Abstract

Refinement technique is used to adapt meshes to the singularities of the numerical solution of any problem. Local refinement is applied is mesh areas where then solution must be accurate and it is necessary to dispose of reliable error indicators or estimators which define the elements that must be refined. Usually, this indicators or estimators are difficult to obtain. On the other hand, derefinement is used to remove mesh elements on which the numerical solution can be easily interpolated. This process can be guided by a parameter that indicates the required precision for the numerical solution. This paper is about of using refinement/derefinement techniques to adapt meshes to the numerical solution but using global refinement instead of local ones. No error estimators or indicators are needed, and two numerical parameters are used to automatize the process.

Keywords: 3-D triangulations, finite element, adaptive meshes, object oriented method, adaptive refinement/derefinement, data structures, wind field models.

1 Introduction

Refinement/derefinement techniques are used when it is necessary to adapt meshes to the singularities of the numerical solution. Usually, local refinement is applied to elements where the solution must be adjusted. Previously, this elements must be determined using any kind of error indicator or estimator. For each mesh element, an error indicator must be computed, and thus it is possible to establish mesh zones where must be carried out the refinement. Depending on the problem, the error indicator or estimator can be more or less difficult to obtain. Moreover, it could be impossible to get a reliable one.

On the other hand, derefinement algorithms removes mesh elements where the numerical solution can be obtained with the desired precision from neighbour elements. Comparing the numerical solution of an element with the interpolated solution, and if it is enough accurate, the element could be removed. Note that in this case we are dealing with numerical parameters that gives us the desired precision, and they are always possible to establish.

In this paper we propose a simple method to carry out meshes adaptation without error indicators or estimators using global refinement. With global refinement all mesh elements divided, so indicators are not needed. After that, a derefinement process will remove elements which are not necessary. Each iteration of this method implies higher computational cost than local refinement, but the total number of iterations of the proposed method could be much less if we not chose a optimal refinement strategy.

In section 2 the algorithm for refinement/derefinement will be briefly presented. Details can be found in [2, 3]. In section 3, a wind field model is commented. It has been used for implementing the algorithm and compare the proposed method with the traditional refinement. This model uses the gradient of the solution as error indicator, and it is detailed in [1, 6]. In section 4 the implementation algorithm is explained, and we present different situations that has been useful to achieve it. In section 5 can be found a problem on which the algorithm has been tested. Besides, a real problem has been implemented with a mesh of the south part of La Palma island. This mesh has been generated with [4]. Finally, in the section 6 a brief comparison between methods is commented.

2 Refinement/Derefinement

The refinement algorithm is based on 8-subtetrahedral subdivision, and it has been presented in [2]. From an initial triangulation τ_0, the goal is to build a sequence of nested meshes $T = \{\tau_0 < \tau_1 < \tau_2 < ... < \tau_m\}$, where τ_{j+1} is obtained from refinement of τj. Each element $t_i \in \tau j$ will have associated an error indicator η_i^j, and it will be refined if:

$$\eta_i^j \geq \gamma \eta_{max}^j \tag{1}$$

η_{max}^j is the maximal value of the error indicator of elements in τ_j. γ is the refinement parameter and $\gamma \in [0, 1]$.

The derefinement algorithm is the inverse of the refinement. It has been presented in [3], and it takes into account the numerical solution in the mesh nodes. Any node $n_i \in \tau_j$ will have computed a numerical solution v_i^j. If we consider n_p and n_q nodes of the surrounding edge of n_i, then n_i can be removed if:

$$\left| v_i^j - \frac{v_p^j + v_q^j}{2} \right| < \epsilon \tag{2}$$

The derefinement parameter ϵ is used to establish the desired precision of the solution in nodes.

