HIGH-PERFORMANCE PROGRAMMING FOR COMPUTATIONAL WIND ENGINEERING APPLICATIONS

Guilherme Luiz Piccoli, glpiccoli@mecanica.ufrgs.br
Elizaldo Domingues dos Santos, edsantos@mecanica.ufrgs.br
Departament of Mechanical Engineering
Universidade Federal do Rio Grande do Sul – Sarmento Leite Street, 425 – Porto Alegre – RS.

Adriane Prisco Petry, adrianep@mecanica.ufrgs.br
Departament of Mechanical Engineering
Universidade Federal do Rio Grande do Sul – Sarmento Leite Street, 425 – Porto Alegre – RS.

Abstract *The present work aims to improve the computational performance in an existent computational fluid dynamic algorithm in order to evaluate wind engineering applications. To perform this study, techniques of parallel computation and high-performance programming are used. Thus, these improvements in the computational performance will allow the addition of specific terms in order to refine the wind behavior characterization. These modifications claim to reduce the computational costs and consequently, make possible the application in wind energy projects, which commonly use large domains and demand high processing time. The numerical method employed is the Finite Element. In the turbulent flow analysis, it is used Large Eddy Simulation (LES) which solve the scales that govern the local flow dynamic (large eddies) and use sub-grid models to solve those with universal character (small eddies). In this work, the sub-grid effects are analyzed by the standard Smagorinsky model. Simulations over bluff bodies commonly employed in the turbulent flow characterization are used to validate the results and evaluate the phenomena. These elements utilization is chosen due to the complex features derived by the impact between the incident flow and the obstacle which resulting in phenomena as impingement, separation, reattachment, circulation and vortices formation.*

Keywords*: CWE, LES, Bluff Body, High-performance programming, Parallel Computation*

1. INTRODUCTION

In a wind power evaluation, an accurate wind motion characterization over terrains will allow an appropriate site selection for a wind farm installation and consequently, the best wind turbines layout. Thus, solutions based on the Computational Fluid Dynamics have been constantly employed in wind power projects in order to describe the related phenomena with high level of refinement. In this work, the flow over a topographic feature is obtained using the Large Eddy Simulation (LES) and the Finite Element Method (FEM), which present excellent results for many wind engineering applications.

In the most of wind farm projects, it is observed statistical analyses to describe the speed behavior over terrains. An accurate prediction of this term is extremely important, once the wind power and the output energy are proportional to its cube. However, this methodology has the drawback to evaluate a micrositting potential based on local measurements of wind direction and speed. According to Derickson *et al.* (2004), as a consequence of this current analysis, there are numerous wind turbine installations that are buffeted by damaging turbulence or are faced with suboptimal wind energy performance. In order to obtain more realistic results, numerical simulations over terrains - based on the Computational Fluid Dynamic (CFD) - have been examined by many researchers and new techniques are constantly developed.

Therefore, one of the most promising techniques in the turbulent flows estimative (especially across bluff bodies) is the numerical approach. However, a high computational effort is demanded due to complexity of the many different interacting scales. According to Silveira Neto (2002) the rate between the large scales and the small scales is proportional to Reynolds number, being for each direction approximately $Re^{3/4}$. How turbulence is naturally three-dimensional, the direct solution of all scales in an isothermal turbulent flow demands a grid proportional of $Re^{9/4}$. Even when turbulence is modeled by LES, the computational effort is huge. Therefore, methodologies to minimize the time processing of simulations are very important in the knowledge of the phenomena, mainly when is desired the analysis of engineering practices applications. Thereby, one of the objectives here is the resources development to become possible the future analysis of external flows, e.g., wind speed prediction over complex terrains in order to obtain an accurate wind turbines layout.

Nevertheless, in order to distinguish the usual CFD applications to the simulations around bluff bodies and topographic features, a new study field was created in the middle of the 1980's: the Computational Wind Engineering (CWE). According to Cook (1986), a body is aerodynamically bluff when the flow streamlines do not follow the surface of the body, but detach from it leaving regions of separated flow and wide trailing vortices. The CWE is a brand-new area and present applications in distinct fields like analysis of diffusion and dispersion, pedestrian comfort, wind characteristics over complex geometries, regional climate and interactive of fluid structure. According to Murakami (1997), two important reasons could define the main difficulties observed in the wind flow characterization.

