Parallelisation of the CFD code of a CFD-NWP coupled system for the simulation of atmospheric flows over complex terrain

F. A. Castro[1,2], C. M. P. Silva Santos[1], J. M. L. M. Palma[1]

[1] CEsA - Centre for Wind Energy and Atmospheric Flows
FEUP - Faculdade de Engenharia da Universidade do Porto
{csantos, jpalma}@fe.up.pt
[2] ISEP - Instituto Superior de Engenharia do Porto
fac@isep.ipp.pt

Abstract. The sequential simulation of atmospheric flows over complex terrain using Computational Fluid Dynamics tools (CFD) leads normally to very large time-consuming runs. With the present day processors only the power available using parallel computers is enough to produce a true prediction using CFD tools, i.e. running the code faster than the evolution of the real weather. In the present work, the parallelisation strategy used to produce the parallel version of the VENTOS® CFD code is shown. A sample of the results included in the present abstract is enough to show the code behaviour as a function of the number of sub-domains, both number and direction along which the domain splitting occurs, and their implications on both the iteration number and code parallel efficiency.

Key words: atmospheric flows, computational fluid dynamics, domain decomposition, micro and mesoscale coupling, wind power prediction

1 Introduction

The sequential simulation of atmospheric flows over complex terrain using Computational Fluid Dynamics tools (CFD) leads normally to very large time-consuming runs, when temporal and spatial descriptions of the flows are needed. These are for example the requirements of the simulations to be used in the Short Term Prediction of the atmospheric flows over complex terrain. The Short Term Prediction means predictions of time periods of 1–3 days, typically, and, in the present context, requires the use of an operational Numerical Weather Prediction (NWP) program coupled to a CFD code for performing a zooming effect over the NWP results, which will produce results with higher accuracy. With the present day processors only the power available using parallel computers is enough to produce a true prediction using CFD tools, i.e. running the code faster than the evolution of the real weather.

In the present work, the parallelisation strategy used to produce the parallel version of the VENTOS® CFD code [1, 2] is presented. This code has been used with success in the site assessment of wind farms and so the natural choice for us to couple with mesoscale codes to produce a Short Term Prediction tool.

In the following sections, we show the fundamental equations being solved (section 2). In section 3 the parallelisation strategy is presented and in section 4 the results are discussed. Conclusions are presented in section 5

2 Mathematical Model

This section covers the fundamental equations, coordinate transformation and the numerical techniques used in the current study. A more complete description of the model can be found in [2].

The continuity (1), the momentum (2), the potential temperature transport (3) and the turbulence model equations (4 and 5) were written in tensor notation for a generic coordinate system.

$$\frac{\partial \left(\rho U_j \beta_k^j\right)}{\partial \xi^j} = 0, \tag{1}$$

$$\frac{\partial \left(\rho J U_i\right)}{\partial t} + \frac{\partial}{\partial \xi^j}\left(\rho U_k U_i \beta_k^j\right) = -\frac{\partial}{\partial \xi^j}\left(P \beta_i^j\right) + \frac{\partial}{\partial \xi^j}\left[\left(\tau_{ki} + \sigma_{ki}\right)\beta_k^j\right], \tag{2}$$

$$\frac{\partial \left(\rho c_p J \theta\right)}{\partial t} + \frac{\partial}{\partial \xi^j}\left(\rho c_p U_k \theta \beta_k^j\right) = \frac{\partial}{\partial \xi^j}\left[\frac{K_\theta}{J}\frac{\partial \theta}{\partial \xi^m}\beta_k^m \beta_k^j\right], \tag{3}$$

where

$$\tau_{ij} = \frac{\mu}{J}\left(\frac{\partial U_i}{\partial \xi^m}\beta_j^m + \frac{\partial U_j}{\partial \xi^m}\beta_i^m\right)$$

and

$$\sigma_{ij} = -\frac{2}{3}\rho k \delta_{ij} + \frac{\mu_t}{J}\left(\frac{\partial U_i}{\partial \xi^m}\beta_j^m + \frac{\partial U_j}{\partial \xi^m}\beta_i^m\right).$$

