

NUMERICAL COUPLING FOR AIR FLOW COMPUTATIONS IN COMPLEX ARCHITECTURES

C. Dobrzynski^{*‡}, O. Pironneau^{*}, and P. Frey^{*‡}

^{*}Laboratoire Jacques Louis Lions
Université P&M Curie, 175 rue du Chevaleret, 75013 Paris, France
e-mail: dobrzyns@ann.jussieu.fr, pironneau@ann.jussieu.fr, frey@ann.jussieu.fr

[‡] INRIA - Gamma
Domaine de Voluceau, BP 105, F-78153 Le Chesnay cedex

Key words: mesh adaptation, Navier-Stokes incompressible, computational fluid dynamics, anisotropic meshes, parallel computing

Abstract. *The numerical simulation of air flow in a building or a house is certainly a very challenging field of engineering applications. The objectives differ from one application to the other, ranging from the optimization of air conditioning systems in buildings to the study of the propagation of harmful products via the air conditioning system. With the recent developments in numerical techniques, it is now possible to carry out, in a reasonable amount of time, a complex three dimensional adaptive simulation based on efficient mesh adaptation algorithms, accurate error estimates, robust and accurate projection methods for solving Navier-Stokes PDE's.*

In this paper, we address the problem of computing the temperature distribution inside a complex geometry featuring various industrial buildings. We present some tools to make the simulations easier and faster. The method is fairly general and applies to other cases as well. Furthermore, we illustrate the natural convection problem on a single room and another example treated here is the heating of the last floor of a two-storeyed furnished house.

1 INTRODUCTION

There are thousands of engineering problems related to air and temperature, from engineering cooling devices to temperature comfort. Some of these problems are relevant to fields such as architecture, which do not normally solve the Navier-Stokes equations for an answer. So air flow and convective heat exchange are modelled by an empirical formula which depends on the size of the windows, the size of the room, the wall material, etc. This method gives a rough estimate of all the energy loss. With the recent developments in numerical techniques, it is now possible to carry out, in a reasonable amount of time, a complex three dimensional adaptive simulation based on efficient mesh adaptation algorithms, accurate error estimates, robust and accurate projection methods for solving Navier-Stokes PDE's.

Flow solvers are making progress especially for three dimensional flows and that is due to a better understanding of turbulence models, a better mastery of the mesh generators and more efficient algorithms for iterative solutions of the pressure equation on parallel processors.

On the modeling side it is well established that at temperatures between -20 and 100 degrees, ambient air is an incompressible Newtonian fluid but its density is a function of the temperature. So the fluid flow is governed by the Navier-Stokes equations for incompressible viscous flows coupled with the temperature equation. As the density varies weakly with the temperature the Boussinesq approximation is sufficient, thus neglecting the variation in density everywhere except in the buoyancy term.

The problem we address here is related to the coupling between the Navier-Stokes equations for an incompressible fluid and a temperature equation with flow convection, via a forcing term. It is also well established that in most cases each equation can be solved independently at each time step (see [1]). Similarly, as the Reynolds numbers for these problems are high, pressure projection algorithms are efficient, giving then an equation for the velocity and a Poisson problem for the pressure at each time step; then the temperature is computed by solving the advection-diffusion equation of energy conservation.

For these incompressible flows the finite volume method is not superior to the finite element methods. The real problem is one of accuracy and multiple scales. For turbulence the k-epsilon model is currently adequate [2] however when computer resources will increase LES will probably be better. For numerical accuracy there is no real choice: adaptivity and if possible non-isotropic mesh elements.

In this paper, we present the equations governing the numerical simulation of air flow and temperature distribution. Then we recall the general mesh adaptation scheme based

on a geometric error estimate and surface and volume meshing algorithms. Finally, results of the numerical simulation on a complex three-dimensional geometry are presented to emphasize the potentiality of the proposed approach.

2 PROBLEM APPROXIMATION

2.1 Incompressible flow model

Let us consider a closed bounded domain $\Omega \in \mathbb{R}^3$. The flow velocity u and pressure p are governed by Navier-Stokes equations for incompressible viscous flow with non constant density ρ , that can be written as follows [3] [4]:

$$\begin{cases} \rho(\frac{\partial u}{\partial t} + u \cdot \nabla u) = \nabla \cdot S - \rho g e_3 \\ \nabla \cdot (\rho u) = 0 \text{ in } \Omega \times (0, T), \end{cases} \quad (1)$$

where S is the stress tensor and g is the gravity.