First, the topographic features are placed within the surface boundary layer. In external flows, roughness elements like: grass, trees, buildings, rocks and hills are present in the Atmospheric Boundary Layer (ABL) and may affect strongly the wind behavior. The numerical prediction of flow over complex terrain has become an important computational tool in the wind engineering applications. Second, the obstacles commonly evaluated, so-called bluff bodies, are non-aerodynamics and demand high level of refinement in their corners.

In wind power studies using the CWE methodology, the computational costs have been featured the major encumbrance to make viable this powerful instrument in the wind characterization. The necessity of large computational domains to describe a micrositing (on the order of kilometers) and the discretization related to the wide range of scales presents in a real turbulent wind flow increase substantially the CPU time required.

2. METHODOLOGY

In the present work, an existent Computational Fluid Dynamic algorithm is speeded up in order to make viable wind engineering applications. Until the present moment, the computational code studied had its configuration faced to Vector Processor Architectures to solve the fluid mechanics equations. Based on high-performance programming and parallel computation techniques, this study aims to propose some modifications in the algorithm construction directed to reduce the CPU time required.

In the discussion about the wind flow over a complex terrain, a bluff body is analyzed. This kind of obstacle is commonly used in wind engineering evaluations due to the complex effects derived by the impact between the incident flow and the obstacle. Phenomena as impingement, separation, reattachment, circulation and vortices formation are observed near the body.

2.1. Mathematical formulation

To simulate isothermal flows in turbulent regime or at high Reynolds numbers is necessary to solve the conservation equations of mass and momentum, with the boundary and initial conditions, which will be defined in the problem description. The approach used in the present work is LES and, according to Findikakis and Street (1982), a spatial filtering process is needed. Therefore, the equations can be written in the following way:

$$\frac{\partial \overline{P}}{\partial t} + C^2 \frac{\partial}{\partial x_j}\left(\rho_0 \overline{v}_j\right) = 0 \qquad \text{(j = 1,2 and 3) in t x } \Omega \text{ (1)}$$

$$\frac{\partial}{\partial t}\left(\rho_0 \overline{v}_i\right) + \frac{\partial}{\partial x_j}\left(\rho_0 \overline{v}_i \overline{v}_j\right) + \frac{\partial P}{\partial x_j}\delta_{ij} - \frac{\partial}{\partial x_j}\left\{ (\upsilon + \upsilon_t)\left(\frac{\partial}{\partial x_j}\left(\rho_0 \overline{v}_i\right) + \frac{\partial}{\partial x_i}\left(\rho_0 \overline{v}_j\right) \right) + \frac{\lambda}{\rho_0}\left(\frac{\partial}{\partial x_k}\left(\rho_0 \overline{v}_k\right) \right)\delta_{ij} \right\} = 0$$

$$\text{(i,j,k = 1,2 and 3) in t x } \Omega \text{ (2)}$$

where: $\overline{(\ )}$ - large scales; C - sound propagation speed (m/s); μ – dynamical viscosity (kg/ms); λ – volumetric viscosity (kg/ms); υ – kinematic viscosity (m²/s); υ_t –kinematic eddy viscosity (m²/s); v_i – velocity in *i* direction (m/s); x_i – spatial coordinate in *i* direction (m); P - pressure (N/m²); δ_{ij} – Kronecker delta; Ω – spatial domain; t - time domain (s). The filtering process is performed using a box filter.

In the LES methodology used in the present algorithm, the scales that govern the local flow dynamic (large eddies) are solved exactly by the Navier-Stokes equations while the small scales are modeling. According to Silveira Neto (2002), the smaller ones are not explicitly simulated since their behavior present characteristics more dissipative, isotropic, short-lived, homogeneous, universal and less affected by the boundary conditions. The small scales contribution over the solved ones is obtained through the sub-grid models. The sub-grid model more disseminated, which is used in the present work, is the standard Smagorinsky. Its consolidation is due to the simple implementation and well designed formulation.

2.2. Numerical model

To obtain the approximated solution of the conservation equations (mass and momentum) is used the Finite Element Method. The Galerkin Method is employed in the spatial discretization, Reddy and Gartling (1994). To treat the transient terms is implemented the Taylor-Galerkin explicit scheme, Donea (1984). Where the variations of primary variables (velocity and pressure) are generated by their expansion in Taylor series until second terms. According to Brasil Júnior (2002), this temporal scheme has good behavior in strongly advective flows and is used also like stabilization scheme. The approximated equations are adequately present in the work of Petry and Awruch (2006).