The eddy viscosity was given by $\mu_t = \rho C_\mu k^2/\epsilon$, where k and ϵ were obtained from

$$\frac{\partial \left(\rho J k\right)}{\partial t} + \frac{\partial}{\partial \xi^j}\left(\rho U_k k \beta_k^j\right) = \frac{\partial}{\partial \xi^j}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial \xi^m}\beta_k^m \beta_k^j\right] + J\left(P_k - \rho\epsilon\right) \tag{4}$$

and

$$\frac{\partial \left(\rho J \epsilon\right)}{\partial t} + \frac{\partial}{\partial \xi^j}\left(\rho U_k \epsilon \beta_k^j\right) = \frac{\partial}{\partial \xi^j}\left[\frac{1}{J}\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial \xi^m}\beta_k^m \beta_k^j\right] + J\left(\frac{C_1 \epsilon}{k}P_k - \frac{C_2 \rho \epsilon^2}{k}\right), \tag{5}$$

where

$$P_k = \sigma_{ij}\frac{1}{J}\frac{\partial U_i}{\partial \xi^m}\beta_j^m. \tag{6}$$

$C_1 = 1.44$	$C_2 = 1.92$	$C_\mu = 0.033$	$\sigma_k = 1.0$	$\sigma_\epsilon = 1.85$

In the above equations, U, P and θ are the Reynolds averaged velocities, pressure and dry potential temperature, ρ and μ are the air density and molecular viscosity, and k and ϵ are the turbulence kinetic energy and its dissipation rate, whose transport equations (4 and 5) and constants C_1, C_2 and σ_k were set as in [3], whereas the constants C_μ and σ_ϵ followed the recommendations by [4]. The turbulent Prandtl number was chosen equal to unity which implies $K_\theta = \mu_t c_p$.

The coordinate system is defined by transforming a physical Cartesian coordinate system x^i into a computational system ξ^i, where the co-factors are $\beta_k^j = J\partial\xi^j/\partial x^k$ and J is the determinant of the Jacobian matrix of the coordinate transformation (cf., [5]). This transformation makes it relatively simple to treat the boundary conditions and to use a structured mesh, where the physical domain boundaries are the coordinate surfaces following the topography.

The transport equations $(1-5)$ were discretized by finite volume techniques using a central differencing scheme for the diffusive terms. The advective terms were discretized by the hybrid scheme. In the case of the momentum equations (2), an alternative 3rd order truncation scheme was used for the advective terms, identical to the QUICK scheme for non-uniform meshes [6].

The resulting set of coupled algebraic equations was solved using the SIMPLE algorithm of [6] and the Tri-Diagonal Matrix Algorithm solver, to take advantage of the structure of the coefficient matrices. The pressure/velocity coupling in non-staggered meshes was treated following the pressure-weighted interpolation as in [7] and [8].

The equations can be solved in time-dependent mode, using a second order implicit scheme, or in steady state formulation, in which case the time derivatives in equations $(2) - (5)$ were dropped. In either case the equations were solved until mass and momentum could be satisfied to a dimensionless residual below 5×10^{-4}.

The development and validation of the VENTOS® code for a series of atmospheric flows, as well as details on boundary and initial conditions, can be found in [1] and [2].

3 Parallelisation Technique

The algorithm of the CFD code used in this work can be said to contain two iterative levels: an inner iteration where the solution of each equation is iterated a small number of sweeps – to each equation corresponds a different subroutine; and an outer iteration which contains all the inner levels and is responsible for the coupling between equations. The equations for different variables are solved in a segregated manner, one after the other. In the inner iterations, we use the Tri-Diagonal Matrix Algorithm (TDMA) (see, for example, [9]) to solve the various algebraic systems. The Message Passing Interface library (MPI) is used to implement the communication between processes.

