Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXI, págs. 137-155
Alberto Cardona, Paul H. Kohan, Ricardo D. Quinteros, Mario A. Storti (Eds.)
Salta, Argentina, 13-16 Noviembre 2012

IMPROVEMENTS TO SOLVE DIFFUSION-DOMINANT PROBLEMS WITH PFEM-2

Juan M. Gimenez[a], **Norberto M. Nigro**[a,b] **and Sergio Idelsohn**[a,c]

[a]*International Center for Computational Methods in Engineering (CIMEC), INTEC-UNL/CONICET, Guemes 3450, Santa Fe, Argentina, http://www.cimec.org.ar*

[b]*Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral. Ciudad Universitaria. Paraje "El Pozo". Santa Fe. Argentina. http://www.fich.unl.edu.ar*

[c]*ICREA Research Professor at the International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain.http://www.cimne.edu.es/*

Keywords: particle, tracking, turbulence,CFD.

Abstract. Particle Finite Element Method - Second generation (PFEM-2) is a method characterized by using both particles and mesh to solve physics equations.

In some previous papers the mathematical and numerical basis of the method with also some results were presented showing a good accuracy and high performance for solving scalar-transport and momentum-transport problems with dominant advection term.

To preserve accurate solutions on the mesh, the method must create or remove particles to have enough information to solve diffusive terms. In these problems and under certain types of velocity fields and/or mesh, some cautions at the particle-updating step must be taken into account to avoid an excessive numerical diffusion in the solutions. In this way, an improved approach to create and remove particles is presented here.

Moreover, when the problem is diffusion-dominant, the advantages of the method are not as clear as in the advection-dominant cases. The explicit calculation of the diffusion traditionally used by PFEM-2 is limited by the dimensionless Fourier number and, in some particular cases, the method may fail. To relax these restrictions, a new model to calculate the diffusion is developed. It is based on the *theta method* which consists on discretizing the non-stationary variable using a weighted mixture between an explicit prediction and an implicit correction. This approach extends the stability limit, while reducing the error, and it could also allow the usage of longer time steps. Even though the implicit may be computationally expensive, exploiting the possibility of factorizing the matrix at the beginning and using it as a preconditioner for solving the diffusion step had shown to have good performance.

To validate the solutions proposed here some tests are solved comparing their different results with the analytic ones. This report is focused on showing how the new models improve the accuracy of the solutions of the older ones and how the computing times are modified.

1 INTRODUCTION

Formulations for transport equations may be split in two classes depending on the approach chosen for the description of the inertial terms, namely Eulerian and Lagrangian approaches. The simulation of incompressible flows has been mainly based on the Eulerian formulation, while Lagrangian formulations justify their popularity solving free-surface flows or complicated multi-fluids flows in which the standard Eulerian formulations are inaccurate or, sometimes, impossible to be used.

When attempting to classify transport-equation solvers it is important to take into 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 on any other part of it (enforcing a strong coupling between time and space). *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 (the coupling between time and space is somewhat relaxed).

While the former are more robust the latter are more efficient. This feature is attributed first to the simplicity of its computation with minimum amount of information needed to update the solution and also to be in a better position attending to the present of hardware technology based on the use of parallel computers and general purpose graphic processor units (GPGPU). Focusing on the efficiency in the latter may also include those methods that being implicit lead to a linear system of equations that may be factorized once and integrated in time with the same factorized matrix as a preconditioner.

The Particle Finite Element Method (PFEM)(Oñate et al., 2004)(Idelsohn et al., 2004) added to Explicit Integration following the Velocity and Acceleration Streamlines (X-IVAS)(Idelsohn et al., 2012b) is called PFEM-2 (Particle Finite Element Method - Second Generation). This method shows that Lagrangian formulations are not only valuable to solve heterogeneous fluid flows with free-surfaces. PFEM-2 proves that even for homogeneous fluid flows, without free-surfaces or internal interfaces, it is able to yield accurate solutions while being competitively fast when compared to state-of-the-art Eulerian solvers.

Even though in previous papers (by example (Gimenez and Nigro, 2011) and (Nigro et al., 2011)) the mathematical and numerical basis of the method was presented, a review of the method will be done in the Section 2. For readers interested in the accuracy and the performance for solving scalar-transport and momentum-transport problems, mostly in dominant advection term we suggest to review those references.

However, this work will show that when the problem to solve is diffusion-dominant, the advantages of the method are not as clear as in the advection-dominant case. The first drawback is due to the limitation on the dimensionless Fourier number for using explicit method in diffusion problems. Moreover, in some particular problems, explicit PFEM-2 may fail. This behavior was found in those cases where the temporal change of the transported variables vanishes due to the shape of the own solution. Another drawback that appear is related to the update of the particle inventory needed on one side to control the accuracy of the projection/interpolation operators between particles and mesh and also to limit the maximum number of particles avoiding an excessive computational cost. The former normally produces numerical diffusion or dispersion in the approximated solution.

To relax the first of these restrictions, in the Section 4.1 a new model to calculate the diffusion is developed. It is based on the *theta method* which consists on discretizing the non-stationary variable using a weighted mixture between an explicit prediction and an implicit correction.

This approach extends the stability limit, while reducing the error, and it could also allow the usage of longer time steps. Another feature is that this model does not increase too much the computational cost due to the possibility of factorizing the matrix of the equations system only once and retaining it as a preconditioner for the remaining time steps. For the second problem, an improved approach to create and remove particles is presented in the Section 3.

Along this work some academic tests, where the original PFEM-2 formulation had failed, will be presented. This paper is focused on showing how the new models improve the accuracy of the solutions when compared with the older ones and how the computing times are modified.

2 PFEM-2 METHOD REVIEW

The goal of PFEM-2 is to solve transport-equations. The method is principally motivated by solving viscous incompressible flow equations as fast as possible. Its formulation allows to find numerical results of others scalar or vectorial transport equations, such as heat equation or turbulence modeling, and, also, the coupling between two or more of them.

The Lagrangian expression for a scalar transport-equation is

$$\frac{D\phi}{Dt} = \nabla \cdot (\alpha \, \nabla\phi) + Q$$

where the unknown is the scalar ϕ (if ϕ is the temperature, this equation is called *Heat Equation*).

On the other hand, *Navier-Stokes* equation system describes the behavior of newtonian viscous incompressible flow, its formulation is based on momentum-transport equation which is normally coupled with the equation for the local mass balance. The Lagrangian expression is:

$$\rho\frac{D\mathbf{v}}{Dt} = -\nabla p + \mu(\nabla\mathbf{v}^T + \nabla\mathbf{v}) + \mathbf{f} \tag{1}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{2}$$

where the unknowns are the velocity vector $\mathbf{v}$ and the pressure p.

It is well known that the time step selected in the solution of the transport equations is stable only for time steps smaller than two critical values: the Courant-Friedrichs-Lewy (Co) number and the Fourier (Fo) number (Donea and Huerta, 2003). The first one concerns with the convective terms and the second one with the diffusive ones. In Eulerian formulation both numbers must be less than a constant order one to have stable algorithms. For convection dominant problems like high Reynolds number flows, the condition $Co < 1$ becomes crucial and limits the use of explicit methods or makes the solution scheme far from being efficient.