Therefore, the solution is conditionally stable and it is necessary to attempt the stability Courant condition, given by:

$$\Delta t \leq \Delta x_i(\min)/(C+U_\infty) \tag{3}$$

where Δt is the time step, Δx_i (min) is the smallest grid element dimension and U_∞ is a velocity reference

2.3. Techniques of numerical optimizations

The optimizations evaluated in the present work befall in two kinds of codes: mono-processing and parallel processing with shared memory. Thus, in the former algorithms, it is analyzed the matrix allocations in order to verify the best manner to manipulate the memory. While, during the parallel processing codes optimizations, it is analyzed the main tasks which may be distributed among many processors. The language employed to parallelize the present code is *OpenMP*.

3. PROBLEM DESCRIPTION

The present study intents to evaluate the flow over a two-dimensional square cross-section bluff body and reveal important features about this kind of problem. In order to compare the obtained results in this work with other authors, the chosen domain respect the dimensions commonly used in this kind of analysis. These geometrical lengths do not cause influence in the flow behavior, allowing a good phenomenon observation. The domain dimensions are scaled with the obstacle dimension *d*.

In Figure 1 can be observed the dimensions of the flow domain: upstream and downstream bluff body distances, x_u, and x_d, respectively, the domain spanwise size, *H* and the obstacle length *d*. The dimensions x_u, x_d, *d*, and *H* are 4.5, 18.5, 1 and 8, respectively. The constant free stream velocity which reaches the obstacle is denominated U_∞. This parameter, together with the characteristic length *d*, allows the Reynolds number definition as $Re_d = U_\infty d/\nu$.

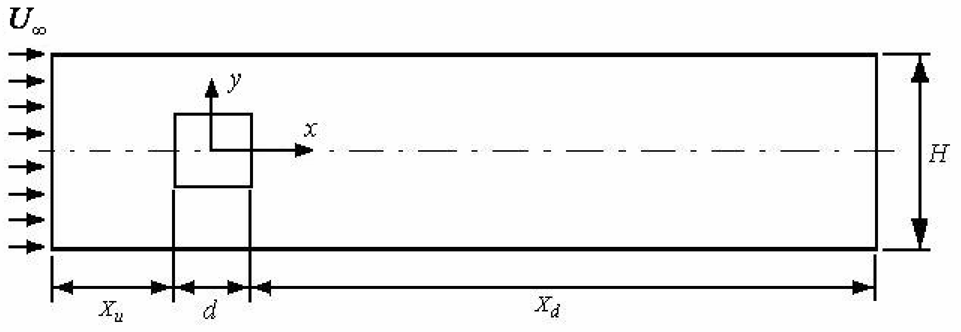

Figure 1. Domain configuration

In the boundary conditions, the following impositions are employed. At the inlet surface, a uniform flow is prescribed for the normal velocity component and at the outlet region, the outflow boundary condition is applied for *x*- and *y*- components. At the lateral boundaries (parallel to streamwise direction), a frictionless condition is simulated by the utilization of a null velocity component at *y*-direction. At last, no slip condition is applied for the square obstacle walls. The initial conditions are zero for velocities components and pressure for all domain.

With reference to domain discretization, many grid resolutions are employed with the purpose of compare the processing time consumed by each one. Afterwards, some of these configurations are used to obtain the physical results, allowing a pertinent discussion about good programming practices in numerical estimative of flows. In Table 1 are described the meshes used during the work.

Table 1. Grid sizes used in the processing time and numerical evaluation.

Grid Number	*Number of Elements (X x Y)*
1	96 x 32
2	144 x 48
3	192 x 64
4	240 x 80
5	336 x 112
6	384 x 128
7	480 x 160
8	576 x 192

The best grid distribution is designed to be refined near the obstacle and coarser farther from this element. This technique of grid design is called grid stretching, see Figure 2 (a). According to Murakami (1998), a large ratio of grid stretching leads to numerical oscillations. In LES methodology, the cut-off wave number of energy spectrum between resolved scale and sub-grid scale is related to the grid size $\overline{\Delta}$. In this case, if the stretching ratio becomes large, the cut-off number differs greatly between two neighboring grids, creating turbulent energy levels significantly different. So, when the LES methodology is used, this same author recommends the stretching grid utilization with caution. According to the same author, the numerical oscillations can be observed also at small aspect ratios higher than 1.05. Therefore, a uniform grid configuration is used to avoid this kind of difficulty, Figure 2 (b).