The code was parallelised using a domain decomposition strategy, where the physical domain, discretised by a mesh of control volumes with a central node, was decomposed into several sub-domains, each being calculated in a dedicated

Table 1. Speed-up and efficiency results for several partition schemes. Two sets are shown, averaged results until convergence and averaged results at the end of 1000 iterations. Time is displayed in seconds and efficiency in %. Speedup corresponds to the ratio of execution times and NP represents the number of processors.

		Until convergence				Per 1000 iterations		
Partitions	NP	Iterations	Time	Efficiency	Speed-Up	Time	Efficiency	Speed-Up
1×1×1	1	1060	18504	-	-	17456	-	-
2×1×1	2	1129	11666	79.3	1.6	10333	84.5	1.7
1×2×1	2	984	9571	96.7	1.9	9727	89.7	1.8
1×1×2	2	1160	9138	101.3	2.0	7877	110.8	2.2
3×1×1	3	1404	16936	36.4	1.1	12063	48.2	1.5
1×3×1	3	968	7511	82.1	2.5	7759	75.0	2.3
1×1×3	3	1181	7779	79.3	2.4	6587	88.3	2.7
4×1×1	4	1815	15425	30.0	1.2	8499	51.4	2.1
2×2×1	4	1068	6997	66.1	2.6	6552	66.6	2.7
2×1×2	4	1194	5184	89.2	3.6	4341	100.5	4.0
1×4×1	4	979	4827	95.8	3.8	4931	88.5	3.5
1×2×2	4	1115	4834	95.7	3.8	4335	100.7	4.0
5×1×1	5	2302	11004	33.6	1.7	4780	73.0	3.7
1×5×1	5	1087	3108	119.1	5.9	2860	122.1	6.1
6×1×1	6	3620	11610	26.6	1.6	3207	90.7	5.4
3×2×1	6	1293	6120	50.4	3.0	4733	61.5	3.7
3×1×2	6	1271	6878	44.8	2.7	5411	53.8	3.2
2×3×1	6	1085	3860	79.9	4.8	3558	81.8	4.9
1×3×2	6	1055	2507	123.0	7.4	2376	122.4	7.4
4×2×1	8	1677	3741	61.8	4.9	2285	95.5	7.6
2×4×1	8	1114	2815	82.2	6.6	2527	86.3	6.9
2×2×2	8	1148	3500	66.1	5.3	3049	71.6	5.7
3×3×1	9	1316	2484	82.8	7.4	1888	102.8	9.3
5×2×1	10	2089	4989	37.1	3.7	2447	71.4	7.1
2×3×2	12	1103	1932	79.8	9.6	1752	83.1	10.0
5×3×1	15	2116	2863	43.1	6.5	1353	86.0	12.9
3×3×2	18	1181	1381	74.5	13.4	1169	83.0	14.9
2×5×2	20	1101	1094	84.6	16.9	994	87.8	17.6

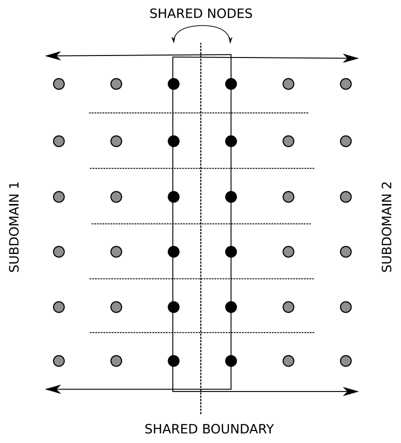

Fig. 1. Representation of overlapped domains

processor. Inside each sub-domain, the code works essentially as its sequential version plus the necessary communications to exchange the boundary information with neighbouring sub-domains. This physical domain decomposition was performed with a fixed overlap of two grid nodes (see figure 1).

Communication between adjoining sub-domains takes place after each sweep of the local TDMA solvers, providing the exchange of the overlapped grid node values. Other communication instances occur at the beginning and end of each of the subroutines, where communication of shared grid node values, momentum fluxes at the control volumes boundaries and some algebraic equation coefficients are exchanged. For convergence checking, global residuals are constructed by collecting local information from the individual sub-domains, which occurs at the end of the outer iteration loop. Similar techniques can be found, for example, in [10].