The law of perfect gas $p = \rho R \theta$ gives an inverse law in terms of the temperature θ at constant pressure. The Boussinesq approximation to Navier-Stokes equations assumes that θ varies little around a mean temperature θ_0 and its effect is felt only in the term g/ρ . Hence one may divide (1) by ρ and obtain:

$$\begin{cases} \frac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p - \nu \Delta u = -e_3 g \frac{\rho}{\rho_0} \\ \nabla \cdot u = 0 \text{ in } \Omega \times (0, T), \end{cases} \quad (2)$$

where ν is the kinematic viscosity and p the reduced pressure.

To evaluate the buoyancy term f/ρ the Boussinesq approximation writes

$$\theta = \theta_0 + \theta', \quad |\theta'| \ll |\theta_0| \quad \Rightarrow \quad \rho = \frac{p_0}{R(\theta_0 + \theta')} \sim \frac{p_0}{R\theta_0} \left(1 - \frac{\theta'}{\theta_0}\right)$$

This can be re-written equivalently as

$$\rho - \rho_0 = -\alpha \rho_0 (\theta - \theta_0) \quad (3)$$

where α is the coefficient of volume expansion. Therefore the variation in density is neglected everywhere except in the buoyancy term which is a linear function of temperature θ . Notice that ρ has gone out of the divergence operator in the mass conservation equation only because we assumed ρ_0 constant.

Notice also that $-e_3 g (\rho_0 + \alpha \rho_0 \theta_0) / \rho_0$ is the gradient of $g(1 + \alpha \theta_0)z$ when θ_0 is constant, so that this term can be absorbed into the pressure and finally $-\rho g e_3 / \rho_0 \sim e_3 g \alpha \theta$.

2.2 Temperature

The temperature equation comes from the energy conservation equation where it is assumed that the flow is incompressible with constant density and the fluid viscosity effects are neglected.

$$\frac{\partial \theta}{\partial t} + u \nabla \theta - \kappa \Delta \theta = 0 \text{ in } \Omega, \quad (4)$$

where κ is the temperature diffusion.

2.3 Boundary Conditions

The boundary $\partial\Omega$ is subjected to the typical no slip or no-stress boundary conditions [4, 5] :

- specified velocity (Dirichlet boundary conditions) :

$$u = w \text{ on } \Gamma_1, \quad (5)$$

- specified tractions (Neumann boundary conditions) :

$$-p + (\nu(\nabla u + \nabla u^T) \cdot n \cdot n) = 0 \text{ and } (\nu(\nabla u + \nabla u^T) \cdot n \cdot s) = 0 \text{ on } \Gamma_2, \quad (6)$$

where $\Gamma_1 \cup \Gamma_2 = \partial\Omega$, n and s represent the outward unit normal and corresponding unit tangent respectively. Γ_1 represents the walls and the flow input, and Γ_2 represents the flow output.

The initial condition is a prescribed velocity:

$$u(x, 0) = u_0(x). \quad (7)$$

Connected to the temperature equation we have the following boundary conditions:

- specified temperature (Dirichlet boundary conditions):

$$\theta = \theta_0 \text{ on } \Gamma_1, \quad (8)$$

- Fourier conditions:

$$\frac{\partial \theta}{\partial n} + a\theta + b(\theta^4 - \theta_e^4) = 0 \text{ on } \Gamma_2, \quad (9)$$

where $\Gamma_1 \cup \Gamma_2 = \partial\Omega$, θ_e the external temperature, a represents the absorption coefficient and b the radiation coefficient.