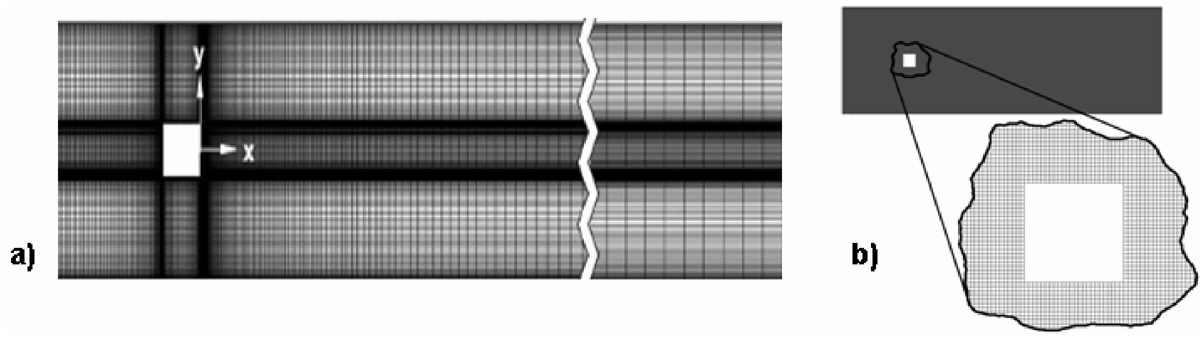

Figure 2. Mesh configuration: (a) stretching design proposed by Kim *et al* (2004) and (b) a uniform grid size of 576 x 192 used in the present work.

4. RESULTS AND DISCUSSION

The obtained results in the present work are distributed in distinct sections. Firstly, it is shown a discussion related to the efforts done in order to reduce the processing time of the code. Thus, high-performance programming alternatives based on the allocation of matrix and vectors are proposed. Afterwards, an evaluation of the parallel computation concept is introduced by the utilization of the *OpenMP* method. The main advantages in this technique utilization are: the possibility to keep a serial code intact, simplicity during the codes programming and dynamic distribution of tasks. According Piller and Nobile (2002), the CFD community is oriented in toward the development of distributed-memory *MPI* applications, but shared-memory *OpenMP* language can be very useful in many circumstances, mainly in algorithms that do not have a pre-conceived design language to receive the parallel implementation.

In the section designated to analyze the flowfield over a square cross-section bluff body, velocities profiles are monitored in the obstacle neighborhood.

4.1. Time processing evaluation

The numerical simulations were performed at the National Center of Supercomputation (CESUP-RS) using computers with Parallel Processor Architectures, so-called Clusters. The parallel computer configuration is a SunFire X2200 with 5 nodes/servers and 1 node retained for management. Each node has 2 dual-core AMD Opteron processors with 1.8 GHz clock and the memory/node amount is 8GB. Therefore, it is possible to execute simulations using shared-memory and distributed-memory parallel processing. By the fact that the Cluster resources are accessed by many users (university departments, research projects and computational courses) it is necessary evaluate the CPU time without external interference, in other words, the simulations are done using the whole node memory available in order to get accurate comparative tests.

The first study realized consists in perform a comparative between the static and dynamic memory allocation. In the former, each array is set at compilation time and never changes and the latter one, the program selects dynamically memory spaces during the program execution. The grid size selection is obtained based on some mesh sizes commonly used in this kind of approach by previous authors: Petry and Awruch (2006), Sohankar (1999) and Santos *et al.* (2007).

As can be seen in Figure 3, it is shown the CPU time required for a single record using both allocation forms. The simulations were evaluated using a time step of 1.1E-05 seconds and 500 iterations for each recording. Comparing with the dynamic allocation, the static one exhibited a better behavior in all meshes studied. This behavior can be attributed to the number of times that the memory is accessed in order to reserve spaces to receive the matrix and vectors components. It is important remember that more analysis points could be required to monitoring with higher accuracy the CPU time behavior. Thus, when the static allocation is chosen it is observed an average reduction of approximately 1.5 times in the processing time. For the studied algorithm using static and dynamic allocation, the grid increase identified a discontinuity between the configurations 4 and 6 for the both situations. This behavior might be related to the memory manipulation due to the program design language. Based on the results introduced in this evaluation, new tests will be performed in order to investigate the origin of this non-linear conduct.

