A fast and accurate method to solve the incompressible Navier–Stokes equations

Article *in* Engineering Computations · February 2013

DOI: 10.1108/02644401311304854

CITATIONS
25

READS
104

5 authors, including:

Sergio Rodolfo Idelsohn
Catalan Institution for Research and Advanced Studies
407 PUBLICATIONS **5,740** CITATIONS
SEE PROFILE

Norberto Nigro
National Scientific and Technical Research Council
208 PUBLICATIONS **956** CITATIONS
SEE PROFILE

Juan M. Gimenez
National Scientific and Technical Research Council
34 PUBLICATIONS **92** CITATIONS
SEE PROFILE

Riccardo Rossi
Universitat Politècnica de Catalunya
110 PUBLICATIONS **1,243** CITATIONS
SEE PROFILE

A fast and accurate method to solve the incompressible Navier-Stokes equations

Journal:	*Engineering Computations*
Manuscript ID:	Draft
Manuscript Type:	Research Article
Keywords:	Navier-Stokes equations, Particle methods, Updated Lagrangian formulations, Real Time CFD

A fast and accurate method to solve the incompressible Navier-Stokes equations

Sergio Idelsohn[a,b], Norberto Nigro[c], Juan Gimenez[c], Riccardo Rossi[b], Julio Marti[b]

[a-] Intitució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain
[b-] Centro Internacional de Métodos Numéricos en Ingeniería (CIMNE), Barcelona, Spain
[c-] Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC), Santa Fe, Argentina

Abstract

Current work aims to highlight the possibilities of a novel lagrangian formulation in dealing with the solution of the incompressible Navier-Stokes equations. Our main goal is to show how the proposed approach, which was originally designed to deal with heterogeneous fluids of free-surface flows, can be competitive with eulerian alternatives even in their range of optimal application. We this objective in mind, we select a number of benchmark examples and prove that the proposed algorithm, provides results which compare favorably with reference solutions and commercial codes both in terms of solution time and of accuracy achieved.

Keywords: Navier-Stokes equations; Particle methods; Large time-steps; Incompressible fluid flows; Updated Lagrangian formulations; Real Time CFD

1. Introduction

Standard formulations for the solution of the incompressible Navier-Stokes equations may be split in two classes depending on the approach chosen for the description of the inertial terms, namely Eulerian and Lagrangian approaches. In the first class, the acceleration is described as the sum of the spatial derivative of the velocity plus the convective term. In the second approach, the acceleration is simply described as the total derivative of the velocity.

Over the last 30 years, computer simulation of incompressible flows has been mainly based on the Eulerian formulation (see [18] for references on this topic). However, with this formulation, it is still difficult to analyze large 3D problems in which the shape of the free-surfaces or internal interfaces changes continuously or in fluid–structure interactions where complex contact problems are involved. In all these problems the computing time is sometimes so high that makes the method unpractical.

In the last few years, several solutions using the Lagrangian formulation to solve the compressible and incompressible Navier-Stokes equations have been developed [2,12,13]. The advantages of these solutions to solve problems with free-surfaces or multi-fluids with complicated internal interfaces haves been demonstrated [15]. In general, these formulations are more expensive than a standard Eulerian approaches if they are used in homogeneous flows, but they justify their popularity in solving free-surface flows or complicated multi-fluids flows in which the standard Eulerian formulations are inaccurate or, sometimes, impossible to be used [15].

When attempting to classify Navier-Stokes solvers it is important to take in account the level of locality of the information needed. One can define as "*implicit*" a solution strategy in which a change in the solution in any part of the domain can potentially influence the solution in any other part of it. "*Explicit*" strategies can hence be understood as strategies in which the solution at a point, within a time step, is only influenced by a portion of the domain around the point. A fundamental feature of implicit solver is thus that they enforce a strong coupling between time and space, while in explicit solvers this coupling is somewhat relaxed. In the second class we include not only the explicit time integration schemes but also methods that lead to a linear system of equations that may be factorized once and integrated in time with the same factorized matrix.

Even though implicit time integration schemes are preferred in the literature against explicit ones, the latter are in a better position attending to the present of hardware technology. This is oriented to the usage of parallel computer and general purpose graphic processor units (GPGPU) [9].

The main target of this work is to show that Lagrangian formulations are not only valuable to solve heterogeneous fluid flows with free-surfaces. We will prove on the contrary that even for homogeneous fluid flows, without free-surfaces or internal interfaces, they are able to yield accurate solutions while being competitively fast when compared to state-of-the-art eulerian solvers. In this sense, we shall also observe that the proposed algorithms were implemented by the authors in two distinct computational codes, one of them within the Kratos framework [10, 11].

The paper will be divided as follows: In a first section we will describe the X-IVAS time integration to be used for de convective terms. This time integration was described before in a previous work of the authors [1] but it is included here for completeness. The Particle Finite Element Method (PFEM) for fixed mesh is then refreshed and then, the way to use it with the X-IVAS time integration of the convective term is presented (PFEM-2). Finally, before showing the numerical examples, the strategy to obtain a fast algorithm to be parallelized is described.

2. The X-IVAS time integration of the convective terms

2.1 General description

Let $\mathbf{x}_p$ be the vector defining the position of a particle in a 3D space, function of the time t that for simplicity we will write $\mathbf{x}_p^t$. At time $t=t^n$ we will write $\mathbf{x}_p^n$, at time $t=t^n+\Delta t=t^{n+1}$ we will write $\mathbf{x}_p^{n+1}$ and in general, in any time between $t=t^n$ and $t=t^{n+1}$ we will write $\mathbf{x}_p^{n+t}$.

Let $\mathbf{V}^{n+t}(\mathbf{x}_p^{n+t})$ and $\mathbf{A}^{n+t}(\mathbf{x}_p^{n+t})$ vectors defining the velocity and the acceleration respectively of a particle $\mathbf{x}_p$ at any time t^{n+t}

$$\begin{cases} \mathbf{V}^{n+t}(\mathbf{x}_p^{n+t}) = \dfrac{D\mathbf{x}_p^{n+t}}{Dt} & (2.1) \\ \mathbf{A}^{n+t}(\mathbf{x}_p^{n+t}) = \dfrac{D\mathbf{V}^{n+t}(\mathbf{x}_p^{n+t})}{Dt} & (2.2) \end{cases}$$

where $\frac{D\phi}{Dt}$ represents the material (Lagrangian) derivative in time of any function ϕ. The material derivative is connected with the spatial derivative by the convective terms:

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + V_i \frac{\partial \phi}{\partial x_i} = \frac{\partial \phi}{\partial t} + \mathbf{V}^T \nabla \phi$$

In all initial value problems like the transient Navier-Stokes equations, the time solution of a problem consists in: given all the variables at time $t = t^n$, find the same variables at time $t = t^{n+1}$. In other words, to integrate in time equations (2.1) and (2.2):

$$\begin{cases} \mathbf{x}_p^{n+t} = \mathbf{x}_p^n + \int_n^{n+t} \mathbf{V}^\tau(\mathbf{x}_p^\tau) d\tau & (2.3) \\ \mathbf{V}^{n+t}(\mathbf{x}_p^{n+t}) = \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+t} \mathbf{A}^\tau(\mathbf{x}_p^\tau) d\tau & (2.4) \end{cases}$$

The accuracy of the results will depend to a great extent on the accuracy of the discretization of the velocity and acceleration in the space, but also in the approximation introduced in the integration of (2.3) and (2.4).

Time integration of the velocity:

The idea of X-IVAS method is to use the velocity streamlines obtained at time step t^n to approximate the final position of a particle $\mathbf{x}_p^{n+1}$.

Let then

$$\mathbf{x}_p^{n+t} \approx \mathbf{y}_p^{n+t} = \mathbf{x}_p^n + \int_n^{n+t} \mathbf{V}^n(\mathbf{x}_p^\tau) d\tau \quad (2.5)$$

Equation (2.5) is explicit because we are using only information at time step t^n. In this case, we are using neither a constant nor a linear approximation of the velocity field. We are using the same high order approximation the velocity field has at time t^n. The only difference with the exact integration (2.3) is that here we are performing the integral (within each time step) following a pseudo trajectory of the particles calculated with the velocity streamline, instead of following the true trajectory (Fig. 2.1).

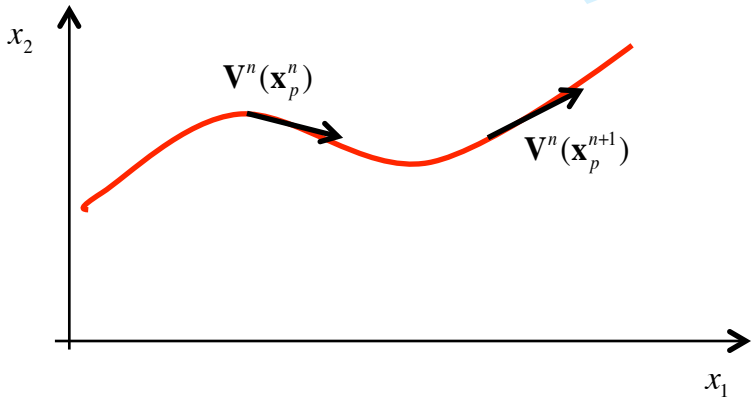

Fig. 2.1 *Integration following the velocity streamlines*

Once discretized, Equation (2.5) may be integrated analytically or using any standard time integration scheme like explicit Runge-Kutta or dividing the integration domain in subdomains. The way to integrate analytically Equation (2.5) inside each triangular element is explained in Ref [1].

Time integration of the acceleration:

The idea proposed in (2.5) may be also used for the acceleration. That is, for improving the time integration of the acceleration while remaining explicit in time. This means to approximate equation (2.4) by:

$$\mathbf{V}^{n+t}(\mathbf{x}_p^{n+t}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+t} \mathbf{A}^n(\mathbf{x}_p^\tau)\, d\tau \tag{2.6}$$

Equation (2.6) represents an integration following the acceleration streamlines (See Fig. 2.2) obtained at time t^n. This may be solved using any of the particle position integrations described before.

$$\begin{cases} \mathbf{x}_p^{n+t} \approx \mathbf{x}_p^n + \int_n^{n+t} \mathbf{V}^n(\mathbf{x}_p^\tau)\, d\tau \\ \mathbf{V}^{n+t}(\mathbf{x}_p^{n+t}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+t} \mathbf{A}^n(\mathbf{x}_p^\tau)\, d\tau \end{cases} \tag{2.7}$$

We must note that equations (2.7) are still explicit because they are using the velocity and acceleration at time t^n (Fig.2.2). The way to integrate analytically Equation (2.7) inside each triangular element is explained in the Ref.[1].

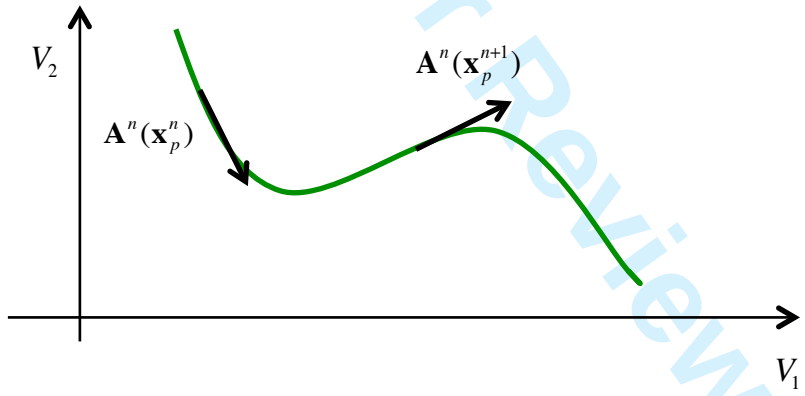

Fig. 2.2 *Integration following the acceleration streamlines*

2.2 The X-IVAS integration applied to the incompressible Navier-Stokes equations

In the Navier-Stokes equation, the acceleration is obtained from the momentum conservation equation:

$$\mathbf{A}^t(\mathbf{x}_p^t) = \frac{1}{\rho^t(\mathbf{x}_p^t)}[\nabla \bullet \sigma^t(\mathbf{x}_p^t) + \mathbf{b}^t(\mathbf{x}_p^t)] \tag{2.8}$$

with the stress tensor

$$\sigma^t(\mathbf{x}_p^t) = \tau^t(\mathbf{x}_p^t) - p^t(\mathbf{x}_p^t)\mathbf{I} \tag{2.9}$$

and the deviatoric tensor

$$\tau^t(\mathbf{x}_p^t) = \mu[\nabla \mathbf{V}^t(\mathbf{x}_p^t) + \nabla^T \mathbf{V}^t(\mathbf{x}_p^t)] \tag{2.10}$$

where ρ is the density, μ the viscosity, p the pressure, $\mathbf{b}$ a volumetric force and $\mathbf{I}$ the identity tensor.