3 NUMERICAL SCHEME

3.1 Navier-Stokes resolution

The numerical resolution of these equations is obtained using a projection method introduced first by Chorin [6] and later improved by numerous works. The key idea is to decouple the solution for u (velocity) and p (pressure) in the original problem into a series of problems to approximate u and $p(T)$ which are "good" approximations of the solution. The general scheme of the projection method, as suggested by [6], is composed of the following stages:

1. Given the initial conditions u_0 with $\nabla \cdot u_0 = 0$ (approximate the gradient $\nabla p(t)$),
2. Solve the momentum equations at projection time $t = T$ (without taking care of the divergence-free constraint), for the *intermediate velocity* \tilde{u} , with $\tilde{u}_0 = u_0$ at $t = 0$:

$$\left\{ \begin{array}{l} \frac{\partial \tilde{u}}{\partial t} + \tilde{u} \cdot \nabla \tilde{u} = \nabla \cdot \tilde{S} + \tilde{f} \text{ in } \Omega, \\ \tilde{u} = w \text{ on } \Gamma_1, \\ (\tilde{S} \cdot n) \cdot n = 0 \text{ and } (\tilde{S} \cdot n) \cdot s = 0 \text{ on } \Gamma_2. \end{array} \right.$$

3. Solve for Φ from:

$$\left\{ \begin{array}{l} \nabla^2 \Phi = \nabla \cdot \tilde{u}(T) \text{ in } \Omega, \\ \frac{\partial \Phi}{\partial n} = 0 \text{ on } \Gamma_1, \\ \Phi = 0 \text{ on } \Gamma_2. \end{array} \right.$$

4. Compute $v = \tilde{u}(T) - \nabla \Phi$ in $\bar{\Omega}$.

The pressure $p(T)$ is given by $p(T) = \frac{\Phi}{T}$,

5. Report v , set $t = 0$, $u_0 = v$ in Ω and on Γ_2 and go to step (2).

In this scheme, the choice of the boundary conditions, for both \tilde{u} and Φ , is crucial and requires the use of the pressure field p_0 and the rate of the change of the pressure field \dot{p}_0 , thus leading to the resolution of two Poisson problems. The overall cost of this scheme related to the need for solving three Poisson problems per projection cycles. Simplifications can be made by ignoring the compatibility conditions for the boundary conditions thus avoiding the tedious computation of ∇p and of $\nabla \dot{p}$ on Γ .

3.2 Temperature resolution

The resolution of the advection-diffusion problem is performed using a SUPG (Streamline Upwind/Petrov-Galerkin) approach as a pure Galerkin approach can perform poorly when dealing with such problem [7]. To solve the non-symmetric matrices we use the iterative Generalized Minimal Residual (GMRES) technique. In our case, we use a specific version of GMRES which allows a variable preconditioner implemented with a reverse communication protocole for more flexibility.

4 MESH ADAPTATION

In this context, it is well known that the quality of the numerical solutions is strongly related to the underlying (shape and size) mesh quality [8]. Hence, it is desirable to find a compromise between the desired accuracy and the time required to compute the solutions, thus to reduce as much as possible the number of degrees of freedom. Mesh adaptation is the key point of this strategy.

4.1 *A posteriori* error estimation

If we assume that the finite element (approximation) error is bounded by the interpolation error [9], the problem consist to characterize the *optimal* mesh on which the interpolation error is bounded by a given tolerance value. In other words, the goal is to equidistribute the interpolation error over the mesh elements in order to control the numerical accuracy of the solution.

We consider the L^∞ norm of the interpolation error defined in a mesh element K as:

$$\|u - \Pi_h u\|_{\infty, K} \leq c \max_{x \in K} \max_{\vec{e} \in E_K} \langle \vec{e}, |H_u(x)| \vec{e} \rangle, \quad (10)$$

where c is a constant, E_K is the set of the edges of K and $|H_u|$ is the absolute value of the Hessian of the variable u . More precisely, we have $|H_u| = \mathcal{R} |\Lambda| \mathcal{R}^{-1}$, with $|\Lambda| = \text{diag}(|\lambda_i|)$, where \mathcal{R} is the eigenvector matrix and $\Lambda = \text{diag}(\lambda_i)$ is the eigenvalue matrix¹.

This error is related to the Hessian of the variable u and the mesh edges, so it provides anisotropic (directional) information. In practice, this estimation can't be used because $|H_u|$ isn't known. So, a metric tensor $\widetilde{\mathcal{M}}(K)$ dependant on H_u is defined as :

$$\varepsilon_K = c \max_{\vec{e} \in E_K} \langle \vec{e}, \widetilde{\mathcal{M}}(K) \vec{e} \rangle. \quad (11)$$

where ε_K denotes the interpolation error.