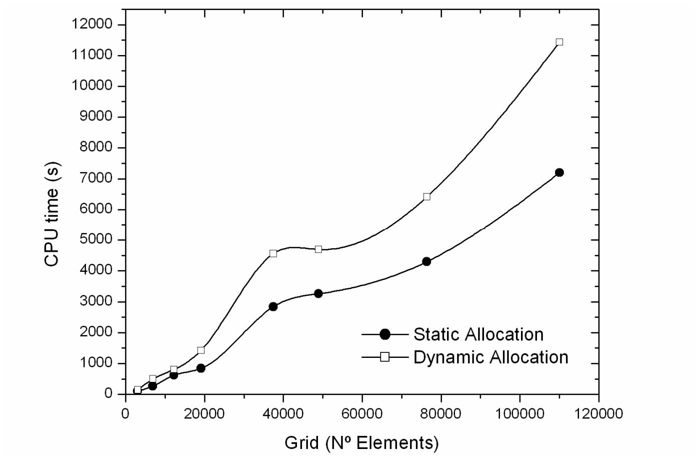

Figure 3 – CPU time for dynamic and static allocation for a mono-processing processor

After the analysis of performance improvement founded on the arrays allocation, the insertion of parallelization directives is evaluated using the *OpenMP* concept. This approach allows that the main program access the shared memory. In other words, all cores will be able to aid the code to solve the problem. With the intention to use correctly this tool, it is analyzed different sections of the algorithm aiming the consumed time stratification and consequently, to identify those which represent a process bottleneck. In Table 2, it is observed a result that has come from this verification, the case tested is grid 01 with 500 iterations. Thereby, as can be confirmed in this same table, great part of the computational efforts are designated to the portion that manipulate the main matrixes per iteration used on the solution of turbulent, incompressible and isothermal flows. To verify if this behavior is global, in other words, not specific to one grid simulated, a Pareto chart with 4 different grids is proposed, Fig. 4. The analysis of the chart indicates that in other grids the time consumption is distributed in a similar manner.

Table 2. Sections created to evaluate the parts of the program that need parallel process.

Part of Program	Name of program part	CPU time (%) - Grid 01	Cumulative CPU time (%) – Grid 01
1	ITERATION MATRIX ATUALIZATIONS	75.79%	75.79%
2	TIME MATRIX ATUALIZATION	9.44%	85.23%
3	ASSEMBLY MATRIX ATUALIZATION	4.58%	89.81%
4	ASSEMBLY ITERATION MATRIX	3.48%	93.29%
5	VARIABLES VARIATION CALCULUS	2.75%	96.04%
6	TURBULENT VISCOSITY CALCULUS	2.29%	98.33%
7	OTHERS	1,67%	100%

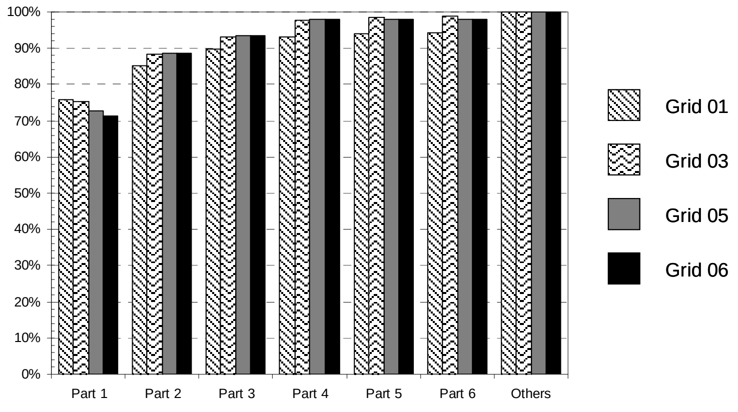

Figure 4. Pareto Chart used during the time consumption analysis for 4 grids.

The results indicate that the first step to improve the code time processing is the implementation of *OpenMP* in Part 1 section. According to Figure 5, the *OpenMP* implementation represented an important speed up in the original computational fluid dynamic code. Based on the parallelization of the most costly program segment, the CPU time exhibited significantly reductions both in dynamic and static allocations. The average reduction observed at Figure 5 is approximately 2.45 times for static allocation with *OpenMP* and 2.68 times for dynamic allocation with *OpenMP*. The best improvement is reported employing the grid configuration 8, which presents a speed up of 3.09 times using static allocation and 3.24 times for dynamic allocation, both comparing without *OpenMP* implementation. For the present computational code, there is a better memory manipulation when the *OpenMP* and static allocation are employed together. When the best and the worst situation are confronted, static allocation with *OpenMP* and dynamic allocation without parallel implementation, respectively, the difference in performance are approximated 3.76 times. This emphasizes the importance of optimizations realized in the present code, mainly when turbulent flows are simulated.