The mass conservation reads

$$\frac{\partial \rho^t(\mathbf{x}_p^t)}{\partial t} + \nabla \bullet [\rho^t(\mathbf{x}_p^t)\mathbf{V}^t(\mathbf{x}_p^t)] = 0 \tag{2.11}$$

For an incompressible flow:

$$\rho^t(\mathbf{x}_p^t) = \rho(\mathbf{x}_p) = \rho_p = cte \neq 0 \tag{2.12}$$

and then (2.11) has the single form:

$$\nabla \bullet \mathbf{V}^t(\mathbf{x}_p^t) = 0 \tag{2.13}$$

Using the X-IVAS method presented before, the Navier-Stokes equations between two times t^n and t^{n+1} reads:

$$\begin{cases} \mathbf{x}_p^{n+t} = \mathbf{x}_p^n + \int_n^{n+t} \mathbf{V}^n(\mathbf{x}_p^\tau)\, d\tau \\ \mathbf{V}^{n+1}(\mathbf{x}_p^{n+1}) = \mathbf{V}^n(\mathbf{x}_p^n) + \dfrac{1}{\rho_p}\int_n^{n+1} [\nabla \bullet \sigma^n(\mathbf{x}_p^t) + \mathbf{b}^n(\mathbf{x}_p^t)]\, dt \end{cases} \tag{2.14}$$

Equation (2.14) is explicit not only for the velocity terms but also for the pressure one. If we try to solve Equation (2.14) dividing the time in several time steps Δt, two different limitations of the time step size are encountered: firstly it is limited due to the speed of sound and secondly, it is limited due to the viscous term. For incompressible flows, the speed of sound is infinity. For this reason, the pressure must remain implicit in order that the time integration scheme remains stable for $\Delta t > 0$. For this reason, we will slightly modify (2.14) in order to allow the pressure to remain implicit. On the other hand, the viscous term may be integrated in time explicitly or implicitly. In the first case, the Fourier number ($Fo = 2\mu\Delta t / \rho h^2$) must remain smaller than one to be stable. For this reason we have modified (2.14) in order to remain implicit not only for the pressure but also for the viscous terms. The new explicit-implicit integration scheme is described next.

Equation (2.14) is replaced by the following:

$$\begin{cases} \mathbf{x}_p^{n+t} = \mathbf{x}_p^n + \int_n^{n+t} \mathbf{V}^n(\mathbf{x}_p^\tau)\, d\tau \\ \rho_p \mathbf{V}^{n+1}(\mathbf{x}_p^{n+1}) = \rho_p \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+1} \left\{ \nabla \bullet \tau^n(\mathbf{x}_p^t) + \nabla \bullet \delta\tau(\mathbf{x}_p^t) - \nabla p^n(\mathbf{x}_p^t) - \nabla \delta p(\mathbf{x}_p^t) + \mathbf{b}^n(\mathbf{x}_p^t) \right\} dt \end{cases} \tag{2.15}$$

We remark the implicit terms for stress and the pressure in the second of the Eq. (2.15), i.e.:

$$\begin{cases} \delta\tau(\mathbf{x}_p^t) = \tau^t(\mathbf{x}_p^t) - \tau^n(\mathbf{x}_p^t) = \mu[\nabla \delta\mathbf{V}(\mathbf{x}_p^t) + \nabla^T \delta\mathbf{V}(\mathbf{x}_p^t)] \\ \delta\mathbf{V}(\mathbf{x}_p^t) = \mathbf{V}^t(\mathbf{x}_p^t) - \mathbf{V}^n(\mathbf{x}_p^t) \end{cases} \tag{2.16}$$

$$\delta p(\mathbf{x}_p^t) = p^t(\mathbf{x}_p^t) - p^n(\mathbf{x}_p^t) \tag{2.17}$$

The integral of the $\delta\tau$ and δp terms in (2.15) will be approximated by a fully implicit backward integration scheme:

$$\int_n^{n+1}\left\{\nabla\bullet\delta\tau(\mathbf{x}_p^t)-\nabla\delta p(\mathbf{x}_p^t)\right\}dt\approx\frac{\Delta t}{2}[\nabla\bullet\delta\tau(\mathbf{x}_p^{n+1})-\nabla\delta p(\mathbf{x}_p^{n+1})] \tag{2.18}$$

Then, the second equation of (2.15) will be split in the following two steps:

$$\begin{cases}\rho_p\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\mathbf{V}^n(\mathbf{x}_p^n)+\int_n^{n+1}\left\{\nabla\bullet\tau^n(\mathbf{x}_p^t)-\nabla p^n(\mathbf{x}_p^t)+\mathbf{b}^n(\mathbf{x}_p^t)\right\}dt+\dfrac{\Delta t}{2}\left\{\nabla\bullet\delta\tau(\mathbf{x}_p^{n+1})\right\}\\ \rho_p\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})-\dfrac{\Delta t}{2}\nabla[\delta p(\mathbf{x}_p^{n+1})]\end{cases} \tag{2.19}$$

Applying the divergence operator to both sides of the second equation of (2.19) and taking into account (2.13) gives:

$$\rho_p\nabla\bullet\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})=\nabla\bullet\left\{\nabla[\delta p(\mathbf{x}_p^{n+1})]\right\}\frac{\Delta t}{2} \tag{2.20}$$

Then the four steps using the X-IVAS technique remains:

Step I) Evaluate explicitly the velocity $\hat{\tilde{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})$ and the new particle position $\mathbf{x}_p^{n+t}$:

$$\begin{cases}\mathbf{x}_p^{n+t}=\mathbf{x}_p^n+\int_n^{n+t}\mathbf{V}^n(\mathbf{x}_p^\tau)d\tau\\ \rho_p\hat{\tilde{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\mathbf{V}^n(\mathbf{x}_p^n)+\int_n^{n+1}\left\{\nabla\bullet\tau^n(\mathbf{x}_p^t)-\nabla p^n(\mathbf{x}_p^t)+\mathbf{b}^n(\mathbf{x}_p^t)\right\}dt\end{cases} \tag{2.21}$$

Step II) Solve implicitly the first equation (2.19) to obtain the velocity correction $\delta\hat{\mathbf{V}}(\mathbf{x}_p^{n+1})$

$$[\rho_p-\mu\frac{\Delta t}{2}(\nabla\bullet\nabla+\nabla\bullet\nabla^T)]\delta\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\delta\hat{\tilde{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1}) \tag{2.22}$$

where $\delta\hat{\tilde{\mathbf{V}}}(\mathbf{x}_p^{n+1})=\hat{\tilde{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})-\mathbf{V}^n(\mathbf{x}_p^{n+1})$

Step III) Solve implicitly the Laplace equation to obtain the pressure increment $\delta p(\mathbf{x}_p^{n+1})$:

$$\rho_p\nabla\bullet\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})=\nabla\bullet\left\{\nabla[\delta p(\mathbf{x}_p^{n+1})]\right\}\frac{\Delta t}{2} \tag{2.23}$$

where $\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})=\hat{\tilde{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})+\delta\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})$

Step IV) Evaluate the new incompressible velocity $\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})$ and pressure $p^{n+1}(\mathbf{x}_p^{n+1})$:

$$\rho_p\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\hat{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})-\nabla[\delta p(\mathbf{x}_p^{n+1})]\frac{\Delta t}{2} \tag{2.24}$$

$$p^{n+1}(\mathbf{x}_p^{n+1})=p^n(\mathbf{x}_p^{n+1})+\delta p(\mathbf{x}_p^{n+1}) \tag{2.25}$$

It must be noted that for homogeneous fluid flows (e.g. fluid with constant viscosity and constant density) both implicit parts, Equation (2.22) and Equation (2.23)

lead to a matrix with constant coefficients. Such matrix may be factorized once during the whole calculation (or an expensive pre-conditioner could be computed) thus allowing a much faster solution phase for the following steps. In an Eulerian formulation, the presence of the convective terms in the tangent matrix make impossible to factorize the whole tangent matrix once. This fact, combined with the possibility of using much larger time steps, provides a competitive lead of the X-IVAS algorithm with respect to the Eulerian counterparts.

3. The Particle Finite Element Method with X-IVAS integration (PFEM-2) applied to the incompressible Navier-Stokes equations

A method using a Lagrangian formulation may be solved using either a moving mesh approach or relying on a fixed "background" mesh. The idea is the following: Once the velocity at the new position $\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})$ has been found by solving the Lagrangian equation, we can either build a new mesh can to connect the new particle position or devise a technique to project the velocity results onto the fixed background mesh. The previous versions of the PFEM [12, 13, 14, 15, 16, 17] relied on continuous remeshing thus implying reforming the Finite Element data structures and solving a different linear algebra problem at each time step. These tasks were identified to be very expensive in terms of CPU time and very hard of parallelize satisfactorily, thus limiting the performance of the algorithm. The fixed mesh algorithm was thus designed to avoid the remeshing step and appeared to provide the extra bonus of allowing dealing with constant coefficient linear algebra problems.

We should remark that the viscous problem to be solved in the implicit viscosity correction step, has constant coefficients exclusively in the case of constant viscosity distribution. Even though most of the flows are turbulent and viscosity depends on the flow itself, thus implying that this case is often not met, there is some interest in laminar flows as in microfluidics and also in direct numerical simulation (DNS) or LES with an explicit treatment of the Reynolds stress tensor. An extended description of the Particle Finite element Method with moving mesh and with fixed mesh may be obtained in [1]. We include in the following section a brief description of the algorithm with fixed mesh.

3.1 The PFEM-2 with fixed mesh

Suppose we have a fixed mesh, and inside each element we have several particles that can move in the entire domain in a Lagrangian way (see Fig. 3.1). At the beginning of each time step we know the velocity at the mesh nodes and also at the internal particles. Once the algorithm explained before is applied to all the particles to obtain $\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})$, we project on the fixed mesh nodes these velocities using an assembly of all the particles surrounding each mesh node. Different approaches are available to perform such projections, for example SPH or MLS (Moving Least Square) techniques could be used for the interpolation, as well as weights based on the position on the top of the underlying mesh. While a complete discussion of this problem falls beyond the scope of current work, as a general observation we should remark that, in order to avoid unwanted artificial numerical diffusion of the results, many particles are needed around each node of the fixed mesh.

Fig. 3.1 shows one possible approach to carry out the projection step. We are interested in computing the projection of a given particle field on the background mesh, we hence propose to compute for each node of such the contribution of the subset of particles placed "in its vicinity". In this work we consider as "close" (in the sense that they lay in the vicinity of a node, the particles that belong to the shadow region highlighted in Fig. 3.1 and corresponding to the node called node A. These particles, shown as red circles and called *prtcle +(n+1)*, represent the particles that at the time step t^{n+1} are inside the shadow region and come from another part of the domain. Their original position is represented in the same figure by yellow circles and are identified as *prtcle +(n)*. This shadow region is defined as the union of the sub-elements corresponding to the node A in all the elements that contain this node. The sub-element are the third parts of the triangle are corresponds to each vertex and formed by the vertex itself, the edges converging at that vertex and the centroid of the triangle. Particles marked in the figure as empty circles represent those particles that belonging to the elements that contain the node A do not contribute because they are out of the shadow region formed by the sub-elements just introduced. The streamlines corresponding to the subset of particles that at the present time are inside the shadow region and contribute to the value of the node A in the mesh at the current time step are drawn as the paths between the stars and the filled circles.