However, the key of the PFEM-2 algorithm is the ability to reach $Co >> 1$ due to the information is transported on the particles. One of the novelty in PFEM-2 respect to its predecessor PFEM concerns with the integration of particle trajectory and the state variables defining the problem. This integration is performed following the streamlines computed explicitly with the information of the previous time step. This new method was named X-IVAS method and it was presented in (Idelsohn et al., 2012b). X-IVAS represents a more stable explicit time integration not limited by Co. Briefly, the algorithm takes the streamlines as stationary on each time step ($\mathbf{v}^n$), then the particle position follows that field and the particle state variables are updated by the change rate determined by the physics equations.

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_0^{\Delta t} \mathbf{v}^n(\mathbf{x}_p^\tau)\, d\tau \tag{3}$$

$$\phi_p^{n+1} = \phi_p^n + \int_0^{\Delta t} \mathbf{g}^n(\mathbf{x}_p^{n+\tau}) + Q^{n+\tau}\, d\tau \tag{4}$$

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \int_0^{\Delta t} \mathbf{a}^n(\mathbf{x}_p^{n+\tau}) + \mathbf{f}^{n+\tau}\, d\tau \tag{5}$$

where $\mathbf{a}^n = -\nabla p^n + \mu(\nabla \mathbf{v}^{nT} + \nabla \mathbf{v}^n)$ and $\mathbf{g}^n = \nabla \cdot (\alpha\, \nabla \phi^n)$, which are nodal variables.

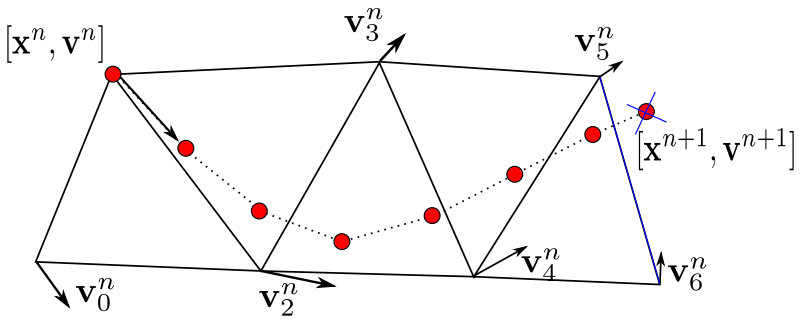


Figure 1: Streamline Integration updating the particle position and state.

To solve the passive scalar transport equation with a known velocity field, the equations (3) and (4) must be used. After streamline integration, nodal values must be updated. If you want to transport N passive scalars with the same velocity field, you only have to solve (4) for each variable with a common convective transport for all the passive scalars. Thereby multi-species problems also may be simulated with PFEM-2 in a more efficient way.

To solve the Navier Stokes equation, systems (3) and (5) are solved coupled with p (Figure 1 shows a graphical interpretation of the streamline integration for incompressible flow problems). A typical Fractional Step Method is used to solve the coupling between the pressure and the velocity (Gimenez and Nigro, 2011)(Idelsohn et al., 2012b). Here, it is also necessary to update nodal states with the states transported by particles.

There are two approaches to update nodal states after streamline integration, each one generates two versions of the method:

- *PFEM-2 Mobile Mesh:* Remesh the geometry with new particles position (particles↔nodes).
- *PFEM-2 Fixed Mesh:* Project states from particles to nodes, preserving the mesh.

This work is devoted to PFEM-2 Fixed Mesh version because avoids the remeshing at each time-step and it has the possibility of factorizing the matrices of the pressure equation and the implicit diffusion step only once. As it was presented in previous works, these features make PFEM-2 Fixed Mesh more efficient than Mobile Mesh.

Having in mind that the incompressibility restriction is non-local, an implicit scheme is needed. This feature normally diminishes the efficiency of explicit incompressible flow solvers causing that the final decision about the selection of the integration scheme be pushed on full implicit solver. In order to keep PFEM-2 explicit and being that Fixed Mesh version may exploit the benefits of having a constant Poisson matrix for the pressure equation, this matrix

is initially factorized (completely or incompletely depending on the problem size) and latterly used as a preconditioner. With this fact in mind and in order to enlarge the time step limited by the dimensionless Fourier number in this work it is proposed to include the possibility to solve part of the diffusion terms in an implicit way using the same idea, an initial factorization chosen as a preconditioner also for the diffusion part. This will be explained in depth in Section 4.1.

Finally, a brief review of the algorithm is presented in 1 and 2.

Algorithm 1 - Time Step PFEM-2 Scalar Transport

1. Calculate scalar change rate on the nodes like a FEM:
   $\int_\Omega \boldsymbol{N}\ \mathbf{g}^n\ d\Omega = -\int_\Omega \nabla \boldsymbol{N}\ \alpha \nabla \phi^n\ d\Omega + \int_\Gamma \boldsymbol{N}\ \nabla \phi^n \cdot \eta\ d\Gamma$
2. Evaluate new particles position and state following the streamlines:
   $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_0^{\Delta t} \mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau$
   $\phi_p^{n+1} = \phi_p^n + \int_0^{\Delta t} \mathbf{g}^n(\mathbf{x}_p^{n+\tau}) + Q^{n+\tau}\ d\tau$
3. Update particles inventory
4. Project state to the mesh:
   $\phi_j^{n+1} = \boldsymbol{\pi}(\phi_p^{n+1})$

Algorithm 2 - Time Step PFEM-2 Incompressible Flow

1. Calculate acceleration on the nodes like a FEM:
   $\int_\Omega \boldsymbol{N}\ \nabla \cdot \boldsymbol{\tau}^n\ d\Omega = -\int_\Omega \nabla \boldsymbol{N} \cdot (\mu \nabla \mathbf{v}^n)\ d\Omega + \int_\Gamma \boldsymbol{N}\ \nabla \mathbf{v}^n \cdot \eta\ d\Gamma$
   $\int_\Omega \boldsymbol{N}\ \nabla p^n\ d\Omega = -\int_\Omega \nabla \boldsymbol{N}\ p^n\ d\Omega + \int_\Gamma \boldsymbol{N}\ p^n \cdot \eta\ d\Gamma$
   $\mathbf{a}^n = \frac{1}{\rho}\nabla \cdot \sigma = \frac{1}{\rho}(-\nabla p^n + \nabla \cdot \boldsymbol{\tau}^n)$
2. Evaluate new particles position and state following the streamlines:
   $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_0^{\Delta t} \mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau$
   $\hat{\mathbf{v}}_p^{n+1} = \mathbf{v}_p^n + \int_0^{\Delta t} \mathbf{a}^n(\mathbf{x}_p^{n+\tau}) + \mathbf{f}^{n+\tau}\ d\tau$
3. Update particles inventory
4. Project state to the mesh:
   $\mathbf{v}_j^{n+1} = \boldsymbol{\pi}(\mathbf{v}_p^{n+1})$
5. Find the pressure value solving the Poisson equation system using FEM:
   $\rho \nabla \cdot \hat{\mathbf{v}}_j^{n+1} = \frac{\Delta t}{2}\Delta[\delta p^{n+1}]$
6. Update the velocity value with the new pressure:
   $\rho \mathbf{v}_j^{n+1} = \rho \hat{\mathbf{v}}_j^{n+1} - \frac{\Delta t}{2}(\nabla p^{n+1} - \nabla p^n)$
   $\rho \mathbf{v}_p^{n+1} = \rho \hat{\mathbf{v}}_p^{n+1} - \frac{\Delta t}{2}\boldsymbol{\pi}^{-1}(\nabla p^{n+1} - \nabla p^n)$