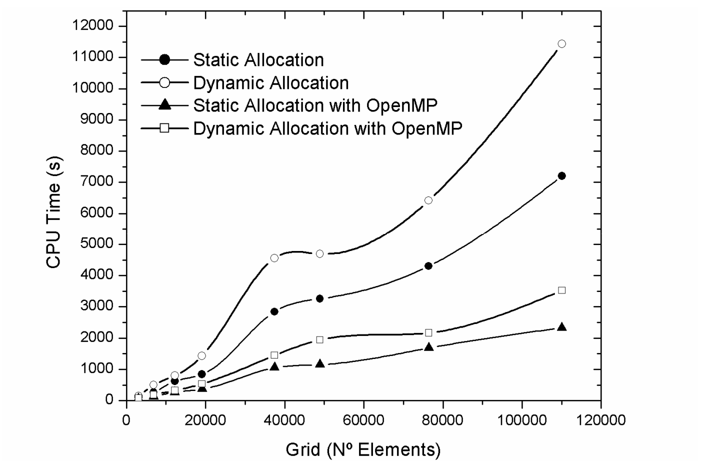

Figure 5. A comparative between a mono-processing code and parallel-processing with 2 dual-core processors for static allocation and dynamic allocation

Afterwards, the *OpenMP* is implemented in the four sections that request more CPU time. According to Table 2, the parts 1, 2, 3 and 4 of the algorithm are parallelized. As the better situation observed in the previous analyses is the utilization of static allocation and *OpenMP*, tests using this configuration are realized. Using 4 processors, the maximum utilization of the code in the cluster is restricted to 400 %. Despite the results do not present an important addition in the CPU time performance, this new situation allows a better utilization of the computational resources, reducing substantially the elapsed time (real-time clock) of the simulations and almost exhausting the improvements provide by the *OpenMP* implementation. Thus, it is observed the increasing of the average CPU utilization from 316% to 395%.

4.2. Flow motion over a bluff body

In this section, the flow motion over a non-aerodynamic obstacle is analyzed aiming future wind engineering applications. The numerical simulation drawback of external wind flows over micrositing is related to the large CPU time required when LES turbulence modelling is used. Thus, simulations over bluff bodies have been an important alternative to improve the phenomenon understanding and validate the codes which are developed.

In this work, an initial study about the velocity behavior in the vicinity of the 2D square cross-section is evaluated. This analysis was developed using a moderate Reynolds number of 3000 and a uniform mesh design. The stretching grid implementation has been also studied, once the uniform one demands a great number of elements and increase greatly the CPU time spent. Among the grids observed in Table 1, the configuration 8 is which presented the best relationship with the physical behavior. In Figure 6, a non-dimensional velocity (u/U_∞) near the bluff body is observed for 0.24 seconds of simulation. The phenomena visualized allow identify trailing vortices in the obstacle downstream and velocity variations during the domain. Thus, the knowledge of these effects may aid the wind farm projects.

In Figure 7, it is shown results at 3 different points in the obstacle neighborhood. Figure 7(a) represents the time- and spanwise-averaged streamwise velocity obtained by a vertical investigation starting from the square center and following the positive *y*- direction. The same process is executed for the Figures 7(b) and 7(c), which are respectively analyzed in the bluff body corner, 0.5 *d*, and in a distance of 1.5 *d* from the obstacle center.

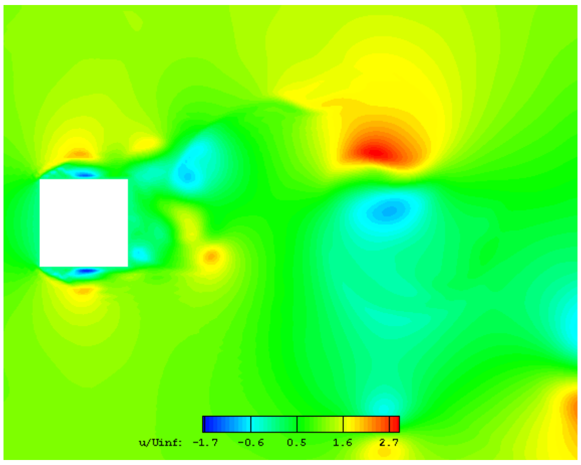

Figure 6. Non-dimensional velocity near the bluff body for 0.24 seconds of simulation.