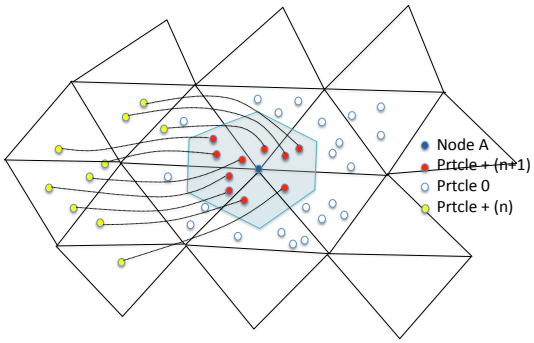

Fig. 3.1: *Fixed mesh and Lagrangian moving particles. How to compute the projection from particle to mesh nodes*

3.2 Spatial discretization of the incompressible Navier-Stokes equations using X-IVAS integration.

We have a fixed mesh and in this mesh we define the approximation on the component of the velocity field and the pressure using the FEM shape functions:

$$\mathbf{V}^n(\mathbf{x}) = \vec{\vec{\mathbf{N}}}^T(\mathbf{x})\vec{\mathbf{V}}^n \tag{3.1}$$

$$p^n(\mathbf{x}) = \vec{\mathbf{N}}_p^T(\mathbf{x})\,\vec{\mathbf{p}}^n \tag{3.2}$$

Starting with first step of the algorithm described in Section 2, the terms $\nabla \bullet \left\{\mu(\mathbf{x})[\nabla \mathbf{V}^n(\mathbf{x}) + \nabla^T \mathbf{V}^n(\mathbf{x})]\right\}$ and $\nabla p^n(\mathbf{x})$ in (2.21) will be approximated in a continuous field and each component will be called $\mathbf{g}^n(\mathbf{x})$ and $\prod^n(\mathbf{x})$ respectively, such that:

$$\mathbf{g}^n(\mathbf{x}) = \vec{\vec{\mathbf{N}}}^T(\mathbf{x})\,\vec{\mathbf{g}}^n \quad \text{and} \quad \Pi^n(\mathbf{x}) = \vec{\vec{\mathbf{N}}}^T(\mathbf{x})\,\vec{\Pi}^n \tag{3.3}$$

To evaluate $\vec{\mathbf{g}}^n$ and $\vec{\Pi}^n$ we will use the same approximation used in the standard FEM, that is:

$$\int_\Omega \vec{\vec{\mathbf{N}}}\left\{ \nabla \bullet [\mu\,(\nabla \mathbf{V}^n + \nabla^T \mathbf{V})] - \mathbf{g}^n \right\} d\Omega\, \vec{\mathbf{V}}^n = 0 \tag{3.4}$$

and

$$\int_{\Omega^n} \vec{\vec{\mathbf{N}}}\left\{ \nabla p^n - \prod^n \right\} d\Omega = 0 \tag{3.5}$$

Integrating by parts the first term in both equations (3.4 and 3.5):

$$-\int_\Omega \left\{ \nabla \vec{\vec{\mathbf{N}}} \mu\,[\nabla \vec{\vec{\mathbf{N}}}^T + \nabla^T \vec{\vec{\mathbf{N}}}^T] \right\} d\Omega\, \vec{\mathbf{V}}^n + \int_\Gamma \vec{\vec{\mathbf{N}}}\, q^n d\Gamma - \int_\Omega \vec{\vec{\mathbf{N}}}\, \vec{\vec{\mathbf{N}}}^T d\Omega\, \vec{\mathbf{g}}^n = 0 \tag{3.6}$$

and

$$-\int_\Omega \left\{ \nabla \vec{\vec{\mathbf{N}}} \vec{\mathbf{N}}_p^T] \right\} d\Omega\, \vec{\mathbf{p}}^n + \int_\Gamma \vec{\vec{\mathbf{N}}} p^n \mathbf{I}\,\mathbf{n}\, d\Gamma - \int_\Omega \vec{\vec{\mathbf{N}}}\, \vec{\vec{\mathbf{N}}}^T d\Omega\, \vec{\Pi}^n = 0 \tag{3.7}$$

we obtain:

$$\vec{\mathbf{g}}^n - \vec{\Pi}^n = (\vec{\vec{\mathbf{M}}})^{-1}(\vec{\vec{\mathbf{K}}}\vec{\mathbf{V}}^n - \vec{\vec{\mathbf{B}}}\vec{\mathbf{p}}^n - \vec{\sigma}^n) \tag{3.8}$$

with

$$\vec{\vec{\mathbf{K}}} = \int_\Omega \left\{ \nabla \vec{\vec{\mathbf{N}}} \mu\,[\nabla \vec{\vec{\mathbf{N}}}^T + \nabla^T \vec{\vec{\mathbf{N}}}^T] \right\} d\Omega \tag{3.9}$$

$$\vec{\vec{\mathbf{M}}} = \int_\Omega \vec{\vec{\mathbf{N}}}\, \vec{\vec{\mathbf{N}}}^T\, d\Omega \tag{3.10}$$

$$\vec{\vec{\mathbf{B}}} = \int_\Omega \nabla \vec{\vec{\mathbf{N}}}\, \vec{\mathbf{N}}_p^T\, d\Omega \tag{3.11}$$

and

$$\vec{\sigma}^n = \int_\Gamma \vec{\vec{\mathbf{N}}}\, q^n d\Gamma - \int_\Gamma \vec{\vec{\mathbf{N}}}\, p^n \mathbf{I}\mathbf{n}\, d\Gamma = \int_\Gamma \vec{\vec{\mathbf{N}}} [\mu(\nabla \mathbf{V}^n + \nabla^T \mathbf{V}^n) - p^n \mathbf{I}]\,\mathbf{n}\, d\Gamma$$
$$q^n = \mu(\nabla \mathbf{V}^n + \nabla^T \mathbf{V}^n)\,\mathbf{n} \tag{3.12}$$

Then, after discretization in space, the first step (Eq. 2.21) will read:

$$\begin{cases} \mathbf{x}_p^{n+t} = \mathbf{x}_p^n + \int_n^{n+t} \vec{\vec{\mathbf{N}}}^T(\mathbf{x}_p^\tau)\, d\tau\, \vec{\mathbf{V}}^n \\ \rho_p \hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1}) = \rho_p \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+1} \vec{\vec{\mathbf{N}}}^T(\mathbf{x}_p^t)\, dt\, \vec{\mathbf{g}}^n - \int_n^{n+1} \vec{\vec{\mathbf{N}}}^T(\mathbf{x}_p^t)\, dt\, \vec{\Pi}^n + \int_n^{n+1} \mathbf{b}^n(\mathbf{x}_p^t)\, dt \end{cases} \tag{3.13}$$

Note 1: all the time integrals in (3.13) may be evaluated by different methods. For linear shape function elements it may be computed analytically inside each finite element. It may also be evaluated dividing the time step Δt in several small sub-steps δt. This is not an expensive operation taking into account that computations are explicit and each particle may be evaluated separately from each other in parallel

Note 2: On the boundaries Γ_V where the velocity is known (Dirichlet boundary conditions), it is also known the acceleration. Let us call it $\left.\frac{D\mathbf{V}}{Dt}\right|_{\Gamma_V}$. On these boundaries, it is necessary to impose the value of $(\mathbf{g}-\Pi)|_{\Gamma_V}$ such that: $\mathbf{g}-\Pi=\rho\frac{D\mathbf{V}}{Dt}-\mathbf{b}$ *on* Γ_V.

After the first step, we project the $\hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})$ on the fixed mesh to obtain a vector with $\delta\hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})$ at the nodes of the fixed mesh $\delta\hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})\Rightarrow\delta\hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x})$.

The next step is to solve implicitly Eq. (2.22) which using a Finite Element approximation reads:

$$[\mathbf{M}(\rho)+\mathbf{K}\frac{\Delta t}{2}]\,\delta\hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x})=\mathbf{M}(\rho)\,\delta\hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x}) \tag{3.14}$$

where

$$\vec{\mathbf{M}}(\rho)=\int_\Omega\vec{\mathbf{N}}\,\rho_{(\mathbf{x})}\vec{\mathbf{N}}^T d\Omega \tag{3.15}$$

The following step is to solve also implicitly (Eq. 2.23). Using a classical FEM approximation reads:

$$\int_\Omega\vec{\mathbf{N}}_p\rho_{(\mathbf{x})}\nabla\bullet\hat{\mathbf{V}}^{n+1}d\Omega-\int_\Omega\vec{\mathbf{N}}_p\nabla\bullet\frac{\Delta t}{2}\nabla[\delta p^{n+1}]d\Omega=0 \tag{3.16}$$

Integrating by parts both terms in (3.16) gives:

$$-\int_\Omega\nabla\vec{\mathbf{N}}_p\rho_{(\mathbf{x})}\vec{\mathbf{N}}^T d\Omega\,\hat{\vec{\mathbf{V}}}^{n+1}+\int_\Omega\nabla\vec{\mathbf{N}}_p\frac{\Delta t}{2}\nabla\vec{\mathbf{N}}_p^T d\Omega\,\delta\vec{\mathbf{p}}^{n+1}+\int_{\Gamma_V}\vec{\mathbf{N}}_p\,\overline{\mathbf{V}}^{n+1}d\Gamma=0 \tag{3.17}$$

or in matrix form:

$$-\vec{\vec{\mathbf{B}}}\,(\rho)\hat{\vec{\mathbf{V}}}^{n+1}+\frac{\Delta t}{2}\vec{\vec{\mathbf{L}}}\,\delta\vec{\mathbf{p}}^{n+1}+\overline{\vec{\mathbf{V}}}^{n+1}=0 \tag{3.18}$$

with

$$\vec{\vec{\mathbf{B}}}\,(\rho)=\int_\Omega\nabla\vec{\mathbf{N}}_p\rho_{(\mathbf{x})}\vec{\vec{\mathbf{N}}}^T d\Omega \tag{3.19}$$

$$\vec{\vec{\mathbf{L}}}=\int_\Omega\nabla\vec{\mathbf{N}}_p\nabla\vec{\mathbf{N}}_p^T d\Omega \tag{3.20}$$

$$\overline{\vec{\mathbf{V}}}^{n+1}=\int_{\Gamma_V}\vec{\mathbf{N}}_p\,\overline{\mathbf{V}}^{n+1}d\Gamma=\int_{\Gamma_V}\vec{\mathbf{N}}_p[\rho_{(\mathbf{x})}\hat{\mathbf{V}}^{n+1}-\frac{\Delta t}{2}\nabla(\delta p)]d\Gamma \tag{3.21}$$

Despite the momentum equations do not need any spatial stabilization in the Lagrangian formulation, equation (3.18) must be stabilized in space for equal order velocity-pressure formulations like this one. Any space stabilization method may be used given, in general, a term like $\vec{\vec{\mathbf{S}}}(\tau)\,\vec{\mathbf{p}}^{n+1}$ (with $\vec{\vec{\mathbf{S}}}(\tau)$ a matrix similar to the Laplacian one but scaled by the stabilization parameter τ) that must be added to (3.18) [14]:

$$[\frac{\Delta t}{2}\vec{\vec{\mathbf{L}}}+\vec{\vec{\mathbf{S}}}(\tau)]\delta\vec{\mathbf{p}}^{n+1}=\vec{\vec{\mathbf{B}}}(\rho)\hat{\hat{\mathbf{V}}}^{n+1}-\bar{\bar{\mathbf{V}}}^{n+1}+\vec{\vec{\mathbf{S}}}(\tau)\vec{\mathbf{p}}^{n} \tag{3.22}$$

After solving (3.21) the pressure at time $t=t^{n+1}$ may be evaluated.

The last step is the evaluation of the final velocity at time $t=t^{n+1}$. Using (2.24):

$$\rho_p\vec{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\hat{\hat{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})-\frac{\Delta t}{2}\delta\Pi^{n+1}(\mathbf{x}_p^{n+1}) \tag{3.23}$$

The Vector $\delta\vec{\Pi}^{n+1}$ may be calculated using (3.5). In matrix notation:

$$\int_\Omega\vec{\vec{\mathbf{N}}}\,\nabla\vec{\mathbf{N}}_p^T\,d\Omega\;\delta\vec{\mathbf{p}}^{n+1}=\int_\Omega\vec{\vec{\mathbf{N}}}\,\vec{\vec{\mathbf{N}}}^T\,d\Omega\,\delta\vec{\Pi}^{n+1} \tag{3.24}$$

or

$$\delta\Pi^{n+1}=(\vec{\vec{\mathbf{M}}})^{-1}\vec{\vec{\mathbf{D}}}\,\delta\vec{\mathbf{p}}^{n+1} \tag{3.25}$$

with

$$\vec{\vec{\mathbf{D}}}=\int_\Omega\vec{\vec{\mathbf{N}}}\,\nabla\vec{\mathbf{N}}_p^T d\Omega \tag{3.26}$$

The stabilization parameter used here may be found in several publications [19,20,21,22].