Finally, for a mesh element K , we introduce the metric tensor $\mathcal{M}(K) = c \varepsilon^{-1} \widetilde{\mathcal{M}}(K)$ (with ε a fixed error) and each mesh edge e must then comply with the equality:

$$\langle \vec{e}, \mathcal{M}(K) \vec{e} \rangle = 1. \quad (12)$$

¹The Hessian matrix is symmetric, it can always be decomposed in such manner.

This metric tensor \mathcal{M} prescribes *unit* edge lengths. We have shown that controlling the mesh edges (*i.e.*, the length of the edges) allows to control (equidistribute) the interpolation error on the mesh elements [10].

4.2 Metric creation

Let ε be the chosen interpolation error. To avoid unrealistic metrics, we define h_{min} (resp h_{max}) the minimal (resp maximal) allowed edge length. The previous section introduced a global bound on the interpolation error over the mesh elements, thus leading to define an anisotropic metric tensor. Practically, we have to bound the eigenvalues λ_i of H_u as follows:

$$\tilde{\lambda}_i = \min(\max_i(c\varepsilon^{-1}|\lambda_i|, h_{max}^{-2}), h_{min}^{-2}). \quad (13)$$

Relative error. The Equation (10) gives a global majoration of interpolation error. However, we often use many variables to define the metric what involve to have a relative majoration of interpolation error. That allows to remove dimensional constraint of the variables. To normalize the Equation (10), we obtain :

$$\frac{\|u - \Pi_h u\|_{\infty, K}}{\|u\|_{\infty, \Omega}} \leq c \max_{x \in K} \max_{\vec{e} \in E_K} \frac{\langle \vec{e}, |H_u(x)|\vec{e} \rangle}{\|u\|_{\infty, \Omega}} \quad (14)$$

Metric intersection. Moreover, in the context of numerical simulations (especially in CFD), it is often desirable to combine various metrics together, each of which associated to a single variable (e.g. pression, temperature, velocity, etc.). To this end, we introduce a metric intersection algorithm based on the simultaneous reduction of the quadratic forms underlying the metric tensors.

Given two metric tensors \mathcal{M}_1 and \mathcal{M}_2 , the intersection metric $\mathcal{M} = \mathcal{M}_1 \cap \mathcal{M}_2$ is represented by the ellipsoid²: $\mathcal{E}_{\mathcal{M}} = \sup_{\mathcal{M}_i \in \mathbf{M}_d} \mathcal{E}_{\mathcal{M}_i} \subset \mathcal{E}_{\mathcal{M}_1} \cap \mathcal{E}_{\mathcal{M}_2}$.

4.3 Unit mesh adaptation

Given a discrete metric tensor field \mathcal{M} prescribing the size and stretching of the elements at the vertices of a mesh \mathcal{H}_i , the aim is to generate a new mesh \mathcal{H}_{i+1} in which all elements comply with the specification \mathcal{M} . To this end, the length of edge PX , incident to P , is computed with respect to \mathcal{M} as:

$$l_{\mathcal{M}(P)}(\overrightarrow{PX}) = \langle \overrightarrow{PX}, \overrightarrow{PX} \rangle_{\mathcal{M}(P)}^{\frac{1}{2}} = \sqrt{{}^t \overrightarrow{PX} \mathcal{M}(P) \overrightarrow{PX}}. \quad (15)$$

²A metric tensor $\mathcal{M}(P)$ can be represented by an ellipsoid \mathcal{E} describing the geometric locus of the points equidistant from point P : $\|\overrightarrow{OP}\|_{\mathcal{M}} = 1$.

According to Equation (12), it is possible to define a normalized metric tensor \mathcal{M} so as to prescribe unit length edges, *i.e.*, $l_{\mathcal{M}(P)}(\overrightarrow{PX}) = 1$. Actually, as the metric varies from vertex to vertex, we need to use the definition of the average length of edge PX :

$$l_{\mathcal{M}}(\overrightarrow{PX}) = \int_0^1 \sqrt{{}^t\overrightarrow{PX} \mathcal{M}(P + t\overrightarrow{PX}) \overrightarrow{PX}} = 1. \quad (16)$$

Given such a metric \mathcal{M} , the desired mesh is such that each and every edge has a *unit length* ($l_{\mathcal{M}}(\overrightarrow{PX}) \approx 1$). Such a mesh is naturally called a *unit mesh* [11], [12].