3 Wind Field Model: Mass Consistent Model in 3-D

This model [6] is based on the continuity equation for an incompressible flow where the air density is constant in the domain Ω and *no-flow-through* conditions on Γ_b (terrain and top) are considered

$$\vec{\nabla} \cdot \vec{u} = 0 \qquad \text{in } \Omega \tag{3}$$

$$\vec{n} \cdot \vec{u} = 0 \qquad \text{on } \Gamma_b \tag{4}$$

We formulate a least-square problem in Ω with $\vec{u}(\widetilde{u}, \widetilde{v}, \widetilde{w})$ to be adjusted

$$E(\vec{u}) = \int_\Omega \left[\alpha_1^2 \left((\widetilde{u} - u_0)^2 + (\widetilde{v} - v_0)^2\right) + \alpha_2^2 (\widetilde{w} - w_0)^2\right] \, d\Omega \tag{5}$$

where the interpolated wind $\vec{v}_0 = (u_0, v_0, w_0)$ is obtained from experimental measurements, and α_1, α_2 are the Gauss precision moduli.

We consider Dirichlet condition for open or *flow-through* boundaries and Neumann condition for terrain and top

$$\phi = 0 \text{ on } \Gamma_a \tag{6}$$

$$\vec{n} \cdot T \, \vec{\nabla}\mu = -\vec{n} \cdot \vec{v}_0 \text{ on } \Gamma_b \tag{7}$$

This problem can be solved using tetrahedral finite elements (see [5]) which leads to a set of 4×4 elemental matrices and 4×1 elemental vectors. These are assembled to form a symmetric linear system of equations which is solved by a preconditioned conjugate gradient method.

The construction of the interpolated wind is a two step process: horizontal and vertical interpolation. Both processes are detailed in [1, 6].

In the generation of adaptive meshes, the local refinement of the domain is necessary due, on one hand, to the geometry and, on the other hand, to the numerical solution. The computation of error estimators or at least suitable error indicators of the numerical solution is carried out to determine the elements to be refined or derefined in a mesh. In this wind model we use one error indicator, proposed in [1], which takes into account the gradient of the solution in each element.

4 Implementation

To obtain the desired mesh, it will be necessary to compare any mesh parameter. We have chosen the number of nodes, and we define w_n as the number of nodes or τ_n. This parameter is used because it indicates the number of points that will be used for computing the numerical solution.

The first implementation can be seen in the Algorithm 1. It was carried out attending to the ϵ parameter. The stop criteria consist on obtaining a mesh τ_n such $w_n = w_{n-1}$. This implies that all new elements added by refinement of τ_{n-1} have been removed by derefinement, so the numerical solution is adjusted in τ_n according to the ϵ parameter. The Figure 1 represents the expected evolution of w_i in the resulting meshes. This number would increase slower in next steps due to it could have mesh zones with the desired precision in their numerical solution.

Algorithm 1 Initial approach

Be τ_0 the initial mesh
n=0
repeat
  n=n+1
  τ'_n = τ_{n-1} globally refined
  Compute numerical solution of τ'_n
  τ_n = τ'_n derefined according to ϵ parameter
until w_n = w_{n-1}

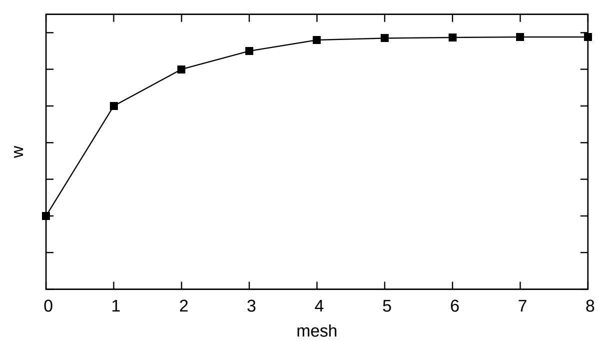


Figure 1: Expected evolution or w

In the first tests the obtained results were good with values of ϵ relatively hight (6 m/s). But with lower values of ϵ (2 m/s) the algorithm does not work properly. In Figure 2 we can see its behavior.

Note that:

1. There are meshes with $w_k < w_{k-1}$. Refinements improve the numerical solution in certain zones and derefinement removes elements introduced many steps before.