As a resume, the Particle Finite Element Method with the X-IVAS integrations with fixed mesh reads:

I) Evaluate the viscous and pressure gradients vector ($\vec{\mathbf{g}}^n-\vec{\Pi}^n$):

$$\vec{\mathbf{g}}^n-\vec{\Pi}^n=(\vec{\vec{\mathbf{M}}})^{-1}(\vec{\vec{\mathbf{K}}}\vec{\mathbf{V}}^n-\vec{\vec{\mathbf{B}}}\,\vec{\mathbf{p}}^n-\vec{\sigma}^n)$$

II) Evaluate at each particle the fractional velocity $\hat{\hat{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})$ and the new particle position $\mathbf{x}_p^{n+1}$:

$$\begin{cases}\mathbf{x}_p^{n+t}=\mathbf{x}_p^n+\int_n^{n+t}\mathbf{V}^n(\mathbf{x}_p^\tau)\,d\tau\\ \rho_p\hat{\hat{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})=\rho_p\mathbf{V}^n(\mathbf{x}_p^n)+\int_n^{n+1}\vec{\vec{\mathbf{N}}}^T(\mathbf{x}_p^t)\,dt\,\vec{\mathbf{g}}^n-\int_n^{n+1}\vec{\vec{\mathbf{N}}}^T(\mathbf{x}_p^t)\,dt\,\vec{\Pi}^n+\int_n^{n+1}\mathbf{b}^n(\mathbf{x}_p^t)\,dt\end{cases}$$

III) Project the fractional velocity $\hat{\hat{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})$ on a vector $\delta\hat{\hat{\mathbf{V}}}^{n+1}$:

$$\hat{\hat{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1})-\mathbf{V}^n(\mathbf{x}_p^{n+1})\Rightarrow\delta\hat{\hat{\mathbf{V}}}^{n+1}(\mathbf{x})$$

IV) Solve implicitly the linear system of equations to obtain $\delta\hat{\hat{\mathbf{V}}}^{n+1}$

$$[\mathbf{M}(\rho)+\mathbf{K}\frac{\Delta t}{2}]\delta\hat{\hat{\mathbf{V}}}^{n+1}=\mathbf{M}(\rho)\delta\hat{\hat{\mathbf{V}}}^{n+1}$$

V) Solve implicitly the linear system of equations to obtain the pressure increment $\delta\vec{\mathbf{p}}^{n+1}$ on the nodes of the fixed mesh:

$$[\frac{\Delta t}{2}\vec{\vec{\mathbf{L}}}+\vec{\vec{\mathbf{S}}}(\tau)]\delta\vec{\mathbf{p}}^{n+1}=\vec{\vec{\mathbf{B}}}(\rho)\hat{\hat{\mathbf{V}}}^{n+1}-\bar{\bar{\mathbf{V}}}^{n+1}+\vec{\vec{\mathbf{S}}}(\tau)\vec{\mathbf{p}}^{n}$$

VI) Evaluate the new pressure gradients vector $\delta\vec{\Pi}^{n+1}$:

$$\delta\vec{\Pi}^{n+1}=(\vec{\vec{\mathbf{M}}})^{-1}\vec{\vec{\mathbf{D}}}\,\delta\vec{\mathbf{p}}^{n+1}$$

VII) Evaluate at each particle the new velocity $\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})$:

$$\rho_p \vec{\mathbf{V}}^{n+1}(\mathbf{x}_p^{n+1}) = \rho_p \hat{\vec{\mathbf{V}}}^{n+1}(\mathbf{x}_p^{n+1}) - \frac{\Delta t}{2}\delta\Pi^{n+1}(\mathbf{x}_p^{n+1})$$

VIII) Evaluate at each node of the mesh the new velocity vector $\vec{\mathbf{V}}^{n+1}$:

$$\vec{\mathbf{V}}^{n+1} = \hat{\vec{\mathbf{V}}}^{n+1} - \delta\vec{\Pi}^{n+1}\frac{\Delta t}{2\rho(\mathbf{x})}$$

IX) Evaluate at each node of the mesh the new pressure vector $\vec{\mathbf{p}}^{n+1}$

$$\vec{\mathbf{p}}^{n+1} = \vec{\mathbf{p}}^{n} + \delta\vec{\mathbf{p}}^{n+1}$$

X) Increase the time $t^{n+1} = t^n + \Delta t$ **and update the variables at the nodes:**

$$\vec{\mathbf{V}}^{n} \Leftarrow \vec{\mathbf{V}}^{n+1}$$

$$\vec{\mathbf{p}}^{n} \Leftarrow \vec{\mathbf{p}}^{n+1}$$

Update also the velocity at the particles:

$$\mathbf{V}^{n}(\mathbf{x}_p^{n}) \Leftarrow \mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})$$

go to I)

It must be noted that for the case of homogeneous fluids (e.g. $\mu = \text{constant}$ and $\rho = \text{constant}$) with fixed mesh, the matrixes $\mathbf{K},\mathbf{M},\mathbf{L},\mathbf{B}$ and $\mathbf{D}$ are constant during all the time increments. However, for heterogeneous fluid flows, some of them (e.g. $\mathbf{K}(\mu),\mathbf{M}(\rho),B(\rho)$ and $\mathbf{S}(\tau)$) may be necessary to re evaluated at each time step.

4. Numerical examples

This section is devoted to show by numerical examples the reliability and the efficiency of the novel method presented in [1]. While in this former paper the moving mesh version was specially treated, in this opportunity the emphasis was focused on the fixed mesh version of the code. The examples chosen were the circular cylinder and a typical NACA 0012 in aeronautic engineering. Even though the formulation is also implemented in 3D as shown in [1] here only 2D simulations are shown. A similar 3D analysis will be performed in the future. In order to prove the accuracy and performance of the proposed method, which represent the two main goals of this paper, we include for each example two sub-sections, one for accuracy and the other for CPU times. In both examples and in all typical cases of hydrodynamics around bodies it is very important to compute the forces produced by the fluid because they are the input for several other engineering computations and design. These forces are typically the drag or the resistance force and the lift and also the moments. Here the accuracy estimate will be the drag and the lift.

4.1 Flow around a circular cylinder at Re=1000.

This example involves the flow past a circular cylinder, another widely solved benchmark problem. The computational domain is rectangular and extended 5 cylinder diameters upwards and in both transverse directions and 15 diameters downwards, with the cylinder diameter D of 1 and centered at the origin.

The velocity at the left rectangle side (the inlet) is prescribed to the free stream velocity that here is of unit magnitude in the horizontal direction. Also in both top and bottom rectangle sides is imposed to the free stream vector value, whereas at the cylinder perimeter is fixed to a non-slip boundary condition The outflow (where both

the x- and y-components of the velocity are free) the pressure is fixed to a reference value, here taken as zero. The Reynolds number is 1000, based on the cylinder diameter and on the prescribed inflow velocity.

The finite element mesh employed consists of 88000 linear triangles, with 44520 nodal points, being refined near the cylinder.

The results achieved here were compared against those got by Mittal [6].

The number of particles inside each mesh element is controlled to be in the range of 1 to 3 by sub-element. The sub-element is defined as the region inside each triangle round each vertex, i.e. there are three sub-elements per triangle, defined as the region bounded by the vertex, both half edges converging at the vertex and the centroid of the triangle. In this way a more controlled interpolation error is obtained.

Accuracy Results:

First the drag and the lift obtained compared against Mittal results are introduced. Fig. 4.1 shows the drag where a good agreement in the oscillation frequency is observed with a drag mean value 10 percent above the reference mean value. The amplitude of this oscillated drag is in good agreement with the reference.

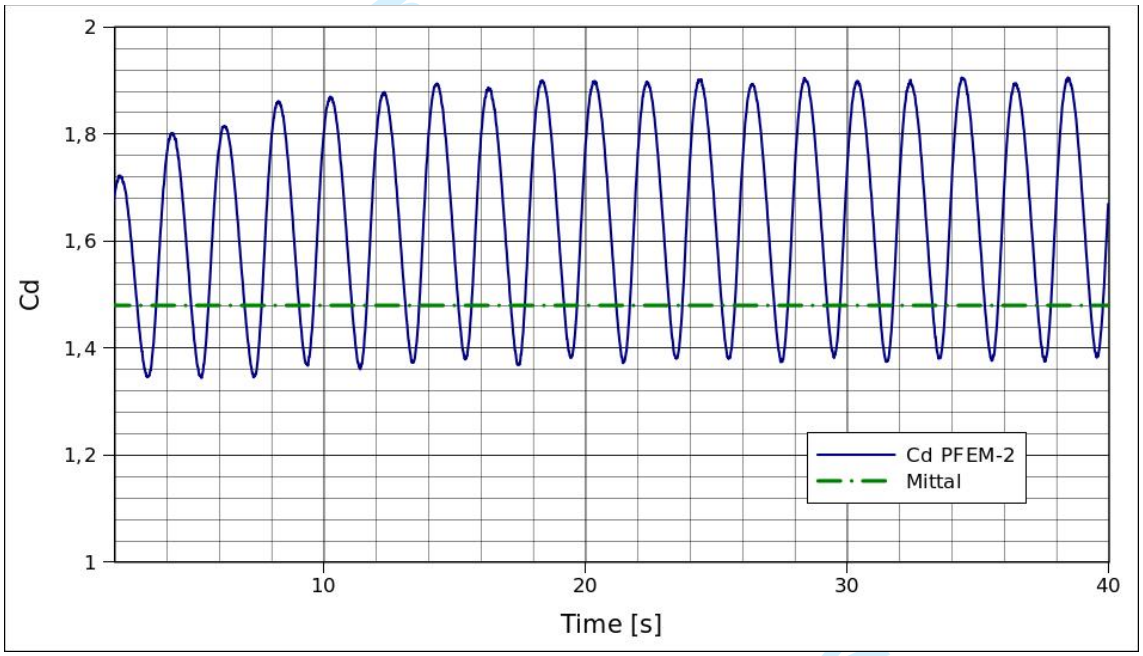

Fig. 4.1: *Drag Coefficient Re=1000. Mittal's dotted line represents* $\overline{C_d}$

Although this is out of scope in the current paper, we would like to observe that the method used in projecting from the particles to the mesh affects to some extent the level of "noise" in the lift and drag evaluation. This aspect will be discussed in detail in a future publication.

The lift force oscillates as expected around the zero value with an amplitude is 20 percent above the reference value. As we will show later they are however in a very close agreement with the predictions of the OpenFOAM® code [8].