Surface mesh adaptation. In our approach, the surface discretization is first adapted using local mesh modification operations in order to generate a unit mesh. This stage is based on the edge length analysis with respect to the discrete metric tensor [13, 14]. Practically, the algorithm computes the length of each mesh edge in the metric and (i) edge too short are deleted while (ii) edges too large are splitted into unit sub-segments. To this end, two sets of local operations are carried out:

- topological: edge flipping, edge collapsing, degree relaxation,
- geometric: edge splitting, node smoothing.

Volume mesh adaptation. Once a new adapted surface has been created, it is used to construct an adapted volume mesh. This stage is also based on the edge length analysis and involves the *Delaunay kernel* to insert newly created internal nodes into the current triangulation. In the anisotropic context, this algorithm has been modified so as to take into account metric tensors at the element vertices [15]. Again, local mesh modification operations are carries out to improve the overall mesh quality.

4.4 General scheme

According to the previous discussion, the global adaptation scheme can be decomposed into the five following stages:

1. Given a mesh \mathcal{H}_i and a solution u_i ,
2. Compute the solution u_{i+1} on the mesh \mathcal{H}_i ,
3. Use the geometric error estimate to compute a discrete metric tensor \mathcal{M}_i ,
4. Adapt the mesh \mathcal{H}_i using \mathcal{M}_i ,
5. Interpolate the solution u_i on the new mesh \mathcal{H}_{i+1} and iterate to step 1 with $i = i + 1$.

5 APPLICATIONS

In this section, we present two results on air flow computation coupled with temperature computation in three-dimensionnal computational domain. Firstly, we consider a single room to illustrate the natural convection problem and secondly, we study the heating of the last floor of a two-storeyed furnished house.

5.1 Computational mesh construction

The domain of interest is defined through a discrete representation, namely a piecewise linear approximation of the domain boundary or a surface mesh. In this case, the initial surface triangulation (Figure 1) contains very few (821) points and can be considered as a geometric mesh [13].

Obviously such a mesh is not well suited for computational purposes and especially not intended for finite element computations as envisaged here. To this end, a computational mesh has been created (using YAMS software[16]), representing an accurate approximation of the domain geometry (the edge length is related to the local curvatures) in which the element shape quality and the mesh gradation have been both taken into account. This initial computational mesh contains 1,252 vertices and 24,138 triangles. Then, a three-dimensional computational volume mesh has been generated using a Delaunay-based approach (using GHS3D software [15]), that contains 43,130 vertices and 225,035 tetrahedra.

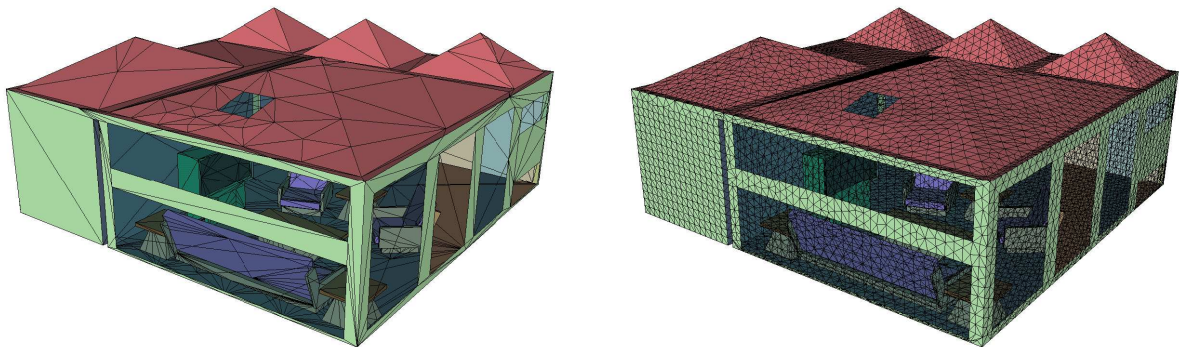


Figure 1: *Geometric coarse mesh describing the domain boundary (on the left). Computational mesh : mesh generated by YAMS (on the right).*

5.2 Parallel implementation

The solver is parallelized by a Domain Decomposition algorithm (using METIS decomposer [17]) with MPI to run on PC cluster. The explicit steps achieve perfect parallelism

of course. It is only the pressure projection which create problems. The parallelization is implemented at the matrix-vector multiply level, so that with a decent preconditioner and with a PC cluster with 2 Gig memory boards the communications are not felt.