All computation is thus parallelised: i.e. not only the algebraic solver, but also the routines that construct the coefficient matrices, the reading of external files and the writing of output files, with the exception of minimal output to standard output to monitor the progress of the simulation and the construction of global residuals and fluxes, which are handled by the master process.

4 Results

4.1 Speed-Up and Efficiency

In this section, we investigate the effect of different partitioning schemes on convergence and computational time, as well as the effect of the size of the mesh on the efficiency of the parallelisation. Speed-up and efficiency results were measured from simulations of a real flow of 60° winds (WNW direction) over a future wind farm at Mendoiro/Bustavade located in the North of Portugal. All simulations were carried out in a cluster with 64 nodes using Intel(R) Xeon(TM) CPU 3.00GHz, and all results shown in this section were obtained by averaging over three different simulations performed at different instances, with the aim of removing oscilations in the performance of the cluster.

The results for several runs, using a mesh of $113 \times 77 \times 45$ ($= 391\,545$) grid nodes and different partition schemes, are presented in table 1 and figures 2–3. In table 1, the number of sub-domains used in each of the computational directions is described in the partitions label, the first column. The total number of processors used in the runs, equal to the number of sub-domains, are shown in variable NP. Two sets of results are shown: (i) results obtained for converged simulations; and (ii) results after a fixed number of iterations (1000). Whilst the former is included to indicate the speed-up of real applications, the latter reveals the actual speed-up of the code in terms of parallel efficiency (i.e. how much CPU time is being consumed in communication overheads, etc.). With respect to the results using this mesh, one may observe that the computational efficiency per 1000 iterations is quite high and that it does not degrade with an increasing number of processors, (see blue dashed line in figure 2, which is a linear regression curve with a slope of 0.849).

However, when CPU times are obtained from completed simulations, the efficiency decreases due to the larger number of iterations that some partition schemes require. Partitioning often reduces the convergence rate because of the slight decoupling that is introduced by the domain splitting. From table 1, one can see that, in this case, partitioning in the first direction (i.e. schemes $2\times1\times1$, $3\times1\times1$, etc.), which is longitudinal with respect to the flow, has the effect of increasing the number of iterations until convergence. This is not always the case, and is highly dependent on the specific flow features. For other flows, it is slicing the domain horizontally that has a strong impact on convergence, because quantities vary more quickly in the vertical direction near the ground. What should be retained, however, is that, despite some reduction in the convergence rate, the parallelisation efficiency until convergence is still very significant: the blue dashed line in figure 3, has a slope of 0.702.

A small number of partition schemes produced parallelisation efficiencies slightly in excess of the maximum theoretical value of SPEED-UP = NP, which can be confirmed by inspection of table 1 or figure 2. Since this occurred mainly for the finer mesh and for partition schemes with relatively few subdomains (namely, $\mathrm{NP} = 2, 4, 5, 6, 9$, which means the size of the subdomains is still fairly large), it is thought that it is not related to issues of memory management. Instead, the explanation is likely to be that the sequential runs were performed in worse computational conditions than these parallel runs (i.e. the computer cluster was heavily loaded, there were filesystem delay issues, etc.), which can lead to some exaggeration of the speed-up of these cases, despite the efforts to minimise this by performing three runs per case.

Figures 2–3 also contain results for a mesh with half the nodes in each direction, $57\times38\times23$ ($= 49\,818$) grid nodes. These data are not tabulated here. It can be seen that for such a smaller case, the parallel efficiency decreases consider-