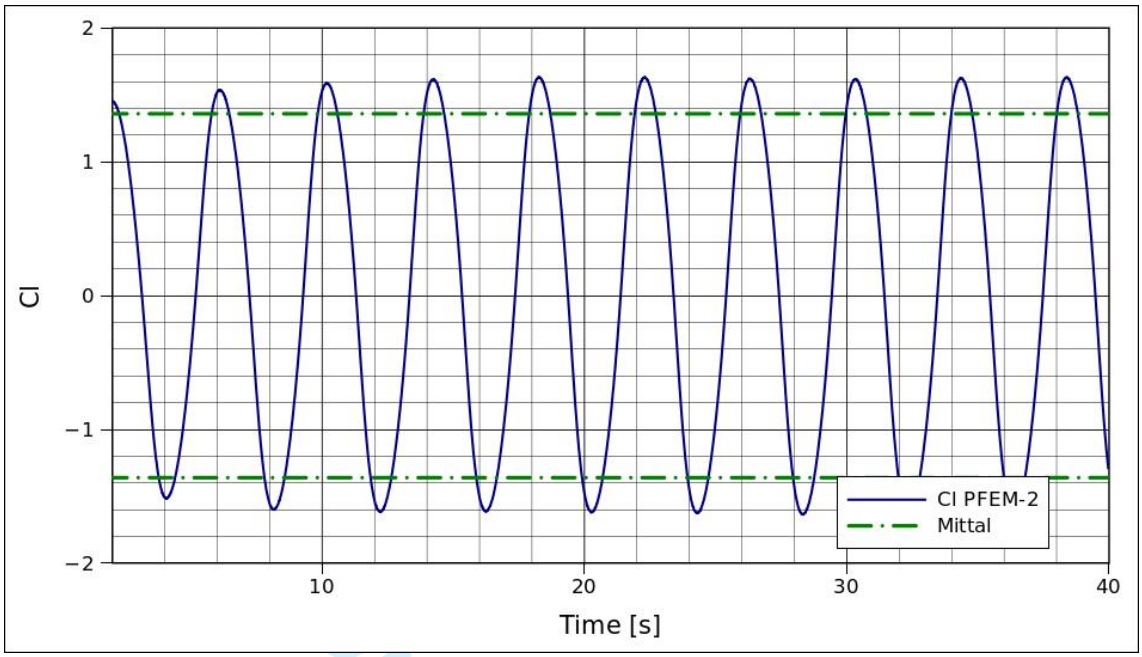

Fig. 4.2: *Lift Coefficient Re=1000*
Mittal's dotted line represents max and min values of $\overline{C_L}$

In order to check the performance of the method presented here a fast CFD suite called OpenFOAM® [8] was taken. Before going in CPU time details the relative accuracy of *PFEM-2* against OpenFOAM ® is presented.

Fig. 4.3 shows the drag obtained using OpenFOAM® with the best set of parameters to reach a good solution. It follows the same tendency of *PFEM-2* using the smooth version of our interpolation from particle to mesh values.

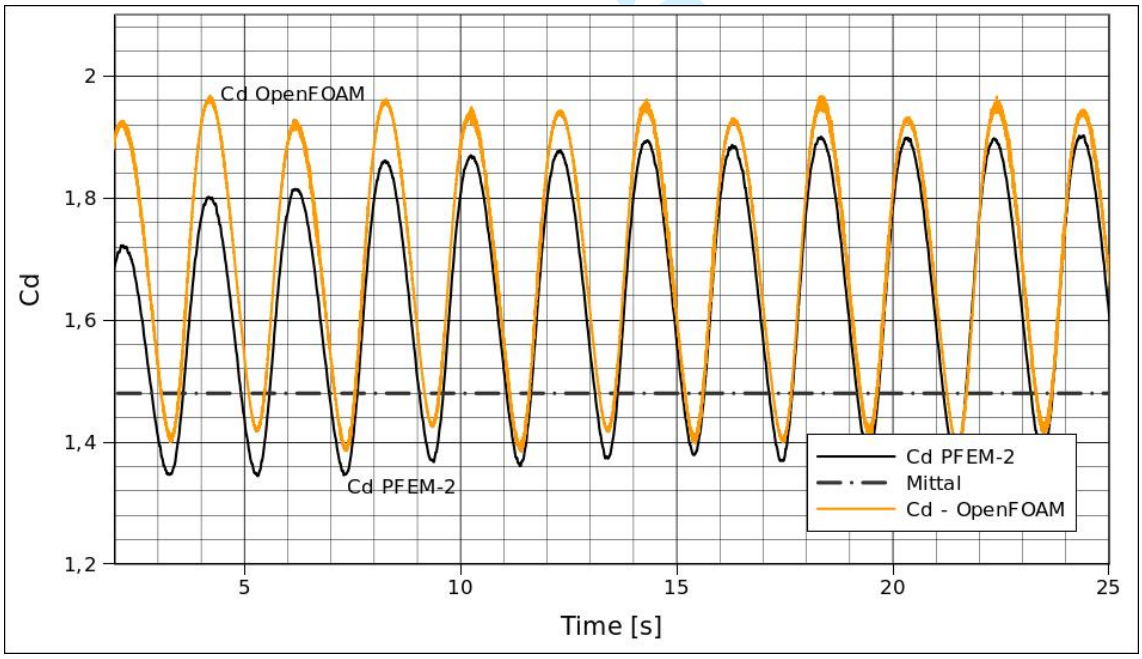

Fig. 4.3: *Comparison Drag Coefficient Re=1000 PFEM-2 vs OpenFOAM®*

Next, the lift is presented with similar conclusions.

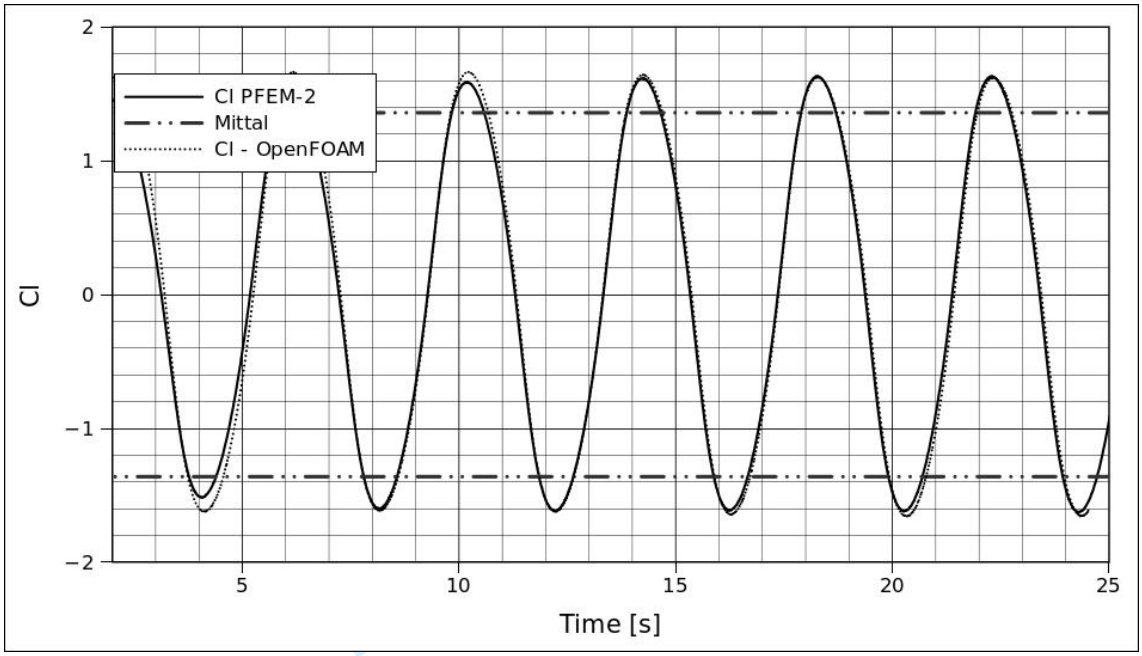

Fig. 4.4: *Comparison Lift Coefficient Re=1000 PFEM-2 vs OpenFOAM®*

The following table summarizes the results obtained where a relative good agreement is achieved.

	Strouhal	$\overline{C_d}$	C_d *Amplitude*	C_L *Amplitude*
Mittal	0.25	1.48	0.21	1.36
PFEM-2	0.2475	1.639	0.245	1.63
OpenFOAM	0.26	1.696	0.25	1.62

Table 4.1: *A brief summary of the comparison among OpenFOAM®, the reference [6] and PFEM-2*

Time Results:

The OpenFOAM® solution was used as a benchmark for the computational time. One second of simulation time was used as the reference interval. According to the good of PFEM-2 to enlarge the time step here a close to 10 for maximum Courant was reached for fixed mesh version with implicit diffusion.

The other versions were used with a more conservative time step. OpenFOAM® has a limit for the maximum Courant number due to nonlinear instabilities that would appear if time step is too long. This maximum Courant number was around 1. In order to get accurate solutions 2 PISO corrections and 2 non-orthogonal corrections were used, with a tolerance about 10^{-6} . Here a summary of the main parameters is included.

- CPU: Intel I7-2600K
- Real Time Simulated: 1 second
- PFEM-2 Fixed Mesh Implicit Diffusion: $\Delta t = 0.025$ [sec]
  (Mean Courant 2, Maximum Courant 9.5)
- PFEM-2 Fixed Mesh Explicit Diffusion: $\Delta t = 0.0125$ [sec]
- PFEM-2 Moving Mesh: $\Delta t = 0.01\$$ [sec]
- OpenFOAM®: $\Delta t = 0.002$ [sec]
  - Solver: icoFoam

- Selected maximum Δt to get Maximum Courant = 1.
- Number of PISO Correctors: 2
- Number of Non Orthogonal Correctors: 2
- U tolerance: $\epsilon_U = 10^{-6}$
- p Tolerance: $\epsilon_p = 10^{-6}$

Fig.4.5 shows that PFEM-2 is around 5 times faster than OpenFOAM® using 4 cores with a scalability of almost 3 times instead of the ideal 4 times (efficiency about 75 percent). At the end of this section a summary of profiling and scalability of *PFEM-2* is presented.

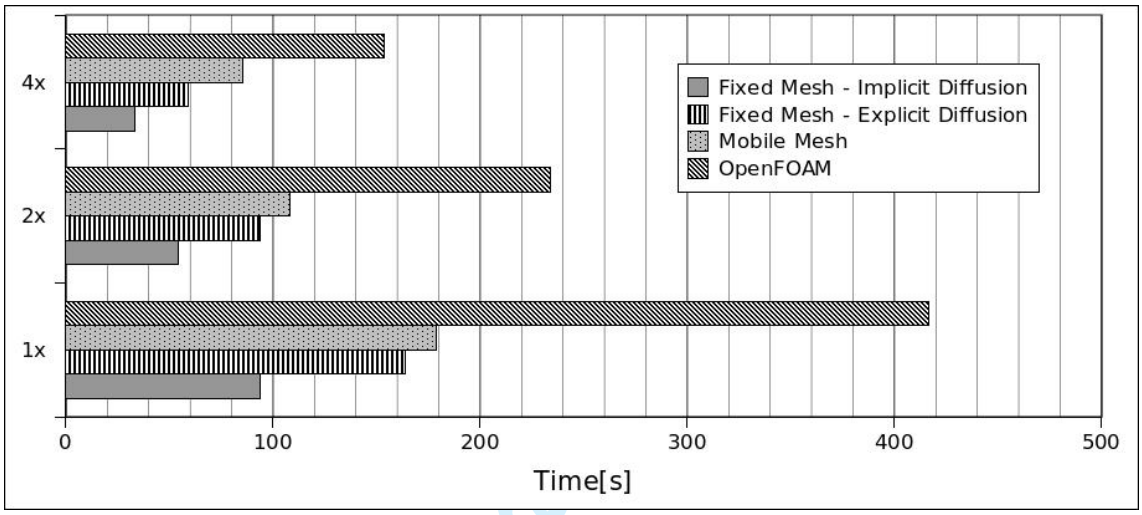

Fig. 4.5: *Cylinder - Re=1000*
CPU-Times comparison between different PFEM-2 algorithms and OpenFOAM® for 1, 2 and 4 cores.

4.2 Flow around a NACA 0012 airfoil

The second example to show the capability of PFEM-2 as a CFD code for engineering applications is the popular NACA 0012 airfoil. This aeronautical profile has been vastly analyzed and huge information about it characteristics has been published.

Here a low and medium Reynolds numbers were chosen in order not to suffer the lack of turbulence modeling. Two several angle of attacks were evaluated. The first is a 6 millions Reynolds number flow at zero angle of attack and the second one was a four degree angle of attack with Reynolds 10 thousands.

The airfoil chord is of unit length and the computational domain is large in order not to interfere the boundary conditions with the airfoil. The computational domain and a detail of the mesh for the latter case are shown in the next figure. For the former case the domain and the mesh employed were similar.