3 PARTICLE INVENTORY MANAGMENT

As mentioned in the Section 2, in the PFEM-2 algorithm, after the streamline integration the state variables are placed on the particles. Both for reasons of incompressibility as for the treatment of the diffusion requires that the information is located on the grid. While for the Mobile Mesh version the mesh is done with the particles themselves, in the Fixed Mesh approach particles and grid are decoupled and a projection $\boldsymbol{\pi}$ from particle states to nodal states should be done.

3.1 Projection Algorithms

Different approaches are available to perform 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. A brief review of the actual techniques for projection in PFEM-2 are presented (the equations presented are for scalar projection. They are also valid for each component of a vector state variable):

- Mean weighted by Shape Function (P-1):

$$\phi_j = \frac{\sum_{p=1}^{P} \mathbf{N}_j(\mathbf{x}_p)\phi_p}{\sum_{p=1}^{P} \mathbf{N}_j(\mathbf{x}_p)} \tag{6}$$

  where P is the number of particles inside a certain region around the node j.

- Mean weighted by Distance (P-2):

$$\phi_j = \frac{\sum_{p=1}^{P} (\mathbf{x}_p - \mathbf{x}_j)^\alpha \phi_p}{\sum_{p=1}^{P} (\mathbf{x}_p - \mathbf{x}_j)^\alpha} \tag{7}$$

  with the same definition of P as just above.

- Weighted Polynomial Least Squares (P-3):

$$\phi_j = h_\theta(\mathbf{x}_j) = \boldsymbol{\theta}^T \mathbf{P}(\mathbf{x}_j) \tag{8}$$

  where:
  $\mathbf{P} = [1\ x\ x^2\ ...]$ on 1d, $\mathbf{P} = [1\ x\ y\ xy\ x^2\ y^2 ...]$ on 2d (truncating at the polynomial order required) and $\boldsymbol{\theta} = (\mathbf{X}^T\ \mathbf{W}\ \mathbf{X})^{-1}(\mathbf{X}^T\ \mathbf{W}\ \mathbf{y})$. It must be noted that to invert the matrix in the calculation of $\boldsymbol{\theta}$, it is required $P >= n$, where n is the number of terms used on P.

3.2 Particle Seeding

For accuracy reasons each one of the presented projection methods requires a certain number of particles in certain region near to each node. Some considerations must be taken into account: the *region around the node* must be defined precisely, and is not assured that there were particles inside each region (specially when high Co numbers are used). Then, new particles must be created at these empty regions. In this section several algorithms attending this issue are presented.

The first approach (S-1), used originally in PFEM-2, consists on setting the states to a new particle interpolating from the nodal states at the previous time step n:

$$\phi_p^{n+1}(\mathbf{x}_p^{n+1}) = \sum_j \lambda_j(\mathbf{x}_p^{n+1})\phi_j^n \tag{9}$$

being λ_j the area coordinates of the particle in the element. Other algorithm (S-2) searches the state following the streamline but in backward direction, thinking in *finding the particle location that, if it had existed at the beginning of the time step, at the end of it would have arrived at the seed position*:

$$\phi_p^{n+1}(\mathbf{x}_p^{n+1}) = \phi_p^n(\mathbf{x}_p^n) + \int_0^{\Delta t} \mathbf{g}^n(\mathbf{x}^{n+\alpha})\, d\alpha \tag{10}$$

The utility of the backward integration to search the state of the new particle is shown in the next example: the pure-convection step problem, which is defined in Figure 2.

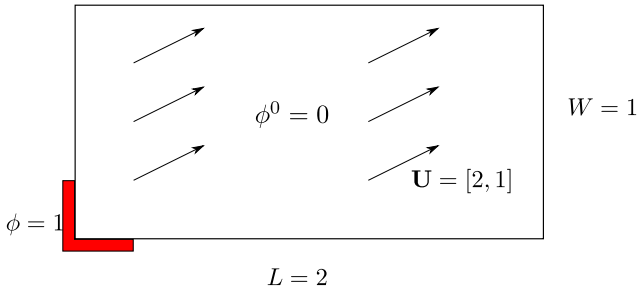
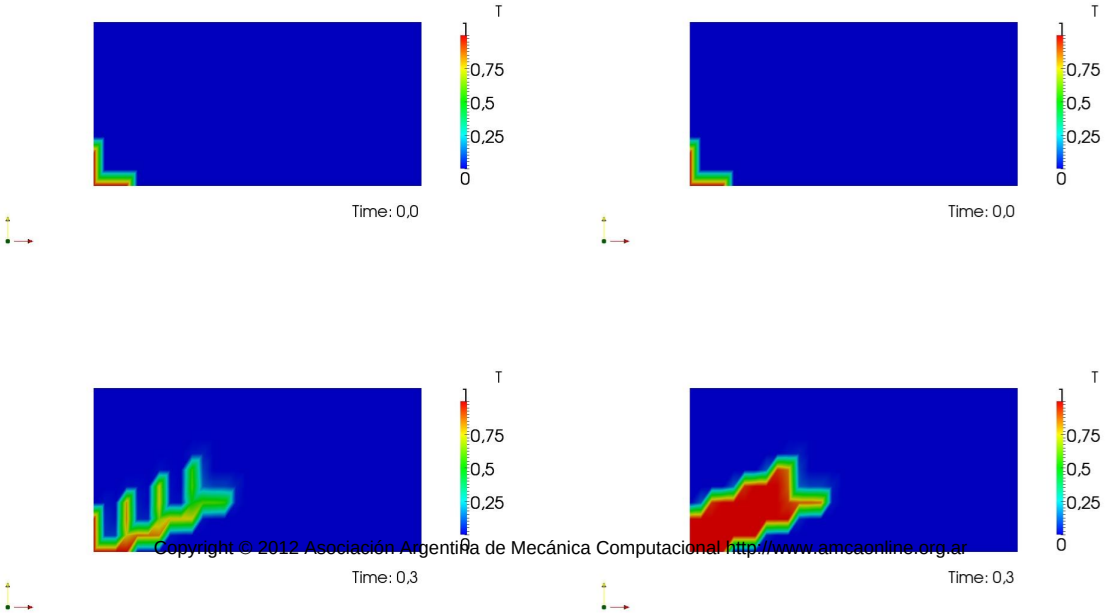


Figure 2: Pure-convection Step Problem

A boundary condition with a sharp discontinuity entered the domain transported with a velocity vector field not aligned with the mesh. Figures 3 show the results, projected on the mesh, achieved when the particles are created using the criterion defined as (S-1) and also when their states are found using backward integration criterion (S-2).