2. There are two groups of meshes:

   - $w_{11} = w_{13} = w_{15} = \ldots$
   - $w_{12} = w_{14} = w_{16} = \ldots$

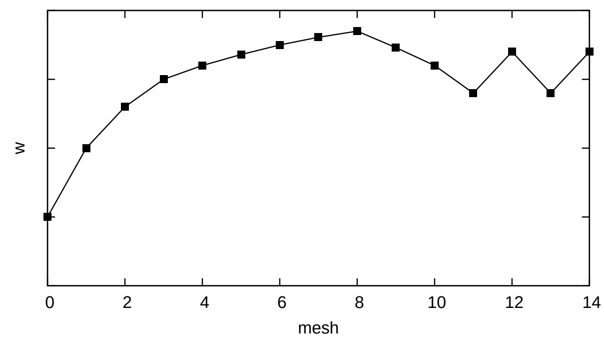


Figure 2: w for test with ϵ = 2 m/s

The second aspect implies that the condition of the Algorithm 1, $w_n = w_{n-1}$, will be never reached. To solve this problem, we have redefined the comparison between meshes. The stop criteria can not be the mentioned before. Instead of that, two meshes will be considered similar if the difference between w_n and w_k ($k < n$) is lesser than a user defined percentage, that is, it will be compared w_n with all the previous w to find two similar meshes.

In final test, with ϵ = 1.5 m/s, we have obtained graphics like Figure 3. This means that the numerical solution will be never adjusted. This problem is a particular case of the wind field model, in elements close the terrain. The difference between the numerical solution of the terrain elements and their adjacent presents low variation with refinements. If new elements are introduced, the difference with this elements is similar than previous, so the numerical solution is not being improved. The derefinement process would not remove that elements, and they will be refined time and again.

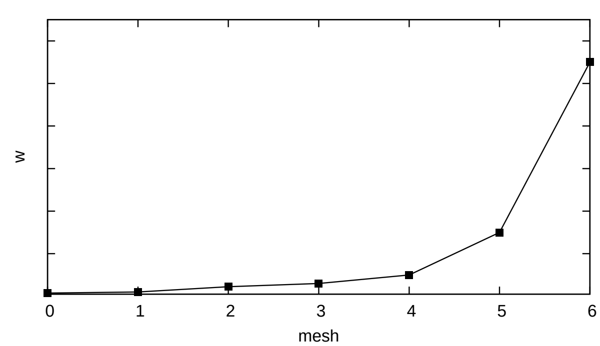


Figure 3: w for test with ϵ = 1.5 m/s

To prevent this, we introduce a new parameter δ. It indicates the minimal size for the edges of any mesh. The derefinement process will be carried out attending to both, ϵ and δ parameters. If an element has any edge lesser than δ, it will be removed. The final implementation can be seen in Algorithm 2.

Algorithm 2 Implemented

Be τ_0 the initial mesh
n=0
loop
  n=n+1
  τ'_n = τ_{n-1} globally refined
  Compute numerical solution of τ'_n
  τ_n = τ'_n derefined according to δ and ϵ parameters
  for i=0 to n-1 **do**
    Exit when $w_n \approx w_i$
  end for
end loop

5 Applications

All the executions were run in a XEON dual processor, with 2 Gb of RAM, under linux and programs compiled with GNU C++. To stop the process we have defined that two meshes are similar if their w_i are different in less than 1%:

$$\frac{w_k}{w_n} \in [0.99, 1.01], k < n \quad (8)$$

The first mesh used is a 3D gauss curve as shown in Figure 4. It consists on 1680 nodes and 7645 elements for a simulated domain of $10000 \times 10000 \times 10000\, m^3$.