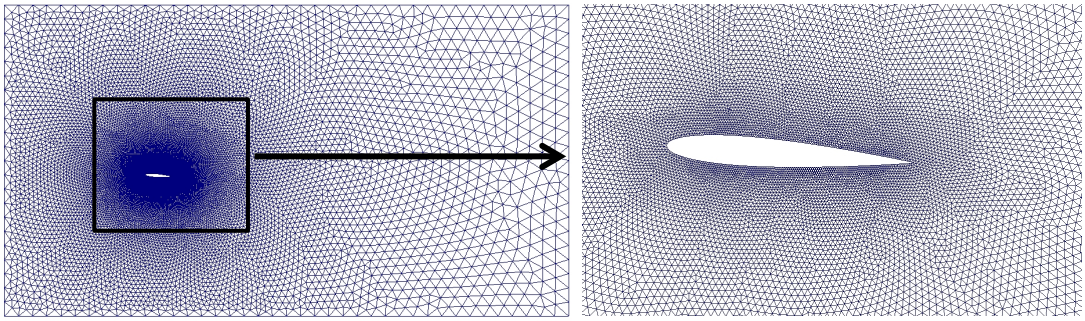

Fig. 4.6: *Computational domain and mesh for four degrees of angle of attack. General view*

NACA 0012 at zero angle of attack and Re= 6 millions:

The reference for this test was the experimental set-up reported in [5].

37568 linear triangles and 18989 nodes compose the mesh used with approximately 212000 particles along the entire domain.

The mean pressure coefficient compared with the reference value is shown in fig. 4.7. This coefficient is obtained as the time average of the ratio between the relative pressure and the reference dynamic pressure as:

$$\overline{C_p} = \frac{\bar{p} - p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2}$$

As it is shown a general good agreement is achieved.

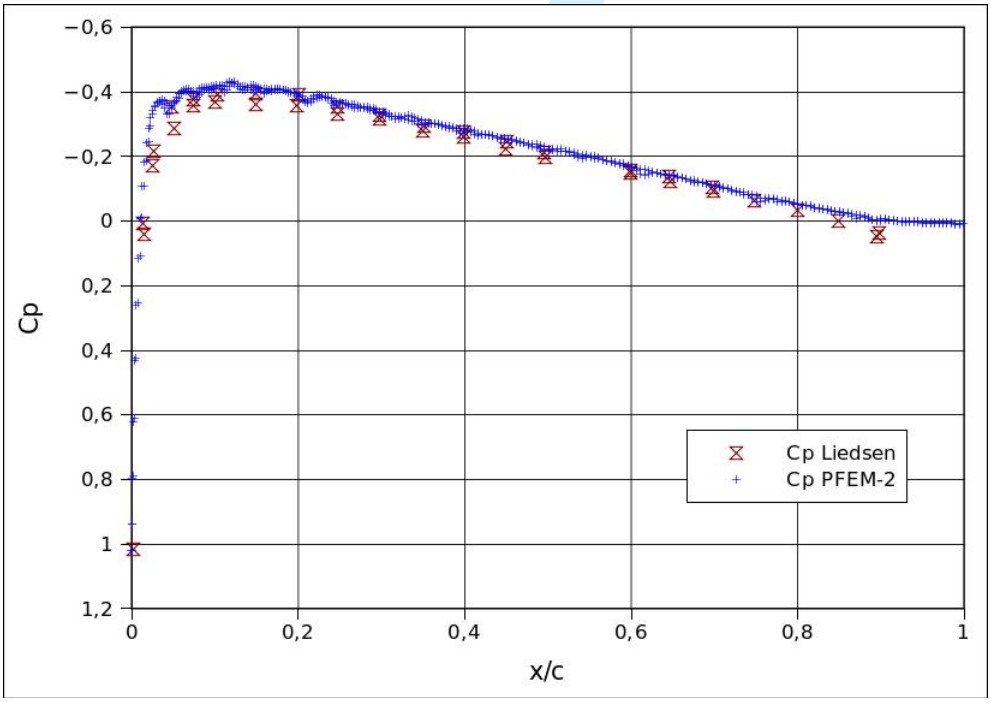

Fig. 4.7: *Mean Pressure Coefficient for Re = 6 million.*

The drag is shown in Fig. 4.8 where the numerical results in average follows the experimental value relatively good, with an 8 percent of difference.

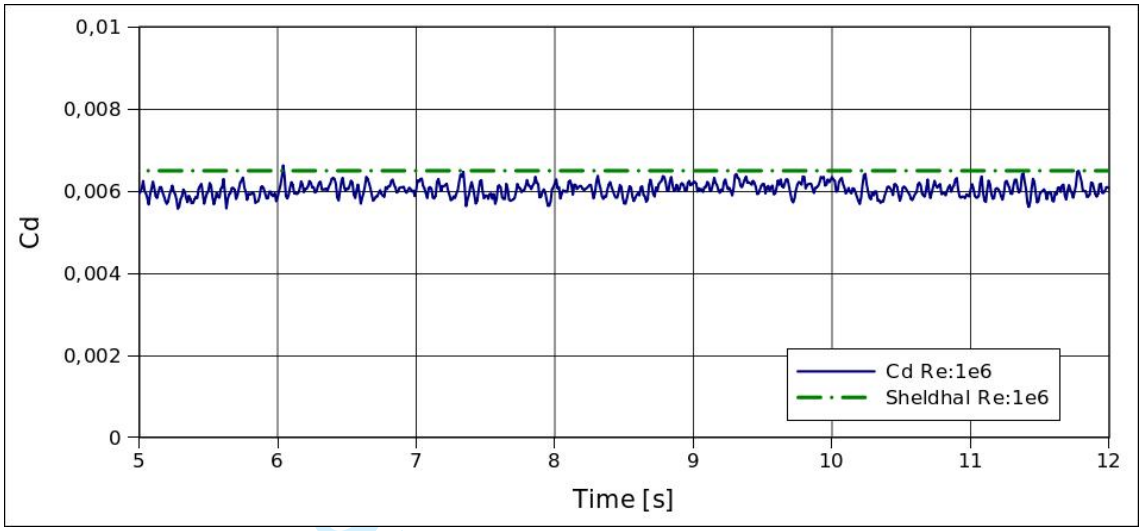

Fig. 4.8: *Drag Coefficient for Re=6 million Sheldhal's dotted line represents* $\overline{C_d}$

The lift in Fig. 4.9 oscillate around zero, as experimentally is observed with a small error.

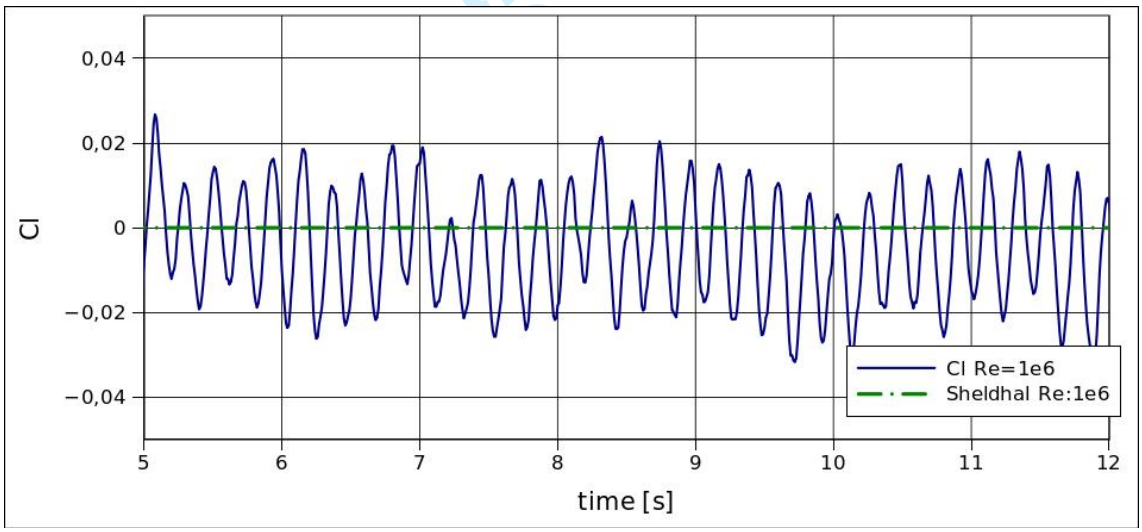

Fig. 4.9*: Lift Coefficient for Re=6 million.*

In general the results look acceptable for engineering computations especially if numerical results do not include any high Reynolds effect. As future work some turbulence modeling is needed to add.

NACA 0012 at 4 degrees angle of attack and Re= 10 thousands:

This example allows enforce the results obtained in the previous sub-section especially when the angle of attack is different from zero as it is normally expected in aeronautical and wind turbine applications. Here the reference is [4]. The mesh is similar to that defined above.

Accuracy results:

The results in terms of accuracy show that the mean pressure coefficient follows the reference results in an acceptable way as it is shown in Fig. 4.10

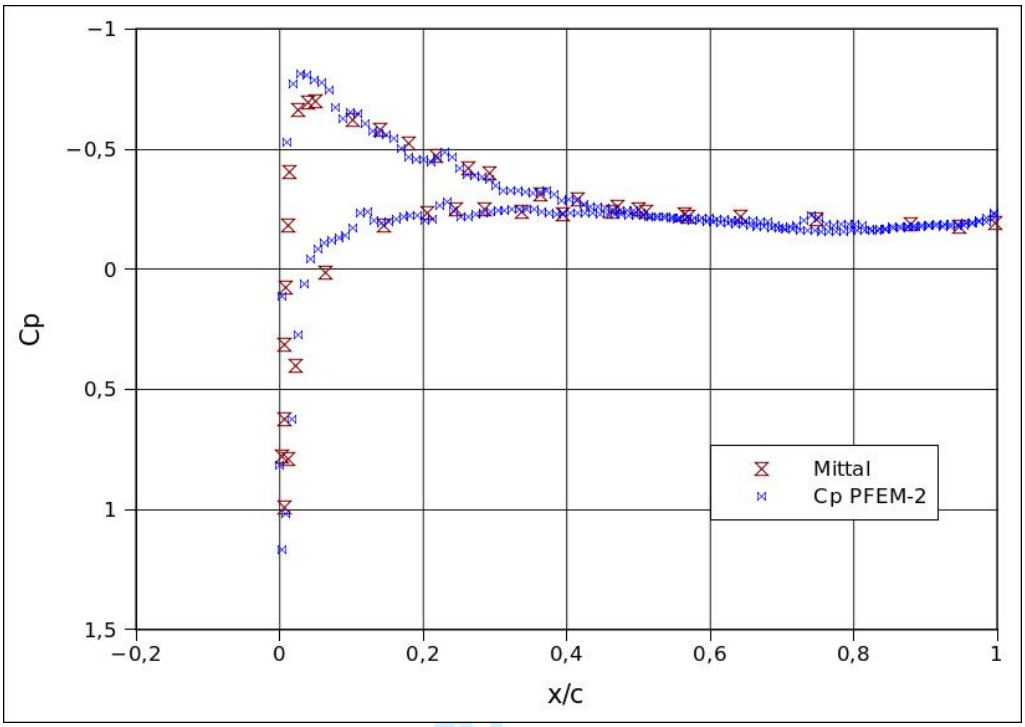

Fig. 4.10: *Mean Pressure Coefficient for Re=10 thousands*

The lift is bounded by the limits taken from the reference as Fig.4.11 plots.

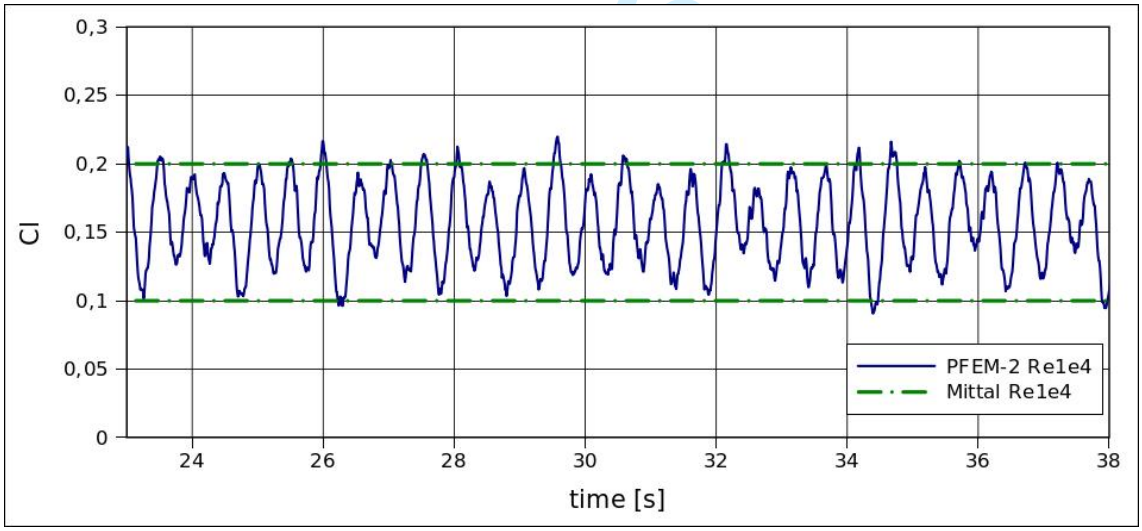

Fig. 4.11: *Lift Coefficient for Re=10 thousands.*
Mittal's dotted line represents max and min values of Lift coefficient.