5.3 Natural convection

Temperature distribution. In this example, we consider a table and a refrigerator within a room which has a window. For the simulation, the refrigerator is heating the room and the window is cooling it. Figure 2 (on the left) illustrates the isosurfaces of the temperature distribution within the room. This isosurfaces show the equilibrium which is created between the hot source and the cool source.

Air flow computation. For this simulation, all walls and internal furnitures are considered as viscous walls. So there is only the temperature which creates an air flow. On the right of the Figure 2, we visualize some streamlines of the velocity vector. We can observe the air movement, it goes up to the hot part and goes back down in the cool part. The modulus of the air velocity is very small (about $10^{-3} m.s^{-1}$).

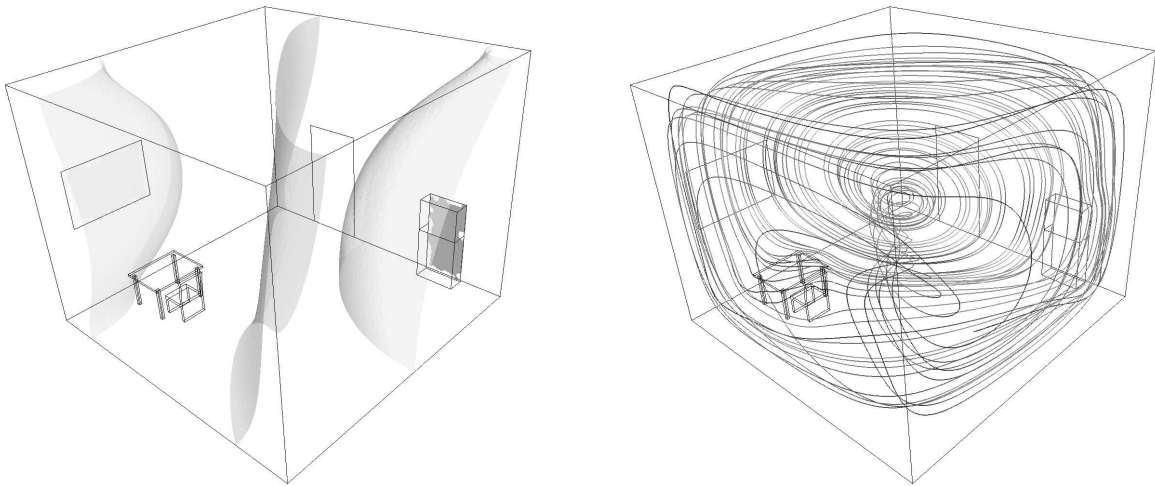


Figure 2: *Isosurfaces of the temperature distribution (on the left). Streamlines of the velocity field (on the right).*

5.4 Heating of a house

Air flow computation. In this example, the geometry we consider is the last floor of a two-storeyed furnished house. Two inputs and two outputs (boundary conditions) have been prescribed on external windows. All house walls and internal furnitures are

considered here as viscous walls. Figure 3 illustrates the distribution of streamlines of the velocity vector.