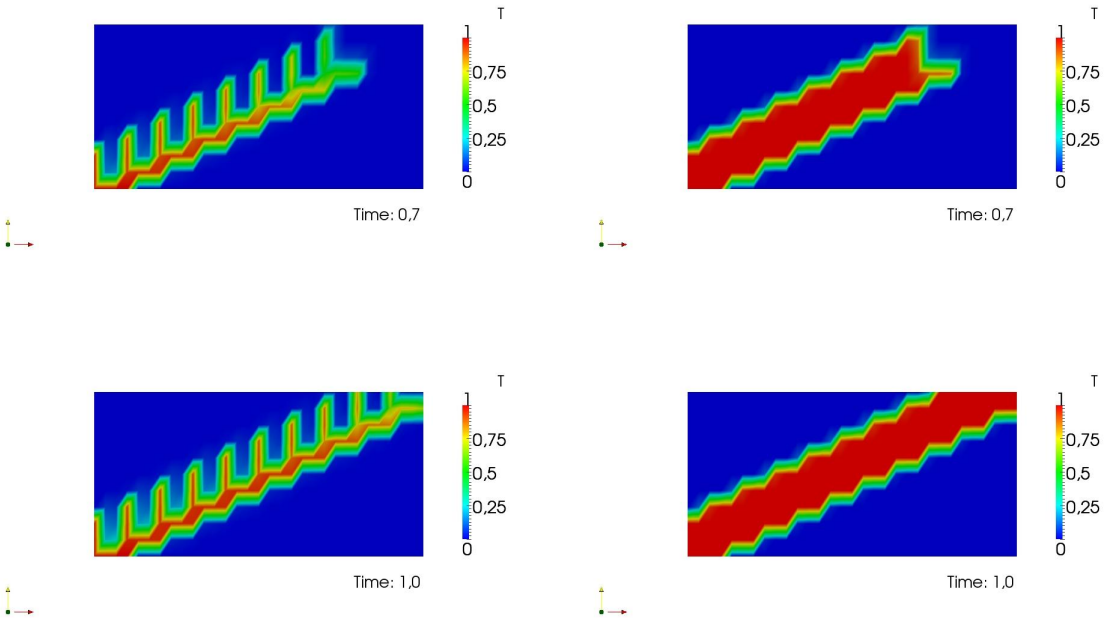


Figure 3: Pure convection transport. Results for different strategies to new particles states. Left: S-1. Right: S-2.

The results show that S-1 criterion fails for this example putting S-2 criterion as a much better selection for seeding particles when it is necessary.

3.2.1 Particle Removing

On other hand, using a lot of particles increases computing times. During the simulation the seeding is frequent and we need to control the amount of particles inside the domain for computational cost reasons. So, the removal action should be defined following some criteria.

Although it is known that particle which leaves the geometry must be deleted, inside the geometry is not clear when particles should be removed and how to do that. Removing particles decreases computing times of the algorithm but also decreases the quality of the solution because it introduces numerical diffusion in an indirect way.

In the first approach (R-1), the particles are not removed unless that two or more of them are in almost the same position. This approach obtained accurate results solving the pure-advective case of the rotating Gaussian signal (Nigro et al., 2011).

A second approach (R-2), consists on requiring minimum number and a maximum number of particles at each sub-element that must be conserved at each time step. The sub-element i is the third parts of the triangle (fourth parts of the tetrahedral in three dimensions) where the area coordinate corresponding to the vertex i is larger than the rest. The idea is to think that if each node has enough number of particles around, the projected state from particles to the node will be accurate. However, as will be demonstrated in the next example, the continuous intrusion to the system could decrease the quality of the solution, specially in scalar problems. However this criterion shows good results in solving incompressible flows.

The example of the *rotating Gaussian signal without diffusion* shows that the numerical diffusion is important when a frequent creation and removing of particles is performed following this criterion. The polar mesh (4390 elements) is the same in all cases, the case consists of a Gaussian transported by a rotating flow without diffusion term. Figure 4 compares the value of the maximum through two laps. The best option for this problem is R-1, while R-2 requires a large range ($[min\_subele; max\_subele]$) of particles not to spread: comparing $[1, 20]$ with $[1, 5]$, in the second option the intrusion in the system is greater and the solution decreases its quality.

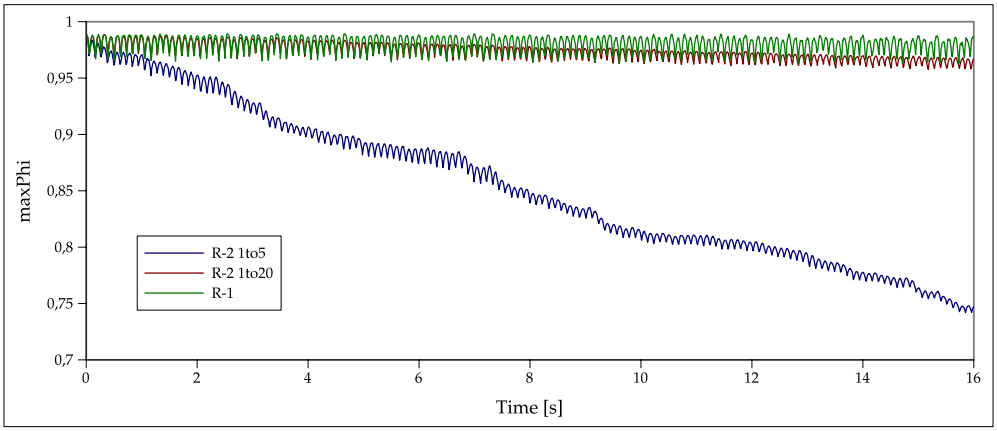


Figure 4: Rotating Gaussian. Evolution of ϕ_{max} for different update particle techniques.

Regarding to computing times (each case using the same small Δt that ensures $Co < 1$) also R-1 is the best, because requires $40[s]$ (mean 40000 particles), whereas $[1, 5]$ needs $50[s]$ (mean 46000 particles) and $[1, 20]$ needs $100[s]$ (mean 120000 particles).

However, R-1 does not work fine for the step's problem (defined in previous subsection), unless it creates new particles using backward integration (criterion S-2). Also, due to the type of projection of the algorithms developed, which searches particles in the sub-elements to send data to nodes, creating new particles in empty elements does not ensure that there will be particles in the region around the node (their sub-elements), so another type of selection of the position of the new particles must be developed.

3.3 Converging into a new algorithm to Update Particle Inventory

Taking account the previous discussion, an algorithm to update particle inventory has been implemented, that includes the benefits of the methods discussed previously and follows the principle of not modifying the system unless it be really necessary.

It was mentioned that the condition *empty-element*, except for some particular problems, shows to work not so fine. The main reason is the lack of particles close enough to nodes. So, the new algorithm must not search *empty-elements*, instead of that, it searches *empty-regions*.