The drag is again under predicted with its amplitude similar to that published in the reference but with a mean value that is about 9 percent away from that.

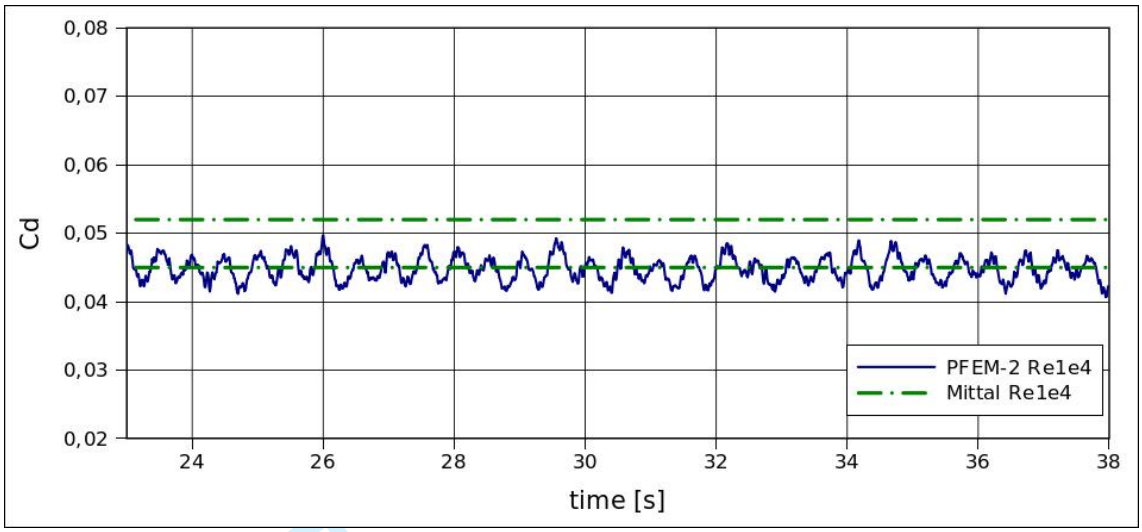

Fig. 4.12 : *Drag Coefficient for Re=10 thousands.*
Mittal's dotted line represents max and min values of Drag coefficient.

In the Fig- 4.13 some snapshots about the vortex shedding is presented. The vortex emission has a frequency of 2.1 Hz that compared with that of 2.3 Hz reported in the reference represents a 10 percent of difference.

Time Results:

As regards with the efficiency for both NACA 0012 tests the present method is again faster than OpenFOAM®, the code taken as the reference for CPU time measurements. In this case the factor of speed up is around 3 instead of 5 obtained for the cylinder test. It is expected that this factor increases with large angle of attack due to an increasing of the unsteadiness of the flow that introduces more restriction on the time step for OpenFOAM® due to the nonlinearities with no significant restrictions on our lagrangian method (PFEM-2).

Again, the simulation features used in the comparison of performance were the following:

- CPU: Intel I7-2600K
- Real Time Simulated: 1 second
- PFEM-2 Fixed Mesh Implicit Diffusion: $\Delta t = 0.015$ [sec]
- PFEM-2 Fixed Mesh Explicit Diffusion: $\Delta t = 0.01$ [sec]
- OpenFOAM®: $\Delta t = 0.002$ [sec]
    - Solver: icoFoam
    - Selected maximum Δt to get Maximum Courant = 1.
    - Number of PISO Correctors: 2
    - Number of Non Orthogonal Correctors: 2
    - U tolerance: $\epsilon_U = 10^{-7}$
    - p Tolerance: $\epsilon_p = 10^{-7}$

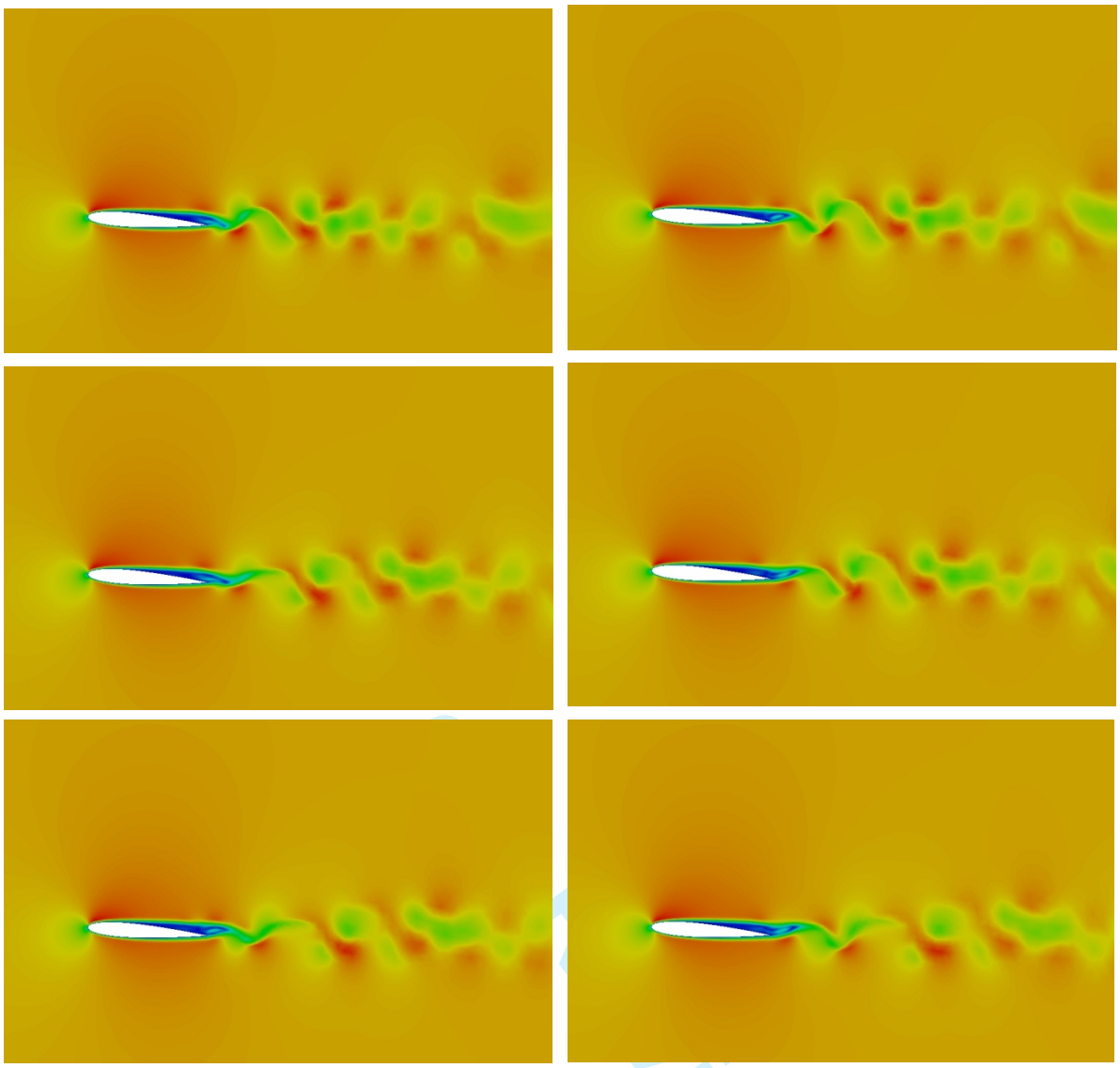

Fig. 4.13: *Snapshots of vortex shedding for NACA 0012 airfoil at four degrees of angle of attack and Re 10 thousands*

Fig. 4.14 shows the CPU time comparison. Regards the profiling of the present method the situation is graphically explained in the next Fig 4.15 where the streamline integration consumes more than half the time. Being this stage of the code highly scalable it is desired that this portion increase. Other parts that have the same feature are the acceleration computation consuming approximately 6 percent and the correction stage consuming 9 percent.

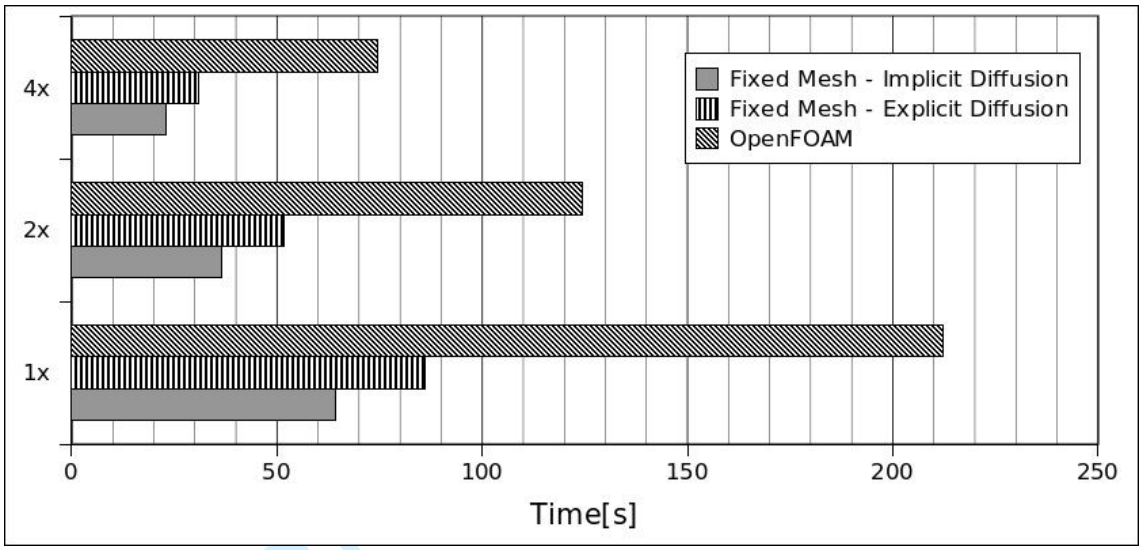

Fig. 4.14: *NACA0012 – 4 degrees of attack angle and Re=10 thousands. CPU-times comparison between different PFEM-2 algorithms and OpenFOAM® for 1,2 and 4 cores.*

Therefore, the more scalable part of the code sums about 65 percent of the total computational cost. The remaining part of the code is formed by the Updating/Projection and the implicit part, i.e. Poisson solver and the diffusive part of the momentum equation, here solved using an implicit scheme in order not to be restricted by Fourier number stability considerations.