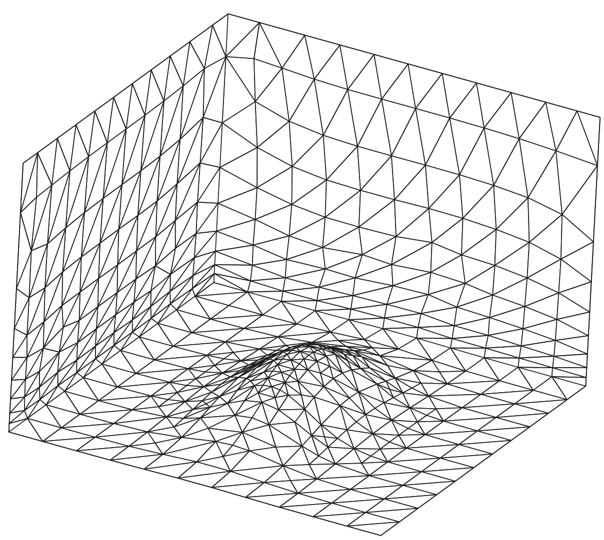
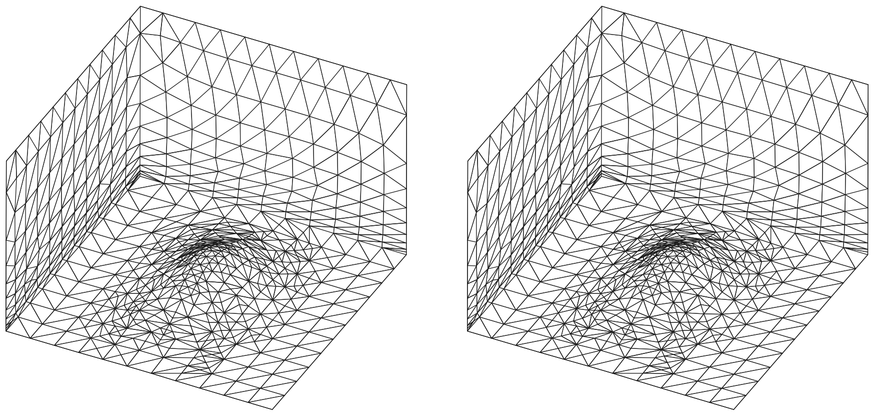


Figure 4: τ_0

The first test was made with parameters ϵ = 2 m/s and δ = 40 m. Five steps were necessary to reach the adjusted mesh (only few meshes are shown in Figure 6(a) and

6(b)). In Figure 1 it can be seen the graphic for w, and in Table 1(b) are printed CPU time for each process.

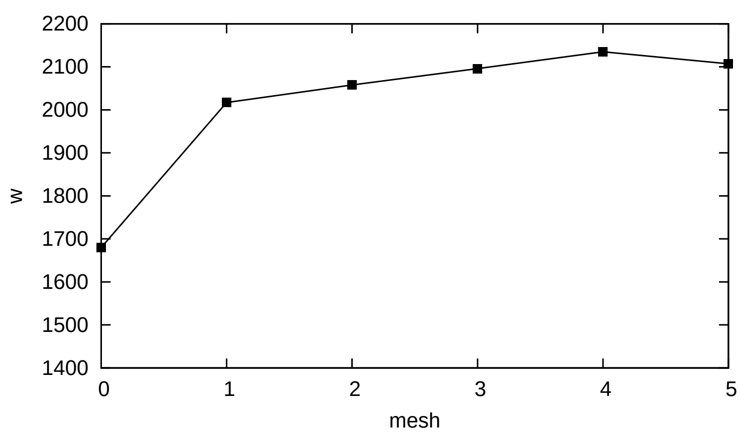
Figure 5: Meshes obtained from τ_0 with ϵ = 2 m/s and δ = 40 m



(a) Mesh τ_3 (b) Mesh τ_5

Table 1: Data for τ_0 with ϵ = 2 m/s and δ = 40 m (Figure 5)