The region j is defined as the geometric space where the node j takes particle information to update its state using some projection operator; therefore, if this region is empty, the node does not have enough information to be updated. Once detected an *empty-region*, only one particle is created in exactly the same position of its node and with a state found using backward integration.

Although to create new particles on the nodes is the least intrusive way to maintain good

solution on the nodes, if the problem is not of *confined flow*, the number of particles will decrease while the simulation runs. Typically in the inlet flow boundary the boundary condition is imposed. Then, creating particles in the *Inlet Elements* and doing backward integration to search the states no error is committed, and more than one particles can be created. The position does not have to be on the nodes and its state will not have error. This approach allows to keep approximately the same numbers of particles during the simulation, which preserves the accuracy of the method.

Particle removing is carried out when two or more particles are in a circle (2d) or sphere (3d) with a radius proportional to the size of the element ($r = \delta h$). This approach allows to use different δs over the geometry, being a new tool to control the number of particles. Graphic representation is presented in the Figure 5c.

Figure 5a shows the neighbor elements and the region belonging to the node j. It must be there at least one particle in the gray zone to have a good projection, else a new particle must be created in the same position of the node and searching its state with backward integration. Figure 5b shows which elements are considered as *inlet elements* and, if they are empty, they must create internal particles.

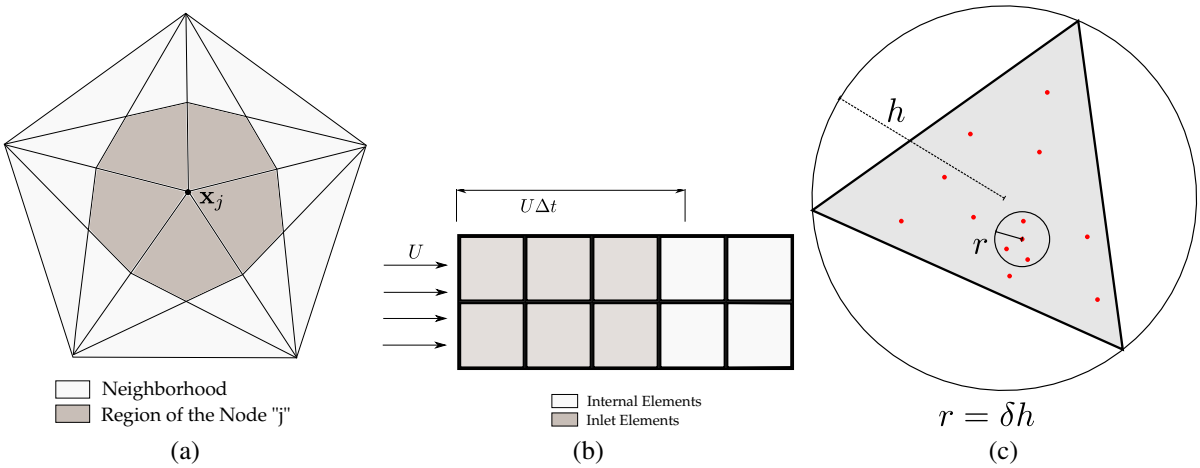


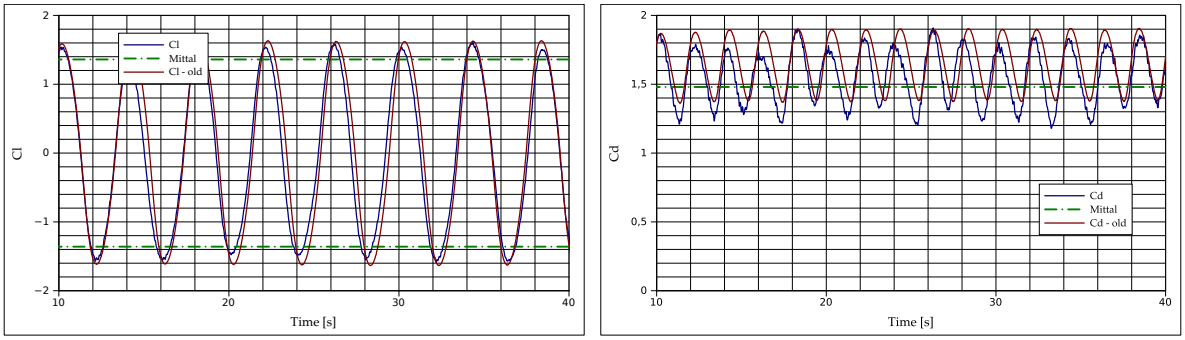
Figure 5: Graphical scheme to understand the new algorithm proposed.

Finally, this algorithm allows to solve all tests presented in this report while other approaches have shown to fail: the *Gaussian rigid rotation* and the *step-2d*.

The last example consists on testing this algorithm in the Navier-Stokes solver (*PFEM-2 Fixed Mesh*). The case chosen is the *Flow Around a Cylinder*, because it presents different zones of refinement and patches of inlet and outlet flow. The results are presented in the Figures 6a and 6b. Similar accuracy in the amplitude and frequency can be observed, but R-2 obtains better definition of the forces signal, specially with Cd.

4 DIFFUSIVE-DOMINANT PROBLEMS

When the problem is diffusive-dominant, the advantages of the method PFEM-2 are not as clear as in the advective-dominant case. The explicit calculation of the diffusion traditionally used by PFEM-2 is limited by the dimensionaless Fourier number and, in some particular cases, the temporal change of the transported variables vanishes due to the shape of its own solution. To relax these restrictions, in the Section 4.1 a new model to calculate the diffusion is presented. Several tests are presented to confirm improvements in the solution.



(a) (b)

Figure 6: Lift Coefficient and Drag Coefficient for Flow Around a Cylinder solved using the new updating algorithm and comparing it with R-2 (called *old* in the graphic).

4.1 Diffusive Implicit Correction

Simulations solving the diffusive term in an explicit way are restricted by $Fo < 0.5$. This is a strong limitation for the time-step, specially on very refined mesh and in diffusive dominant problems (where $Fo > Co$). Due to explicit PFEM-2 suffers this stability constraint the possibility of enlarging the time step may be lost when the flow locally turns to be diffusive. As we have mentioned normally in the vicinity of bodies some refinement is done to capture boundary layers and flow separations and locally the Fourier number increases. Also, in some particular cases, the temporal change of the transported variables vanishes due to the shape of the own solution: when the time-step is chosen such that the integral of the curvature of the function ϕ_j^n vanishes, the method will not apply diffusion on ϕ, so that the solution will be wrong. This case may be present for traveling waves with diffusion.

A new approach to solve the diffusive term is based on the *theta method* which consists on discretizing the non-stationary variable using a weighted mixture between an explicit prediction and an implicit correction.

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \theta \mathbf{g}^{n+1} + (1-\theta)\mathbf{g}^n \tag{11}$$

Doing a first step in explicit way

$$\frac{\hat{\phi}^{n+1} - \phi^n}{\Delta t} = (1-\theta)\mathbf{g}^n \tag{12}$$

and doing the correction in an implicit way, this is subtracting (12) from (11), it follows

$$\frac{\phi^{n+1} - \hat{\phi}^{n+1}}{\Delta t} = \theta \mathbf{g}^{n+1} \tag{13}$$

The new PFEM-2 algorithm is presented in 3. With $\theta = 0$ the algorithm turns to be fully explicit (Algorithm 1), while with $\theta = 1$ the algorithm solves only the diffusion in an implicit way.