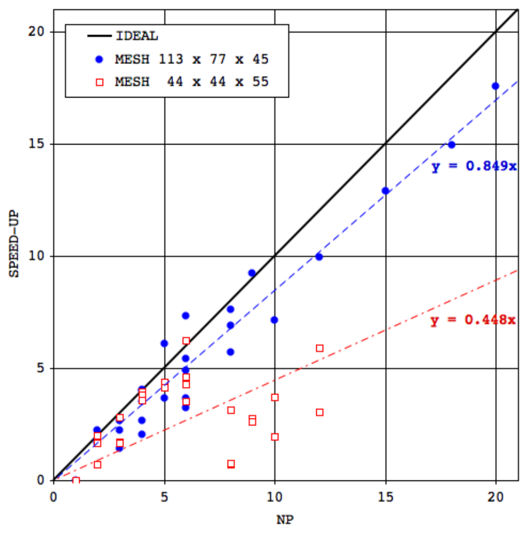

Fig. 2. Speed-up versus number of processors, NP, per 1000 iterations for two mesh sizes.

ably. This is to be expected since the bulk of the computational work performed by the algebraic solver, where most gains are obtained when parallelising, has a much smaller weight in the overall CPU time, when the mesh is small. This is especially true when the number of processes is larger than 6, which means the larger subdomain has less than 8000 grid nodes.

The increase in calculation speed obtained by the present parallelisation strategy was sufficient to produce fast enough calculations, when using for example 9 processors, that enabled us to forecast in 2.5 days a forecast horizon of 5 days.

4.2 Wind Prediction Results

In this section, results of the coupling between the parallel version of VENTOS and the mesoscale code are presented and compared to field measurements. This approach consists of two different simulations, mesoscale and microscale (CFD), with one way coupling linking them. The objective is that the mesoscale model, feed but planetary weather simulations, provides an accurate numerical weather prediction for a large (130×130 km) area, encompassing the wind farm. The CFD code then uses the mesoscale results as boundary conditions, bringing additional accuracy due to the higher resolution meshes and more accurate terrain representation.

A test case for this procedure was performed for the flow over the Mendoiro/Bustavade wind farm site. Three different occasions were selected from the year 2006; two winter occasions, 10 to 15 of January and 1 to 5 of December

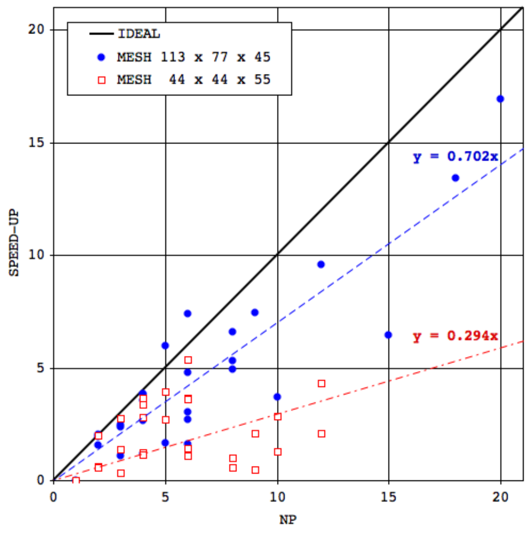

Fig. 3. Speed-up versus number of processors, NP, until convergence for two mesh sizes.

and a Spring/Summer occasion, 1 to 5 of June. The 2006 year was chosen only because of the availability of both experimental and NCEP (National Centers for Environmental Prediction) results, the latter used to drive the WRF simulations (Weather Research Forecast code from the National Center for Atmospheric Research (NCAR) and others, see `http://www.wrf-model.org`).

The VENTOS simulations used a mesh of 39 × 39 × 55 grid nodes with $3 \times 3 \times 1$ partitions, covering a domain of 22×22 km in the horizontal and 7500 m in the vertical. The mesh was almost uniform in the horizontal directions and concentrated near the ground in the vertical direction, where control volumes of 5 m height were used. The mesoscale simulations, using WRF-ARW core (version 2.2.1 and WPS pre-processing system), were done using a mesh of 44 × 44 × 31 grid nodes in a single processor, using a uniform mesh spacing in the horizontal directions of 3000 m and 31 eta levels in the vertical, reaching almost 20 km in height. The time steps of VENTOS and WRF were 2 s and 10 s respectively.