(a) Evolution of w



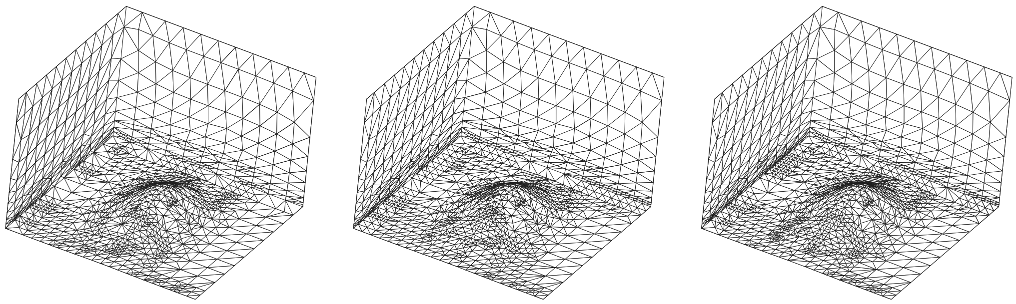
(b) CPU Time in seconds

Initial τ_{n-1}	w_{n-1}	Refine	w_n'	Compute	Derefine	Final τ_n	w_n
τ_0	1680	3.72	11787	5.93	2.68	τ_1	2017
τ_1	2017	5.18	12591	5.31	3.16	τ_2	2058
τ_2	2058	5.39	12865	5.52	3.45	τ_3	2096
τ_3	2096	5.76	13084	5.51	3.83	τ_4	2135
τ_4	2135	6.09	13326	5.98	4.00	τ_5	2107

Another test was made with parameters ϵ = 1.5 m/s and δ = 80 m. For this problem, twelve steps were necessary to adjust the mesh (Figure 6 and Table 2).

In both test the algorithm worked properly, and they were useful to validate and adjust the method.

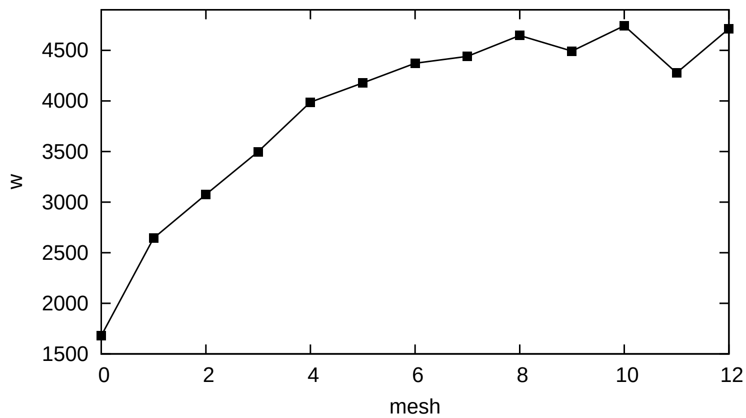
Figure 6: Meshes obtained from τ_0 with ϵ = 1.5 m/s and δ = 80 m



(a) Mesh τ_6 (b) Mesh τ_9 (c) Mesh τ_{12}

Table 2: Data for τ_0 with ϵ = 1.5 m/s and δ = 80 m (Figure 6)

(a) Evolution of w



(b) CPU Time in seconds

Initial τ_{n-1}	w_{n-1}	Refine	w_n'	Compute	Derefine	Final τ_n	w_n
τ_0	1680	3.78	11787	6.01	2.94	τ_1	2644
τ_1	2644	6.84	15857	7.43	4.57	τ_2	3075
τ_2	3075	8.32	18775	9.69	7.05	τ_3	3498
τ_3	3498	9.82	21564	11.67	8.02	τ_4	3987
τ_4	3987	11.73	24532	14.40	8.15	τ_5	4178
τ_5	4178	12.26	25982	16.70	10.46	τ_6	4372
τ_6	4372	13.36	27250	16.85	10.77	τ_7	4440
τ_7	4440	13.96	27597	18.23	11.43	τ_8	4647
τ_8	4647	14.64	29158	20.26	12.35	τ_9	4492
τ_9	4492	14.76	28123	18.52	11.49	τ_{10}	4743
τ_{10}	4743	16.12	29617	18.55	12.43	τ_{11}	4277
τ_{11}	4277	14.60	26783	16.73	11.61	τ_{12}	4713

We have also used a real geometry generated with [4, 5]. It represents part of the south of La Palma island, which can be seen in Figure 7. It consists on 4535 nodes and 21137 elements for a real domain of $45600 \times 31200 \times 6000\ m^3$.