Algorithm 3 - Time Step PFEM-2 Scalar Transport Explicit Diffusion - Implicit Correction

1. Calculate scalar change rate on the nodes like a FEM:
   $\int_\Omega \boldsymbol{N}\ \mathbf{g}^n\ d\Omega = -\int_\Omega \nabla \boldsymbol{N}\ \alpha \nabla \phi^n\ d\Omega + \int_\Gamma \boldsymbol{N}\ \nabla \phi^n \cdot \eta\ d\Gamma$
2. Evaluate new particles position and state following the streamlines:
   $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_0^{\Delta t} \mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau$
   $\hat{\phi}_p^{n+1} = \phi_p^n + \int_0^{\Delta t} \mathbf{g}^n(\mathbf{x}_p^{n+\tau}) + Q^{n+\tau}\ d\tau$
3. Update particles inventory
4. Project state to the mesh:
   $\hat{\phi}_j^{n+1} = \boldsymbol{\pi}(\hat{\phi}_p^{n+1})$
5. Implicit correction:
   $\phi_j^{n+1} = \hat{\phi}_j^{n+1} + \Delta t\ \theta \mathbf{g}_j^{n+1}$.
6. Interpolate state to particles:
   $\phi_p^{n+1} = \hat{\phi}_p^{n+1} + \boldsymbol{\pi}^{-1}(\delta\phi_j^{n+1})$.

An standard FEM formulation is used to compute the implicit correction. This problem may be solved either with the approaches *absolute* (14) or *incremental* (15).

$$[\mathbf{M} + \theta\ \Delta t\ \mathbf{K}]\ \phi^{n+1} = \mathbf{M}\ \hat{\phi}^{n+1} + \Delta t\ \mathbf{F}^{n+\frac{1}{2}} \tag{14}$$

$$[\mathbf{M} + \theta\ \Delta t\ \mathbf{K}]\ \delta\phi^{n+1} = -\theta\ \Delta t\ \mathbf{K}\ \hat{\phi}^{n+1} + \Delta t\ \mathbf{F}^{n+\frac{1}{2}} \tag{15}$$

where $\mathbf{M}$ is a mass matrix, $\mathbf{K}$ is the stiffness matrix and $\mathbf{F}$ is the load vector of a standard FEM discretization. It must be noted that the matrix $[\mathbf{M} + \theta\ \Delta t\ \mathbf{K}]$ for $k \neq k(t)$ and $\Delta t = cte$ does not depend on the time, then it can be factorized at the beginning of the computation and used as a preconditioner afterward with a significant cpu-time reduction.

4.2 Analytic Diffusion 1D: Sinusoidal Signal

A diffusive-dominant problem with analytic solution is presented. It is solved using explicit, implicit and semi-implicit schemes for diffusion and using different Fo values.

The problem is:

$$\frac{\partial \phi}{\partial t} = \alpha \frac{\partial^2 \phi}{\partial^2 x} \quad \forall x(0,1) \tag{16}$$

$$\phi(x=0,t) = \phi(x=L,t) = 0 \quad t > 0 \tag{17}$$

$$\phi(x,t=0) = \sin(kx) \quad t = 0 \tag{18}$$

with its analytic solution as:

$$\phi(x,t) = \sin(kx) e^{-(2\pi k x)^2 t} \tag{19}$$

where $\alpha = 1$ is chosen, the wave number of the problem is $k = \frac{2\pi}{\lambda} = 4$ and the mesh size is $\Delta x = 0.02$.

Figure 7 shows that using $Fo = 5$ more accurate results are obtained choosing a semi-implicit scheme. While explicit simulations are not accurate with $Fo = 5$, the solution with $Fo = 0.5$ is useful. These results show that temporal integration error in PFEM-2 behaves

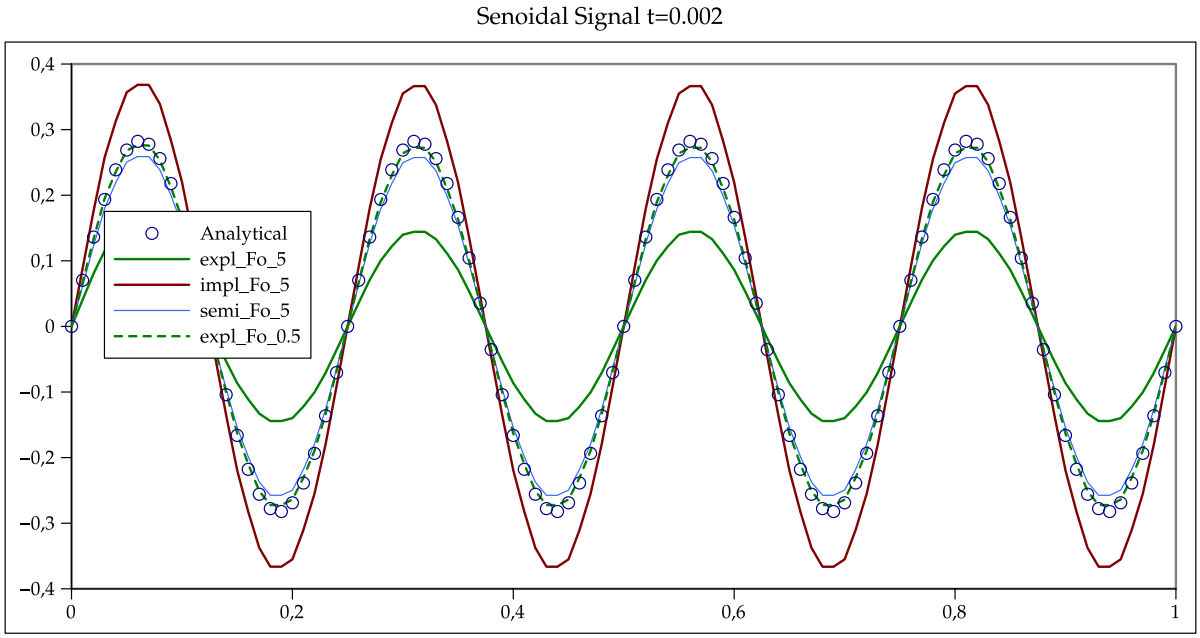


Figure 7: Comparison between Explicit ($\theta = 0$), Implicit ($\theta = 1$) and Semi-Implicit ($\theta = 0.5$) schemes for diffusion in PFEM-2 with the analytic solution at $t = 0.002$.

as usual, i.e. semi-implicit or Crank Nicholson schemes are more accurate than first order explicit or fully implicit schemes. However for semi-implicit solvers we need to give up the idea of an explicit solver with severe consequences on the efficiency. Do not forget that one of the main goals in PFEM-2 development is having a robust (stable) solver that allows to switch between accuracy and efficiency with greater freedom. But, due to the matrix for the implicit part of the computation of the diffusion can be factorized once at the beginning, it is possible to run simulations using big time-steps without loosing accuracy and efficiency. This idea of combining explicit schemes with efficient implicit schemes gives Fixed Mesh PFEM-2 its stronghold. Efficient implicit schemes means solving linear systems in an iterative way with good preconditioners.