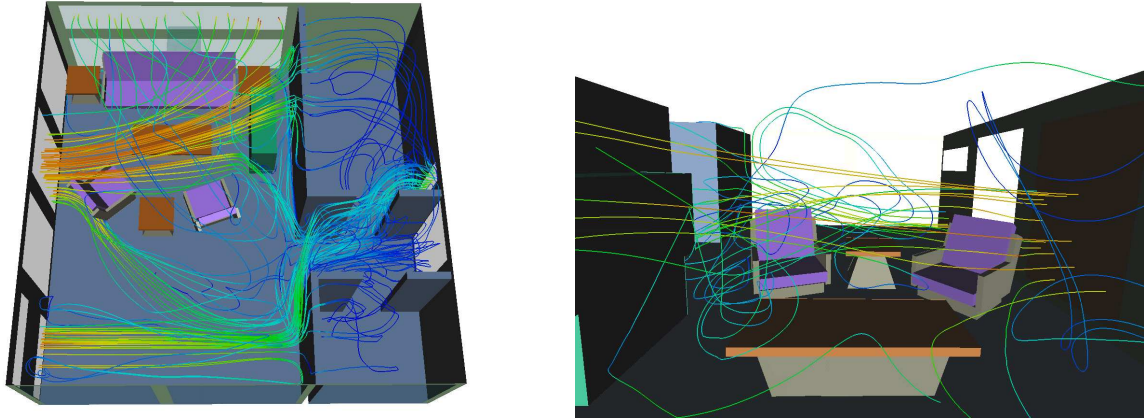


Figure 3: *Streamlines of the velocity field.*

Temperature distribution. Two heat sources have been set on two windows and absorption and diffusion conditions have been prescribed on the remaining part of the domain. Figure 4 illustrates the isosurfaces of the temperature distribution within the building.

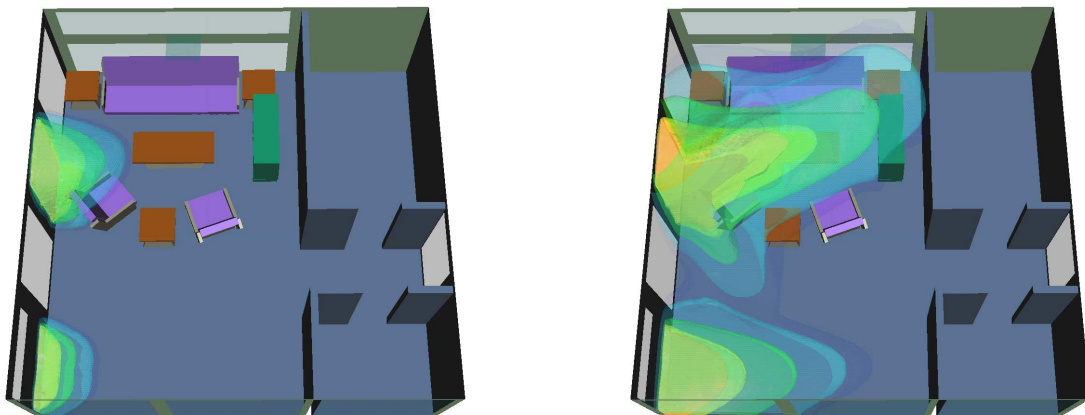


Figure 4: *Isosurfaces of the temperature distribution within the house.*

Mesh Adaptation. The result presented here is a first anisotropic result. To create the metric, we consider three variables corresponding at the three velocity components. Indeed, if we adapted based on the L_2 norm of the velocity field, the vortex are not captured because like seen in last example, the modulus of velocity is very small in this zones. In this example, the mesh has been adapted 7 times and at each iteration, we compute about 1 seconde of physical time. The time to create the new mesh is close to 5 minutes on a PC workstation. The time to solve the Navier-Stokes equations is close to 4 hours 30 minutes on 6 PC workstation. The time required to solve the advection-diffusion problem is close to 6 hours 30 minutes on 6 PC workstation. The last mesh constains about 96,000 vertices and 535,800 tetrahedra. Figure 5 shows various cut through the adapted volume meshes at final iteration.