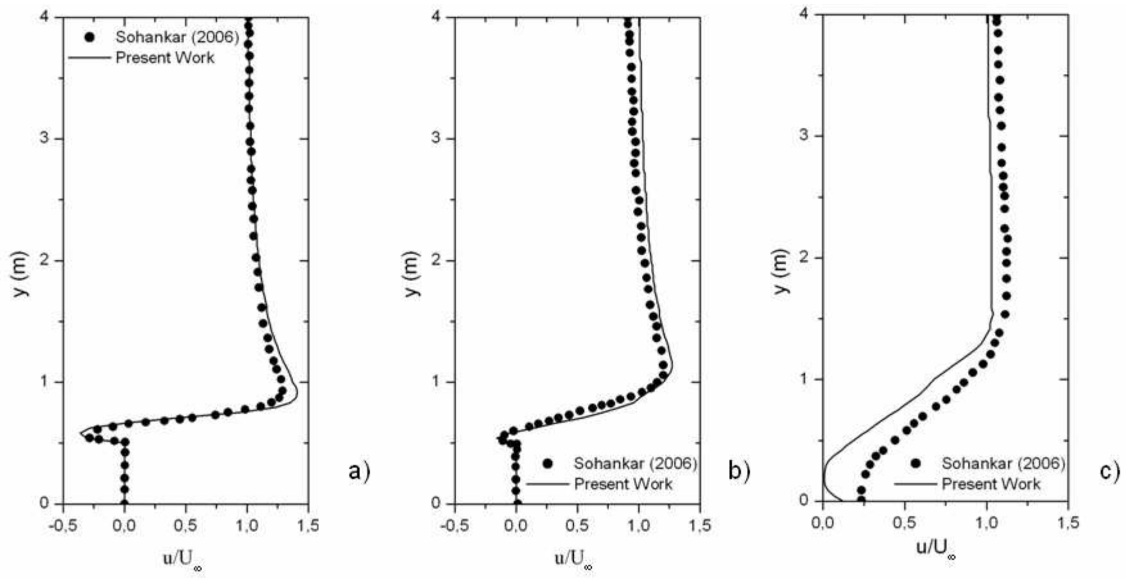

Figure 7. Time- and spanwise-averaged streamwise velocity profiles in the bluff body vicinity at (a) *x = 0*; (b) *x=0.5 d* and (c) *x = 1.5 d*.

The profiles shown in Figures 7(a) and 7(b) present good similarity with the results stated by Sohankar (2006). Effects like the boundary layer separation are captured with accuracy near the obstacle walls, as can be observed. In the region located at bluff body downstream, Figure 7 (c), the results indicate a slight deviation from those observed by the above author. This behavior can be attributed to the simplification employed in this work: the two-dimensional domain utilization. According to the same author, 2D results for mean drag are in reasonable agreement with experiments, although other quantities for $Re \geq 200$ are in contrast with experimental data and 3D results. Other factor that may influence the flow near the obstacle is the turbulence modeling chosen. According to Sohankar (2006), standard Smagorinsky model has the drawback to be incapable of taking into account the length scales reduction near the solid walls and other anisotropic flows. Others disadvantages, which Murakami (1997) ratified in his work, are that the above model presents characteristics absolutely dissipative and its proportionality coefficient (C_S), which is a constant in all over the domain, must be adequately adjusted for analyzing the flowfield.

5. CONCLUSIONS

The Large Eddy Simulation has been figured an important approach in the wind engineering applications. However, until the present moment, this methodology application is almost restricted to classical cases which claim mesh sizes lesser than those needed in a micrositing velocity investigation.

In this way, it is extremely necessary that reduces in the processing time being evaluated in order to guarantee that practicable CPU times could be reached in wind power numerical analyses. Thus, due to the fast CPU hardware advances observed in the last years and the available techniques of parallelization like *OpenMP* and *MPI*, the numerical simulation will gleam favorable trends and become an important computational tool in the wind speed prediction. The results obtained in this work reveal that the implementation of *OpenMP* directive can improve substantially the algorithm performance. In the future works, in order to continue the time reduction process, new parallel methods will be tested. In the flow characterization, a three-dimensional domain and stretching grid configurations will be performed to acquire more refined results.