4.3 A Pathological case: Sinusoidal Signal Travelling

In this section, a pathological case is presented. The explicit calculation of the diffusion updates the state variable with the integral of the second derivative of the variable itself, i.e. the integral of the curvature. When certain conditions are accomplished, that integral vanishes and the explicit diffusion is null generating wrong new states. However, an implicit calculation of the diffusion solves that problem.

The problem consists on a sinusoidal wave transported by a field $\mathbf{v}$ with a not negligible diffusive term. The idea is to force numerical and physical parameters searching that the integral of the curvature of the function vanishes at each time-step.

If the length traveled by a particle is multiple of the length wave of the signal ($U\Delta t = m\lambda$), then $x_p^{n+1} = x_p^n + m\lambda$, hence, the rate of change of the variable (its curvature κ) will be null because $\kappa = \dfrac{d^2\phi}{dx^2} = \dfrac{d^2}{dx^2}[\sin(\dfrac{2\pi}{\lambda}x)] = C\ \sin(\dfrac{2\pi}{\lambda}x) = \mathbf{g}$ y $\int_{x^n}^{x^{n+1}} \mathbf{g}\ dx = 0$. This pathological situation has a very low probability and only is present in Lagrangian formulations where advection and diffusion are weakly coupled.

The problem to solve consists on:

$$\frac{\partial \phi}{\partial t} + U\frac{\partial \phi}{\partial x} = \alpha \frac{\partial^2 \phi}{\partial^2 x} \quad \forall x \in (0, \infty) \tag{20}$$

$$\phi(x = 0, t) = \sin(\omega t) \quad t > 0 \tag{21}$$

$$\phi(x, t = 0) = \sin(\frac{2\pi}{\lambda} x) \quad t = 0 \tag{22}$$

where the advection and the diffusion can be analytically solved in an uncoupled way, allowing to determinate the decay of the signal.

$$\phi(x, t) = \sin(\frac{2\pi}{\lambda}\,[x - (x_0 + Ut)])\; e^{-(\frac{2\pi}{\lambda})^2 t} \tag{23}$$

Using the parameters:

- $U = 5000$
- $\alpha = 1$
- $L_x = 10$
- $\lambda = 0.25$
- $\Delta x = 0.025$
- $\Delta t = 0.0001$ $(Fo = 0.16)$

In Figure 8 results obtained with different values of θ are presented comparing with analytic decay. The most accurate simulation is using $\theta = 1$, using $\theta = 0$ decay is not observed and with other values for θ intermediate solutions are obtained. Finally, a corrective step of a erroneous explicit prediction does not ensure accurate results due to the bad performance of explicit schemes for this very special case.

It must be emphasized that the presented case rarely appears in non-academic problems, but it allows to demonstrate another reason to choose an implicit calculation of the diffusion instead of an explicit.

4.4 Implicit Calculation of the Viscous Diffusion

The theta method can also be adopted to calculate the viscous effects on incompressible flow problems. Again, this strategy allows to extend the maximum time-step without the limitation of the Fourier number. The expressions are similar to the scalar case presented in equations 11, 12 and 13, but replacing ϕ with $\mathbf{v}$ and $\mathbf{g}$ with $\nabla \cdot \boldsymbol{\tau}$ where $\boldsymbol{\tau} = \mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T)$.

Finally, the algorithm for the implicit correction of the viscous stress tensor is presented in Algorithm 4.

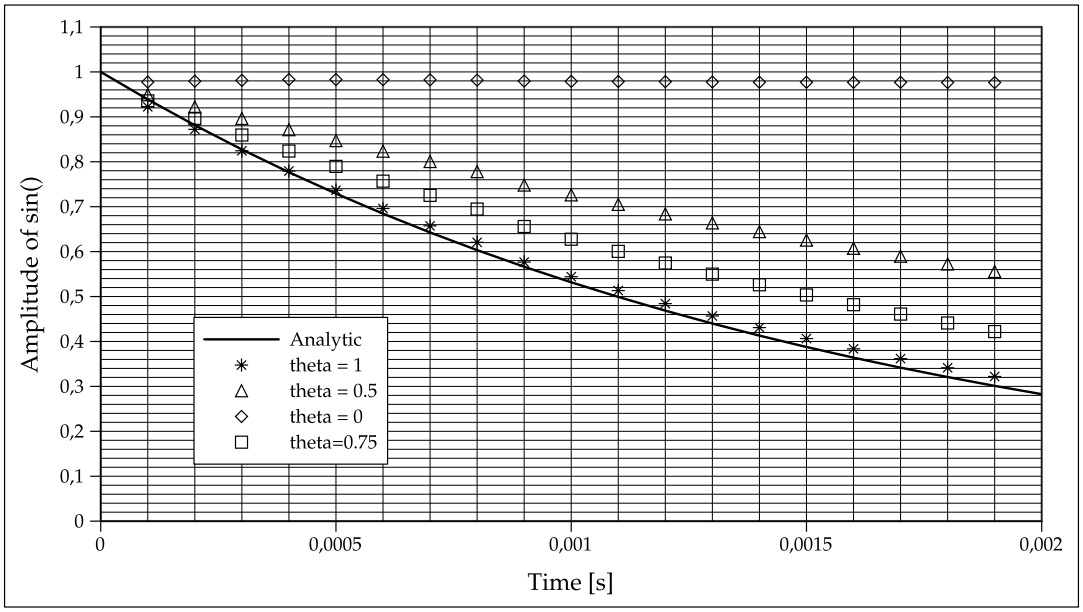


Figure 8: Temporal evolution of sinusoidal amplitude.

Algorithm 4 - Time Step PFEM-2 Incompressible Flow with Implicit Correction of the viscous diffusion.