Representations of the horizontal extensions and meshes used in both codes are shown in figures 4 and 5. The zooming effect produced by VENTOS is near 6×, as can be seen in figure 4. From figure 5, it can be seen that VENTOS uses the WRF topography at the boundaries, being then operated a transition to a higher resolution description of the topography. This transition occurs inside a region surrounding the VENTOS domain of 3000 m.

To produce the VENTOS results the WRF simulations were first performed. In the WRF simulations, the results were written to disk every 20 minutes, i.e. every 120 time steps. These results were then interpolated to the boundaries

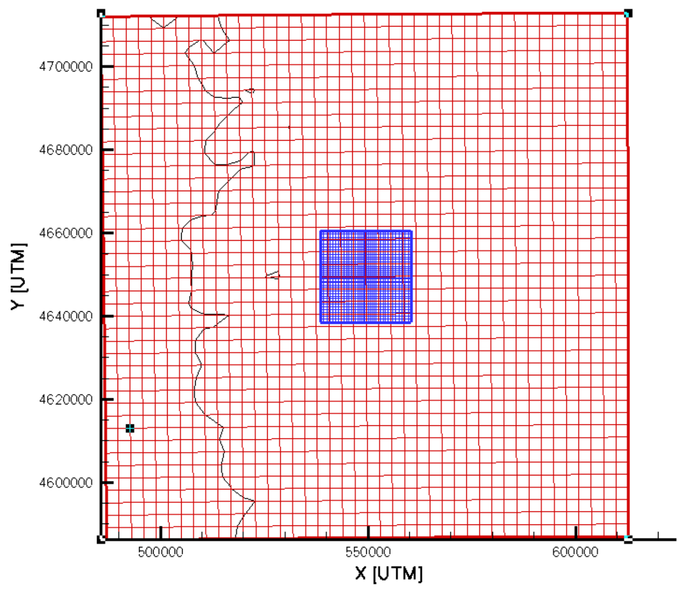

Fig. 4. Meshes used by VENTOS (small extension) and WRF. The black line shows the West Portuguese coast.

of the VENTOS mesh, producing files corresponding to each of the 20 minutes snapshots from WRF. During the VENTOS simulations the boundary conditions were updated every time step using linear interpolations in time between WRF snapshots. All the simulations (VENTOS and WRF) were performed in the aforementioned cluster during normal operation days. The VENTOS simulations took nearly 1 cluster day for every 5 days of real time when the $3 \times 3 \times 1$ partitioning was used, whilst the sequential version would require more cluster days than days of real time.

For the site under study results from three meteorological masts operated with cup anemometers at 60 m above the ground level were available. For this study we present only results for one mast, known as PORT267.

The VENTOS and WRF results for the horizontal velocity magnitude (V_h) are compared with cup anemometer results in figures 6-8. Figure 9 compares the VENTOS and WRF predictions for the potential temperature (θ).

Figure 6 shows the time series of V_h for the warmer occasion under study, 10 to 15 of June. For this period the wind speed was not very height and so thermal effects are more prone the determine the flow behaviour. In this case, the VENTOS results were not a significant improvement over the WRF results.