Problems related to the Smagorinsky turbulence model may be overcame by the dynamic model utilization. Developed by Germano et al (1991), this model uses a proportionally factor which is evaluated during the computational process. Despite this model increase the CPU time required, simulations based on the dynamic concept have been presented better results in wind engineering applications when compared with the standard model used in this study.

6. ACKNOWLEDGEMENTS

The authors would like to thanks to CESUP-RS (National Center of Supercomputation) and Professor Masuero by the technical support.

7. REFERENCES

Brasil Júnior, A.C.P., 2002, “Turbulence – First Volume”, Editors: Freire, A.P.S., Menut, P.P.P and Su, J.,ABCM, Rio de Janeiro, RJ, Brazil, 272 p.

Cook, N. J., 1986, “The designers guide to wind loading on building structures: part 1”, Butterworths, London, England.

Derickson, R. G., McDiarmid, M., Cochran, B. C., Peterka, J. A., 2004. “Resolving Difficult Issues of Wind Power Micrositing in Complex Terrain”, AWEA Global WINDPOWER 2004 Conference, Chicago, USA.

Donea, J., 1984, “A Taylor-Galerkin Method for Convective Transport Problems”, International Journal for Numerical Methods in Engineering, Vol. 20, pp. 101-119.

Findikakis, A.N., Street,R.L., 1982, “Mathematical Description of Turbulent Flows”, Journal of Hydraulics Division, ASCE, Vol. 108, N° HY8, paper 17265, pp. 887-903.

Germano, M., Piomelli, U., Moin, P, Cabot, W.H., 1991, “A dynamic subgrid-scale eddy viscosity model”, Phys. Fluids A3 (7), pp. 1760-1765.

Kim, D., Yang, K., Senda M., 2004, “Large Eddy Simulation of Turbulent Flow Past a Square Cylinder Confined in a Channel”. Computer & Fluids, Vol. 33, pp.81-96.

Murakami, S. 1997, “Current Status and Future Trends in Computational Wind Engineering”, Journal of Wind Engineering and Industrial Aerodynamics, 67-68, pp.3-34.

Murakami, S. 1998, “Overview of Turbulence Models Applied in CWE-1997”, Journal of Wind Engineering and Industrial Aerodynamics, 74-76, pp.1-24.

Petry, A.P., 2002, “Numerical Analysis of Three-dimensional Turbulent Flows Using the Finite Element Method and Large Eddy Simulation” (In Portuguese), Dsc. Thesis, Programa de Pós-Graduação em Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil.

Petry, A.P, Awruch, A.M., 2006, “Large Eddy Simulation of Three-Dimensional Turbulent Flows by the Finite Element Method”, Journal of the Brazilian Society of Mechanical Sciences and Engineering, Vol. 28, pp. 224-232.

Piller, M, Nobile, E, 2002, “Direct Numerical Simulation of Heat Transfer Heat Transfer in a Square Duct”, International Journal of Numerical Methods for Heat and Fluid Flow, Vol. 12, Nº6, pp. 658-686.

Reddy, J.N., Gartling, D.K., 1994, “The Finite Element Method in Heat Transfer and Fluid Dynamics”, CRC, Boca Raton, Florida, USA, 390 p.

Santos, E.D., Petry, A.P., Xavier, C.M., 2007, “Large Eddy Simulation of Incompressible, Three-dimensional and Non-Isothermal Flows in Cavities and Channels” (In Portuguese), 8º Congreso Iberoamericano de Ingenieria Mecánica, Cusco, Perú, < http://www.pucp.edu.pe/congreso/cibim8/pdf/16/16-36.pdf >

Silveira Neto, A., 2002, “Turbulence – First Volume”, Editors: Freire, A.P.S., Menut, P.P.P and Su, J.,ABCM, Rio de Janeiro, RJ, Brazil, 272 p.

Silveira Neto, A., Grand, D., Métais, O., Lesieur, M., 1993, “A Numerical Investigation of the Coherent Vortices in Turbulence Behind a Backward-Facing Step”, Journal of Fluid Mechanics, Vol. 256, pp. 1-25.

Sohankar, A. 2006, “Flow Over a Bluff Body from Moderate to High Reynolds Numbers Using Large Eddy Simulation”, Computer & Fluids, Vol. 35, pp.1154-1168.

8. RESPONSIBILITY NOTICE