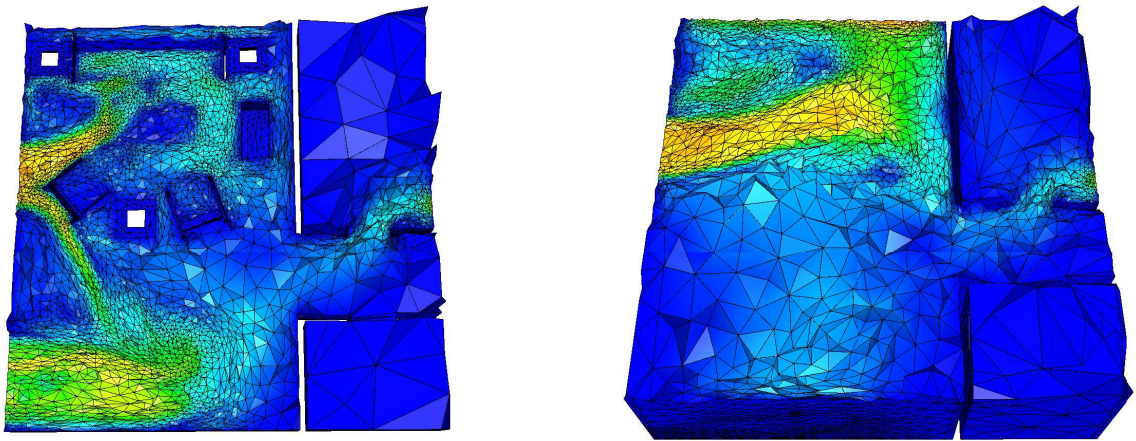


Figure 5: *Adapted meshes: cut through the tetrahedra and associated velocity field.*

6 CONCLUSIONS

The results presented in this paper have demonstrated the feasibility and the efficiency of the global approach for coupling Navier-Stokes and advection-diffusion problems in an adaptative scheme. Indeed, the anisotropy mesh adaptation reduces by far the number of nodes, so increase the computation speed. However, progress should be done to find a better adaptation criterion to create more *optimal* mesh.

7 ACKNOWLEDGEMENTS

The first author would like to acknowledge the french embassy in Helsinki and the *Association Franco-Finlandaise pour la Recherche Scientifique et Technique* for their support to permit her to come to the European Congress on Computational Methods in Applied Sciences and Engineering 2004.

REFERENCES

- [1] L. LANDAU AND F. LIPSCHITZ (1964) *Mécanique des fluides*, editions Mir
- [2] B. MOHAMMADI, O. PIRONNEAU (1994), *Analysis of the K-Epsilon Turbulence Model*, Wiley
- [3] R. GLOWINSKI (1984), *Numerical methods for nonlinear variational problems*, Springer-Verlag, New York.
- [4] O. PIRONNEAU (1989), *The finite element method for fluids*, Wiley, Chicester.
- [5] G. MEDIĆ, B. MOHAMMADI (1999), *NSIKE: An Incompressible Navier-Stokes Solver for Unstructured Meshes*, RT-3644, INRIA-Rocquencourt.
- [6] J.A. CHORIN (1967), *A numerical method for solving incompressible viscous flow problems*, *J. Compt. Phys.*, 2, 12-26.
- [7] A. QUARTERONI AND A. VALLI (1994), *Numerical approximation of partial differential equations*, Springer-Verlag, Berlin, Heidelberg.
- [8] P.G. CIARLET (1991), *Basic Error Estimates for Elliptic Problems*, in *Handbook of Numerical Analysis*, vol II, Finite Element methods (Part 1), P.G. Ciarlet and J.L. Lions Eds, North Holland, 17-352.
- [9] F. GUIBAULT, P. LABBÉ, J. DOMPIERRE (2002), *Adaptivity Works! Controlling the Interpolation Error in 3D*, *Proc. Fifth World Congress on Computational Mechanics*, Vienna, Austria, July.
- [10] F. ALAUZET ET P.J. FREY (2003), *Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques*, RR-4759, INRIA Rocquencourt.
- [11] H. BOROUCAKI ET P.L. GEORGE (2000), *Quality mesh generation*, *C.R. Acad. Sci. Paris*, t 328, Serie IIb, 505-518.
- [12] P.L. GEORGE ET H. BOROUCAKI (1998), *Génération automatique de maillages tridimensionnels respectant une carte de taille*, *Revue européenne des éléments finis* 7(4), 339-363.
- [13] P.J. FREY(2000), *About surface remeshing*, In *Proc.of 9th Int. Meshing Roundtable*, New Orleans, LO, USA, 123-136.
- [14] R. LÖHNER (1996), *Regridding Surface Triangulations*, *Jour. of Comput. Phys.*, **126**, 1-10.

- [15] P.L. GEORGE (2002), *Premières expériences de maillage automatique par une méthode de Delaunay anisotrope en trois dimensions*, RT-0272, INRIA Rocquencourt.
- [16] P.J. FREY (2001), *YAMS : A fully Automatic Adaptive Isotropic Surface Remeshing Procedure*, RT-0252 INRIA-Rocquencourt.
- [17] KARYPIS, GEORGE AND KUMAR, VIPIN (1998) *A fast and high quality multilevel scheme for partitioning irregular graphs* *SIAM Journal on Scientific Computing*, **20**, 359-392