Block-Jacobi

Figure 2: Parallel Scalability

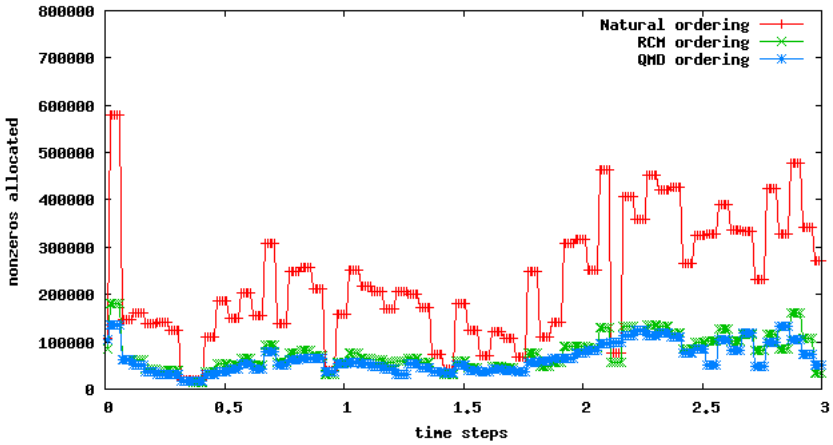
Finally, in Figure 3, we show the number of nonzeros allocated for ILU(1) preconditioner for all orderings and 64 processors. As we expected, the number of allocated nonzeros is much larger for natural ordering than RCM and QMD orderings for both Additive Schwarz and Block-Jacobi. Figure 3 also illustrates the dynamic nature of the AMR/C processes.

5. CONCLUDING REMARKS

In this work, we have compared the behavior of Block ILU preconditioners in Krylov solvers for parallel solution of sparse algebraic systems arising from evolution problems. Of particular interest was the iterative solver behavior when adaptive mesh refinement and coarsening (AMR/C) was also utilized. Moreover, reordering strategies were applied to local subdomains for ILU preconditioning. We considered a scalar field problem where the transient



(a) ASM



(b) Block-Jacobi

Figure 3: Nonzero allocated for the sphere problem using ILU(1) preconditioner and 64 processors.

convection-diffusion equation was discretized using SUPG stabilized finite element formulation. Block-Jacobi and Additive Schwarz, two domain decomposition techniques, were used to build approximate preconditioners on each subdomains. The results have shown that Block-Jacobi presented lower computational cost when the number of processors increases. Furthermore, local ILU was sensitive to the unknowns ordering in terms of CPU time. Comparing orderings schemes,the RCM ordering has better performance.

Acknowledgements

Computer time at the SGI ICE 8200 system was provided by NACAD/COPPE/UFRJ. We acknowledge support from CNPQ, ANP and Petrobras. Prof. G.F. Carey is partially supported by DOE. This work is developed with the cooperation agreement between ICES,UT Austin and COPPE.

REFERENCES

B. S. Kirk, J. W. Peterson, R. H. S. & Carey, G. F., 2006. libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Journal Engineering with Computers*, vol. 22, pp. 237–254.

Brooks, A. & Hughes, T., 1982. Streamline Upwind/Petrov-Galerkin formulations for convec-

tion dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. and Engrg.*, vol. 32, pp. 199–259.

Catabriga, L., Valli, A., Melotti, B., Pessoa, L., & Coutinho, A., 2006. Performance of lcd iterative method in the finite element and finite difference solution of convection-diffusion equations. *Communications in Numerical Methods in Engineering (in press)*, vol. 22, pp. 643–656.

Codina, R., Onate, E., & Cervera, M., 1992. The intrinsic time for the streamline upwind - PG formulation using quadratic elements. *Comput. Methods Appl. Mech. and Engrg.*, vol. 94, pp. 239–262.

Dai, Y. & Yuan, J. Y., 2004. Study on semi-conjugate direction methods for non-symmetric systems. *International Journal for Numerical Methods in Engineering*, vol. 60, pp. 1383–1399.

Elias, R. & Coutinho, A., 2007. Stabilized edge-based finite element simulation of free-surface flows. *Int. J. Numer. Meth. Fluids*, vol. 54, pp. 965–993.

Gropp, W. D., Kaushik, D. K., Keyes, D. E., & Smith, B. F., 2001. High-perfomance parallel implicit cfd. *Parallel Computing*, vol. 27, pp. 337–362.

Hestenes, M. & Stiefel, E., 1952. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, vol. 49, n. 6, pp. 409–436.

Kalro, V. & Tezduyar, T., 1998. Parallel iterative computational methods for 3d finite element flow simulations. *Computer Assisted Mechanics and Engineering Sciences*, vol. 5, pp. 173–183.

LeVeque, R. J., 1996. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Numer. Anal.*, vol. 33, pp. 627–665.

Osher, S. & Fedkiw, R., 2000. *Level set methods and dynamic implicit surfaces*, volume 153. Springer.

PETSc, visited 08-02-2009. http://www-unix.mcs.anl.gov/petsc/petsc-2/. vol. .

Saad, Y. & Schultz, M. H., 1986. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, vol. 7, n. 3, pp. 856–869.

Saad, Y. & van der Vorst, H. A., 2000. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, vol. 123, n. 1-2, pp. 1 – 33.

Silva, R. S., Almeida, R. C., Galeao, A. C. N., & Coutinho, A., 1997. Iterative local solvers for distributed krylov-schwarz method applied to convection-diffusion problems. *Comput. Methods Appl. Mech. Engrg.*, vol. 149, n. 1-4, pp. 53–362.

Trilinos, visited 03-06-2009. Trilinos home page. http://trilinos.sandia.gov/resources.html.

van der Vorst, H., 1986. A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, vol. 13, pp. 631–644.