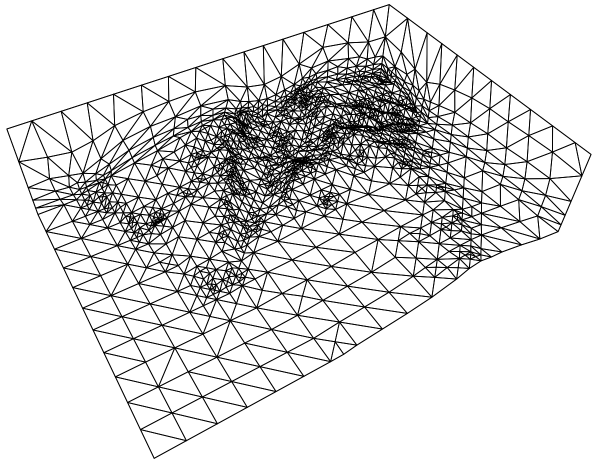


Figure 7: lp_0

The goal was to obtain an adjusted mesh according to ϵ = 4 m/s and δ = 40 m. Only five steps were necessary. In Figure 8 we can see meshes generated and in Table 3 information about the process.

On the other hand, we have used the same mesh lp_0 with the traditional method, that is, local refinement - error estimation - derefinement. The γ parameter for refinement was adjusted to 0.6 and ϵ parameter was equal than the above run (4 m/s). δ parameter was not necessary. Meshes obtained are similar that shown in Figure 8, but in Tables 4 and 5 we can see that number of steps is higher.

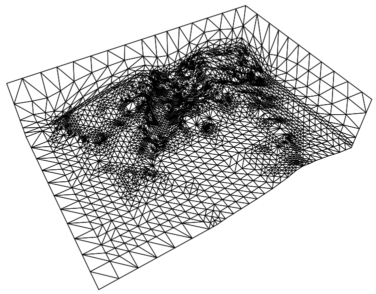
6 Conclusion

To adjust meshes to a numerical solution with local refinement is necessary to establish any error indicator or estimator, and not always is easy. The number of steps depends on both, position and number of singularities of numerical solution. The computational cost in each step is lower than the proposed method due to the number of elements treated.

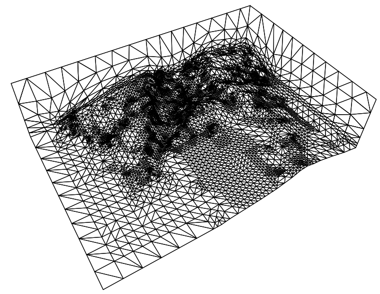
With global refinement all singularities are treated simultaneously. We avoid the error estimation and the method only depends on numerical parameters: ϵ and, in our case, δ. Of course, due to all mesh elements are involved in each step the computational cost is higher, but the number of steps is lower.

If the number of step is very high with local refinement, this method could be advantageous not only for using numeric parameters but CPU time.

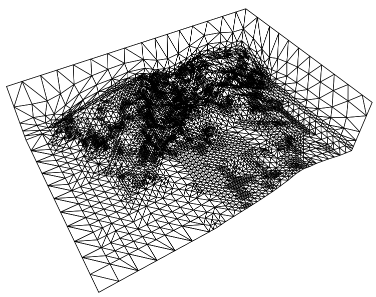
Figure 8: Meshes obtained from lp_0 with ϵ = 4 m/s and δ = 40 m



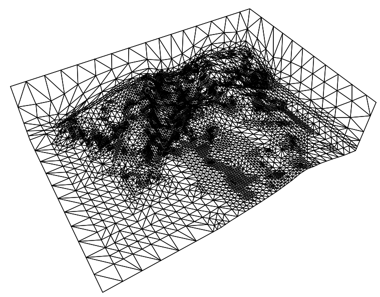
(a) Mesh lp_1



(b) Mesh lp_2



(c) Mesh lp_3



(d) Mesh lp_5

Acknowledgements

This work has been supported by the Spanish Government, “Ministerio de Educacin y Ciencia” and FEDER, grant contacts: CGL2004-06171-C03-02/CLI.