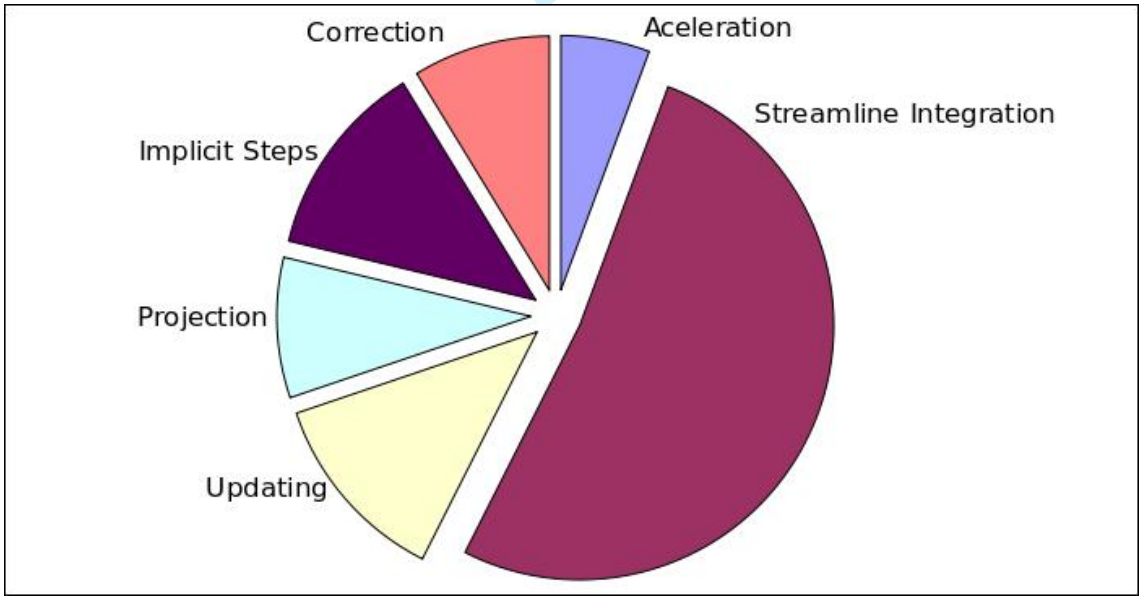

Fig. 4.15: *Time proportion for each stage of the algorithm (PFEM-2 Fixed-Mesh Implicit Diffusion)*

Looking at the performance measured in terms of the scalability (Fig. 4.16) it is noted that the streamline integration stage is almost ideal while the acceleration and the correction stages look in the middle range. The three remaining parts, the implicit, updating and projection have some limitations in being scalable from their inherent structure of computation. The updating is hard to speed up due to it requires a strong

memory management with removing, adding and reordering memory storage. For the implicit solution is well known the difficulties for improving this item. Here its impact is overestimate because we adopt for the viscosity contribution to the momentum equation an implicit treatment. Perhaps an adaptive solution where implicit viscous contribution are localized around solid bodies where refinement tends to drastically limit the time step due to Fourier number stability could improve this situation in the future.

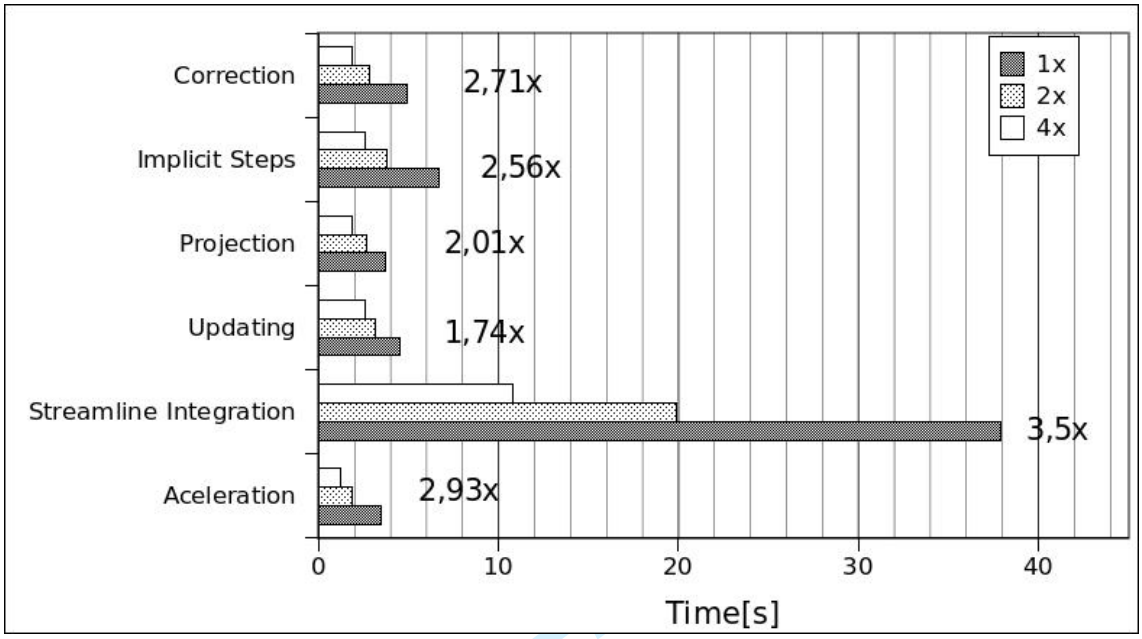

Fig. 4.16: *Scalability for each stage of the algorithm PFEM-2 Fixed-Mesh with Implicit Diffusion*

5. Conclusions and future work

A new version of the novel method *PFEM-2* originally introduced in [1] was presented in this paper.

As it was mentioned above the restrictions on performance imposed by the parallel re-meshing and the assembling/solver stages of the implicit part of the moving mesh version of *PFEM-2* has motivated to review the method incorporating this fixed mesh version that mainly has the advantages of getting away the re-meshing stage and reducing significantly the cost of the implicit part due to the freezing of the factorized matrices involved in those computations.

This new version shows to reduce even more the CPU time as compared with a popular and fast CFD code OpenFOAM® reaching at present a very attractive position in terms of accuracy and efficiency.

The ideas behind this method are constantly in evolution but the present status of the method allows qualifying as one of the fastest method for CFD.

Some particular conclusions are:

- $\overline{C_p}$, C_d and C_L curves are reasonable and have good shape and values according to engineering needs. These results were compared with other software (OpenFOAM®), other numerical results (Mittal [6], Ladson [7], Park [3]) and other experimental results (Sheldhal [5]), obtaining good approximations in all cases.

- Probably fluid instabilities previous to fully developed turbulence and turbulent effects might introduce some changes in the vortex dynamics that currently our *PFEM-2* method is unable to capture. This consideration may justify the small differences observed in our results.
- The performance on the PFEM-2 depends on the choice of the particle-to-mesh transfer operator. No "natural" choice exists in performing this step and hence different possibilities are open for testing. This aspect will be subject of extensive future research
- The efficiency measured in terms of CPU time for reaching a given final time in the simulation (here takes as 1 second) had shown important advantages of the present method against OpenFOAM® being this new fixed mesh version even more fast than the previous moving mesh one. In this paper a factor between 3 and 5 was obtained at the same level of accuracy.
- In terms of scalability the present method has in general a 75 percent of efficiency compared with 65 percent of scalability produced by OpenFOAM®. This 75 percent is achieved with an almost ideal (100 percent) speed-up of the streamline computation reduced by the poor scalability of implicit computations that in this method tends to have a small impact and some other memory operations that are inherently inefficient.

Future work will be oriented to the following:

- Extend this method to multi-physics, especially thermally driven flows, RANS turbulence modeling and multispecies computations. This task is very straightforward and will give the method good reputation for a wider engineer community.
- In terms of efficiency the shared memory parallelism is limited by hardware, therefore a distributed memory parallel code should be developed.
- The simulation of multi-fluids is another target but it needs much more work of design and probably a complete review of the current algorithm.

Acknowledgements

This work was partially supported by the European Research Council under the Advanced Grant: ERC-2009-AdG ''Real Time Computational Mechanics Techniques for Multi-Fluid Problems''. Norberto Nigro and Juan Gimenez want to thanks to CONICET, Universidad Nacional del Litoral and ANPCyT for their financial support (grants PICT 1645 BID (2008), CAI+D 65-333 (2009)). Riccardo Rossi acknowledges the support of the MEC through the eDams project (BIA2010-21350-C03-00).

References

1. Idelsohn, S.R., Nigro N.M., Limache A. and Oñate, E. (2012) "Large time-step explicit integration method for solving problems with dominant convection", *Comp. Meth. in Applied Mechanics and Engineering*, 217-220, PP 168-185.
2. Idelsohn, S.R., Oñate, E. and Del Pin, F. (2004). "The particle finite element method a powerful tool to solve incompressible flows with free-surfaces and breaking waves". *Int. J. Num. Meth. Engng.,* Vol. 61, pp. 964-989.
3. Park, J (1998) "Numerical Solutions of flow past a circular cylinder at Reynolds Numbers up to 160", *KSME international Journal*, Vol 12, No 6, pp 1200-1205
4. Srinath D.N. and Mittal S. (2010), "Optimal aerodynamic design of airfoils in unsteady viscous flows", *Computer Methods in Applied Mechanics and Engineering*, Vol. 199, pp.1976-1991

5. Sheldhal, R.E.and Klimas, P.C. (1981), "Aerodynamic Characteristics of Seven Airfoil Sections Through 180 Degrees Angle of Attack for Use in Aerodynamic Analysis of Vertical Axis Wind Turbines", *SAND80-2114, Sandia National Laboratories, Albuquerque, New Mexico*
6. Mittal S. and Kumar V. (2001), "Flow-Induced vibrations of a light circular cylinder at Reynolds numbers 1000 to 10000*", Journal of sound and vibration*, Vol. 254, No. 5 pp. 923-946.
7. Ladson, C. L., (1988) "Effects of Independent Variation of Mach and Reynolds Numbers on the Low-Speed Aerodynamic Characteristics of the NACA 0012 Airfoil Section," NASA TM 4074
8. OpenCFD, (2009). OpenFOAM®, The Open Source CFD Toolbox, User Guide. OpenCFD Ltd.
9. Mossaiby, F.; Rossi, R.; Dadvand, P.; Idelsohn, S.R, (2011),."OpenCL-based implementation of an unstructured edge-based finite element convection-diffusion solver on graphics hardware", *International journal for numerical methods in engineering,* in press
10. Dadvand, P.; Rossi, R.; Oñate, E.(2010) "An object-oriented environment for developing finite element codes for multi-disciplinary applications". *Archives of computational methods in engineering*.Vol. 17, No. 3, pp. 253 – 297
11. Dadvand, P.; Rossi, R.; Gil, Marisa; Martorell, X.; Cotela, J.; Juanpere, E.; Idelsohn, S.R.; Oñate, E , (2012),. "Migration of a Generic Multi-Physics Framework to HPC Environments", *Computers and fluids*., in press
12. Oñate, E.; Idelsohn, S.R.; Celigueta, M.A.; Rossi, R ,(2008),."Advances in the particle finite element method for the analysis of fluid–multibody interaction and bed erosion in free surface flows",. *Computer methods in applied mechanics and engineering*.Vol. 197, No.:19-20 ,pp. 1777 - 1800.
13. Idelsohn S.R., Marti J., Limache A. and Oñate E., (2008) "A unified Lagrangian formulation for elastic solids and incompressible fluids. Aplication to fluid- structure interaction problems via the PFEM", *Computer Methods in Applied Mechanics and Engineering*,,Vol 197, pp 1762-1776
14. Idelsohn S.R., Marti J., Souto-Iglesia A. and Oñate, E., (2008), "Interaction between an elastic structure and free-surface flows: experimental versus numerical comparisons using the PFEM", *Computational Mechanics*, Vol 43, pp 125-132
15. Larese, A.; Rossi, R.; Oñate, E.; Idelsohn, S., (2008), "Validation of the particle finite element method (PFEM) for simulation of free surface flows", *Engineering computations,*.Vol. 25, No. 4, pp. 385 – 425
16. Ryzhakov, P.; Rossi, R.; Idelsohn, S.R.; Oñate, E. (2010), "A monolithic Lagrangian approach for fluid–structure interaction problems"., *Computational mechanics*., Vol. 46, No. 6, pp. 883 - 899
17. Ryzhakov, P.; Oñate, E.; Rossi, R.; Idelsohn, S.R., (2010), "Improving mass conservation in simulation of incompressible flows", *.International journal for numerical methods in engineering*.
18. Donea, J. and Huerta, A., (2003), Finite Element Methods for Flow Problems, J. Wiley, ISBN: 0-471-49666-9
19. Hughes T. and Tezduyar, T. (1984), "Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations", *Computer Methods in Applied Mechanics and Engineering,* Vol. 45 , pp. 217–284
20. Tezduyar, T. Mittal, S, Ray, S. and Shih, R., (1992), "Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements", *Computer Methods in Applied Mechanics and Engineering*, Vol. 95, pp. 221–242
21. Badia, S. Codina, R. (2008), "Pressure segregation methods based on a discrete pressure Poisson equation. An algebraic approach", *International Journal for Numerical Methods in Fluids*, Vol. 56, pp. 351-382
22. Codina R., Principe J., Guasch O. and Badia S., (2007), "Time dependent subscales in the stabilized finite element approximation of incompressible flow problems", *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, 2413-2430