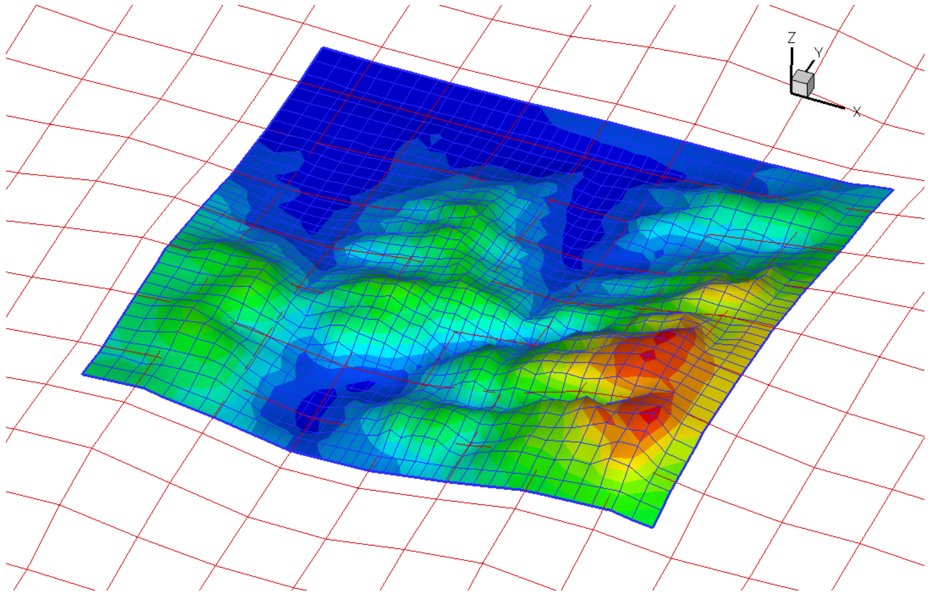

Fig. 5. Meshes and topography used by both codes.

This is partially explained by the simpler thermodynamics and heat-transfer models used in VENTOS.

For the winter occasions, figures 7-8, the mean observed wind speed is higher and the overall improvement introduced by VENTOS is very noticeable. In the windiest period of the study, the first days of December, the improvement was quite significant, when the WRF results were showing wind speeds almost 40% lower than the experimental results.

The potential temperature predicted by VENTOS for December follows very well the WRF results, as can be seen in figure 9. This shows that the thermodynamics and heat-transfer models used in VENTOS are well suited for the application in mind, i.e. the wind power prediction, that deals mainly with the operation of wind turbines for velocities above nearly 5 m/s.

The rms errors in m/s for the 3 periods under study are presented in table 4.2. For the more interesting case, from the wind power point of view, the December period, VENTOS reduced the error at PORT267 1.7 times.

5 Conclusions

The parallel version of the VENTOS CFD code was developed with the aim of producing short term weather prediction. The parallelisation of a CFD code was performed using a domain decomposition strategy, where the physical simulation domain, discretised by a mesh of control volumes with a central node, was decomposed into several sub-domains, each being calculated in a dedicated

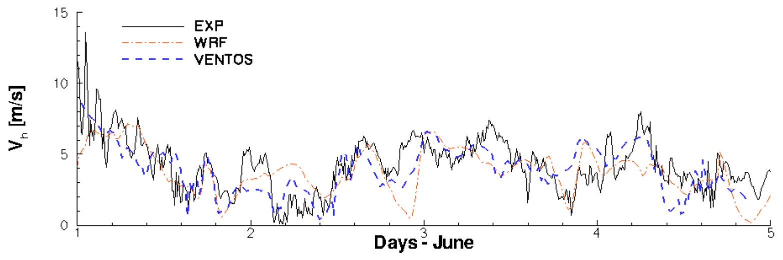

Fig. 6. Time series of the horizontal velocity for the June occasion.

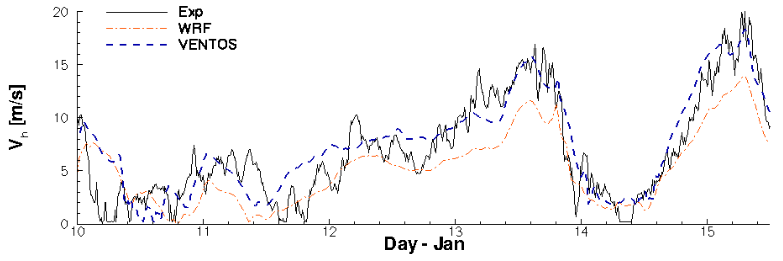

Fig. 7. Time series of the horizontal velocity for the January occasion.

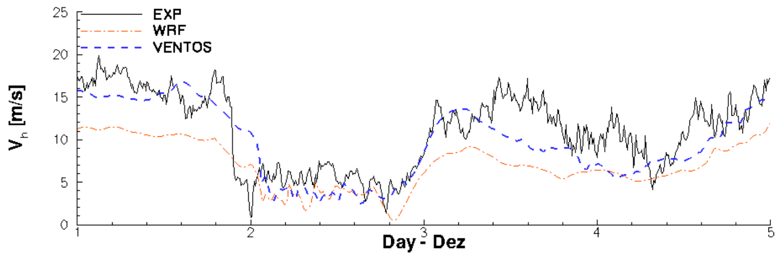

Fig. 8. Time series of the horizontal velocity for the December occasion.