References

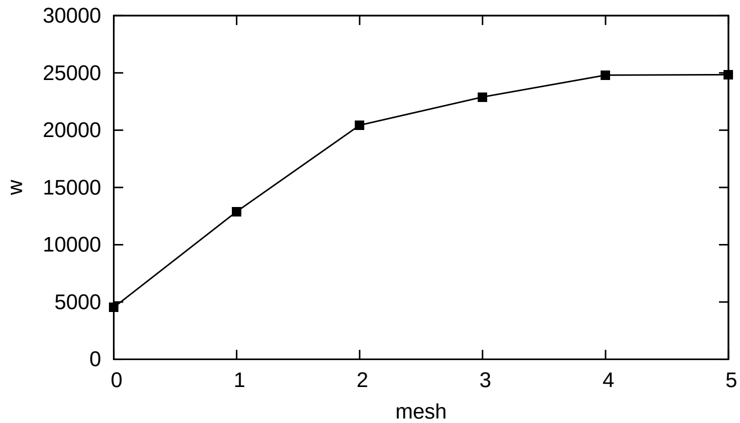
[1] G. Montero, E. Rodríguez, R. Montenegro, J.M. Escobar and J.M. González-Yuste, *“Genetic Algorithm for an Improved Parameter Estimation with Local Refinement of Tetrahedral Meshes in a Wind Model”*, Advances in Engineering Software, 2005; 36:3-10.

[2] J.M. González-Yuste, R. Montenegro, J.M. Escobar, G. Montero and E. Rodríguez, *“Local Refinement of 3-D Triangulations Using Object-Oriented Methods”*, Advances in Engineering Software, 2004; 35:639-702.

[3] J.M. González-Yuste, R. Montenegro, J.M. Escobar, G. Montero and E. Rodríguez, *“Implementation of a Refinement/Derefinement Algorithm for Tetrahedral Meshes”*, Proceedings of the The Fourth International Conference on Engineering Computational Technology, Lisbon 2004.

Table 3: Data for lp_0 with ϵ = 4 m/s and δ = 40 m (Figure 8)

(a) Evolution of w



(b) CPU Time in seconds

Initial lp_{n-1}	w_{n-1}	Refine	$w_n^{'}$	Compute	Derefine	Final lp_n	w_n
lp_0	4535	9.04	32139	23.16	8.37	lp_1	12898
lp_1	12898	29.83	83648	99.84	27.71	lp_2	20426
lp_2	20426	46.58	125989	177.27	42.19	lp_3	22887
lp_3	22887	52.47	139159	211.89	53.05	lp_4	24806
lp_4	24806	57.92	150243	203.95	59.67	lp_5	24845

[4] J.M. Escobar, E. Rodríguez, R. Montenegro, G. Montero and J.M. González-Yuste, *"Simultaneous Untangling and Smoothing of Tetrahedral Meshes"*, Computer Methods in Applied Mechanics and Engineering, 2003; 192:2775-87.

[5] R. Montenegro, G. Montero, J.M. Escobar, E. Rodríguez, and J.M. González-Yuste, *"Tetrahedral Mesh Generation for Environmental Problems over Complex Terrains"*, Lecture Notes in Computer Science, 2002; 2329:335-44.

[6] G. Montero, R. Montenegro and J.M. Escobar, *"A 3-D Model for Wind Field Adjustment"*, Journal of Wind Engineering and Industrial Aerodynamics, 1998; 74-76:249-261

Table 4: Data for lp_0 with γ = 0.6 and ϵ = 4 m/s

Initial lp_{n-1}	w_{n-1}	Refine	$w_n^{'}$	Compute	Derefine	Final lp_n	w_n
lp_0	4535	5.90	18200	10.59	5.03	lp_1	12768
lp_1	12768	19.75	44340	35.01	19.00	lp_2	17593
lp_2	17593	28.38	45780	39.78	19.62	lp_3	18773
lp_3	18773	37.23	42330	33.26	22.35	lp_4	19108
lp_4	19108	38.88	41944	31.12	23.72	lp_5	19304
lp_5	19304	45.69	49265	44.49	27.29	lp_6	19519
lp_6	19519	48.27	52984	50.30	29.66	lp_7	20058
lp_7	20058	56.93	55388	57.96	31.93	lp_8	20294
lp_8	20294	59.61	54401	45.93	32.72	lp_9	20536
lp_9	20536	52.58	51416	49.49	32.04	lp_{10}	20505

Table 5: Evolution of w in adjusting of lp_0 using global (Table 3) and local (Table 4) refinement