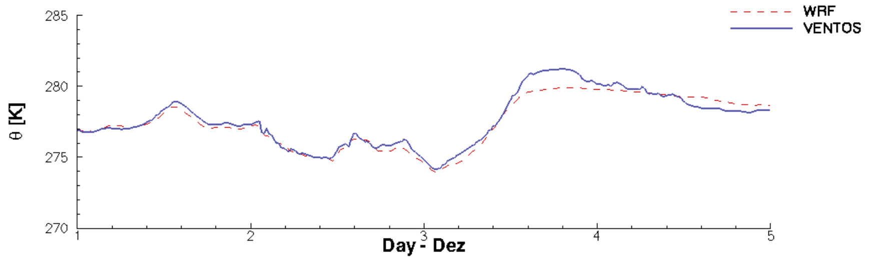

Fig. 9. Time series of the potential temperature for the December occasion.

Table 2. RMS errors in m/s for the predictions at PORT267 for the 3 occasions under study.

Month	VENTOS	WRF
June	1.49	1.59
January	2.17	2.66
December	2.62	4.35

processor. The Message Passing Interface library (MPI) was used to implement the communication between processes. High parallel efficiencies were obtained ($> 80\%$) even for 20 processors. The parallelisation introduced some decoupling between sub-domains which can degrade the converge rate for certain cases. Nevertheless, efficiencies of 70% are still obtained. The adopted strategy and its numerical implementation permitted sufficiently faster execution times to enable true predictions using the current CFD code.

A test case using a real flow over the wind farm of Mendoiro/Bustave, in Portugal, showed that the coupling procedure can improve the mesoscale results. The improvement was more noticeable in windy conditions, the preferred situation for the application in mind, the wind power production. In the better case, a reduction of $1.7\times$ in the error of the mesoscale results were obtained.

Acknowledgements. Carlos Silva Santos would like to ackowledge the support of FCT, Fundação para a Ciência e Tecnologia, through grant SFRH/BPD/18655/2004, co-financed by program POCI 2010 and the European Union.

References

1. Castro, F.A.: Numerical Methods for the Simulation of Atmospheric Flows over Complex Terrain (in Portuguese). PhD thesis, Faculty of Engineering of Porto (1997)
2. Castro, F., Palma, J., Lopes, A.S.: Simulation of the askervein flow. part 1: Reynolds averaged Navier–Stokes equations (k-ε turbulence model). Boundary-Layer Meteorology (2003)
3. Jones, W.P., Launder, B.E.: The prediction of laminarization with a two-equation model of turbulence. International Journal of Heat and Mass Transfer **15** (1972) 301–314
4. Beljaars, A.C.M., Walmsley, J.L., Taylor, P.A.: A mixed spectral finite-difference model for neutrally stratified boundary-layer flow over roughness changes and topography. Boundary-Layer Meteorology **38** (1987) 273–303
5. Knupp, P., Steinberg, S.: Fundamentals of grid generation. CRC Press (1994)
6. Ferziger, J.H., Perić, M.: Computational Methods for Fluid Dynamics. 3rd edn. Springer (2001)
7. Rhie, C.M., Chow, W.L.: Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA Journal **21** (1983) 1525–1532

8. Miller, T.F., Schmidt, F.W.: Use of a pressure-weighted interpolation method for the solution of the incompressible Navier-Stokes equations on a nonstaggered grid system. Numerical Heat Transfer **14** (1988) 212–233
9. Ferziger, J.H., Perić, M.: Computational Methods for Fluid Dynamics. Springer (1996)
10. Durst, F., Schafer, M.: A parallel block-structured multigrid method for the prediction of incompressible flows. Int J Numer Meth Fl **22** (1996) 549–565