1. Calculate acceleration on the nodes like a FEM:
   $\int_\Omega \boldsymbol{N}\ \nabla\cdot\boldsymbol{\tau}^n\ d\Omega = -\int_\Omega \nabla\boldsymbol{N}\cdot(\mu\nabla\mathbf{v}^n)\ d\Omega + \int_\Gamma \boldsymbol{N}\ \nabla\mathbf{v}^n\cdot\eta\ d\Gamma$
   $\int_\Omega \boldsymbol{N}\ \nabla p^n\ d\Omega = -\int_\Omega \nabla\boldsymbol{N}\ p^n\ d\Omega + \int_\Gamma \boldsymbol{N}\ p^n\cdot\eta\ d\Gamma$
   $\mathbf{a}^n = \frac{1}{\rho}\nabla\cdot\sigma = \frac{1}{\rho}(-\nabla p^n + (1-\theta)\nabla\cdot\boldsymbol{\tau}^n)$
2. Evaluate new particles position and state following the streamlines:
   $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_0^{\Delta t}\mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau$
   $\hat{\mathbf{v}}_p^{n+1} = \mathbf{v}_p^n + \int_0^{\Delta t}\mathbf{a}^n(\mathbf{x}_p^{n+\tau}) + \mathbf{f}^{n+\tau}\ d\tau$
3. Update particles inventory
4. Project state to the mesh:
   $\hat{\mathbf{v}}_j^{n+1} = \boldsymbol{\pi}(\hat{\mathbf{v}}_p^{n+1})$
5. Implicit correction:
   $\rho\hat{\mathbf{v}}_j^{n+1} = \rho\hat{\hat{\mathbf{v}}}_j^{n+1} + \Delta t\ \theta\nabla\cdot\boldsymbol{\tau}_j^{n+1}$.
6. Find the pressure value solving the Poisson equation system using FEM:
   $\rho\nabla\cdot\hat{\mathbf{v}}_j^{n+1} = \frac{\Delta t}{2}\Delta[\delta p^{n+1}]$
7. Update the velocity value with the new pressure:
   $\rho\mathbf{v}_j^{n+1} = \rho\hat{\mathbf{v}}_j^{n+1} - \frac{\Delta t}{2}(\nabla p^{n+1} - \nabla p^n)$
   $\rho\mathbf{v}_p^{n+1} = \rho\hat{\mathbf{v}}_p^{n+1} - \frac{\Delta t}{2}\boldsymbol{\pi}^{-1}(\nabla p^{n+1} - \nabla p^n)$

The equation system for the implicit correction of the viscous diffusion can be solved using the same strategy as presented in 14 or 15. Also, for $\mu \neq \mu(t)$ and $\Delta t = cte$ the matrix does not depend on the time, then it can be factorized only once.

4.4.1 Flow Around a Cylinder in three dimensions

Even though the flow around a cylinder was previously solved in two dimensions with good agreement against another 2d results (Idelsohn et al., 2012a), however there were some differences between those numerical results and the experimental ones. Those differences were due to the simplification of the geometry which did not capture some 3d effects (modes on the solution) and generated excessive aerodynamic coefficients. Then, to avoid these problems and to check the behavior of the implicit correction of the viscous diffusion, the flow around a cylinder in three dimensions was solved with the approach *Fixed Mesh* and was compared with Mittal simulations (Mittal, 2001) and also with numerical results obtained using the widely recommended Open Source software OpenFOAM®.

Experimental data is extracted from Roshko (Roshko, 1960), where the wind tunnel had $8D$ of height, $11D$ along downstream and the cylinder is spanned $8D$.

The geometry is $21D$ of longitude, $11D$ of height and $5D$ of thickness. The mesh used by PFEM-2 and OpenFOAM® has around a half million tetrahedral elements and approximately a hundred thousand nodes. The number of particles for PFEM-2 is around four millions. On other hand, Mittal 3d simulations were performed with a mesh of $0.9e6$ hexahedral elements and around $0.9e6$ nodes. There was 150 non-uniformly spaced elements in the span-wise direction (allowing to capture B modes). The Reynolds number simulated was $Re = 1000$.

	Strouhal	$\overline{C_d}$	C_d Amplitude	C_l amplitude
Experimental	0.21	1.02	-	-
Mittal (2d)	0.25	1.48	0.21	1.36
PFEM-2 (2d)	0.2475	1.639	0.245	1.63
Mittal (3d)	0.2	1.18	-	0.1 to 0.3
PFEM-2 (3d)	0.185	1.16	-	0.2 to 0.3
OpenFOAM® (3d)	0.195	1.22	-	0.5

Table 1: Comparison table Cylinder Re=1000 between Experimental results, reference 2d and 3d numerical results from Mittal, icoFoam solver from OpenFOAM®, and PFEM-2 2d and 3d. Due to more than one frequency appears on Cd curves in 3d simulations, some columns of amplitude values are not completed.

A capture of the velocity magnitude isourfaces ($magU < 0.8U$) with the background mesh is presented in Figure 9.

The Strouhal number based on the dominant frequency in the variation of the lift coefficient is $St = 0.2$. For a quasi-infinite cylinder, the value of St is, approximately, 0.21. It has been reported that a cylinder with $L/D = 50$ is needed to achieve this value. For smaller aspect ratio cylinders the ending effects cause a reduction in the St. Compared to the 3D calculations, the 2D results overestimate the Strouhal number and the mean drag coefficient. As expected, the results from 3D computations are closer to the experimental observations.

Solver	Δt_{max}	Co_{mean}	Co_{max}	CPU Time
PFEM-2 Implicit Diffusion	$0.1[s]$	0.9	≈ 8	$78.7[s]$
PFEM-2 Explicit Diffusion	$0.06[s]$	0.55	≈ 4.5	$104.9[s]$
OpenFOAM® (3d)	$0.03[s]$	0.3	≈ 3	$603.4[s]$

Table 2: Comparison table for computing times with different solvers.

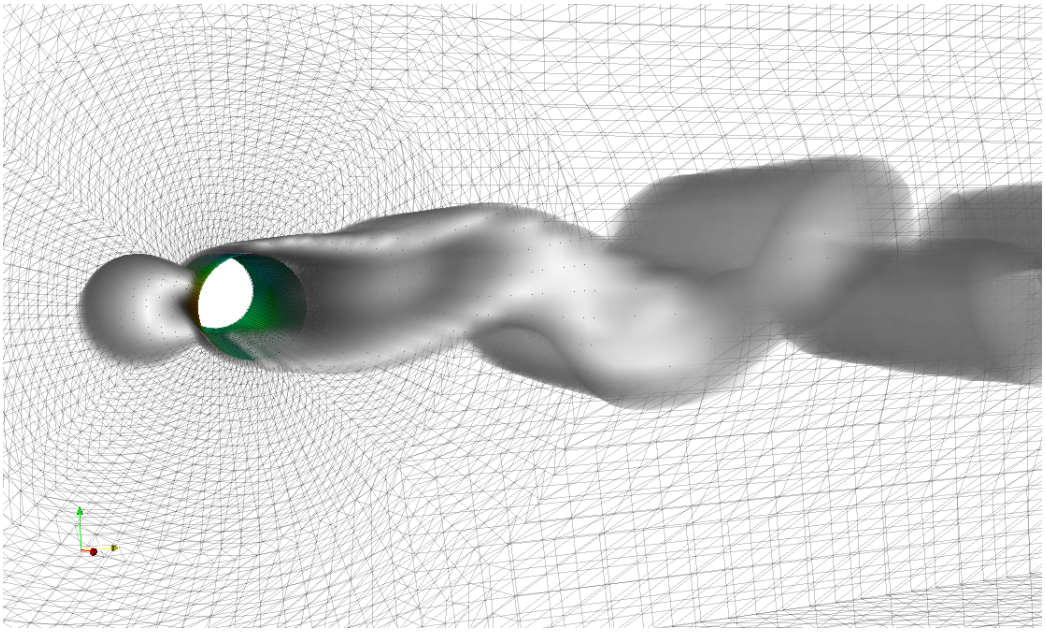


Figure 9: Velocity magnitude isosurfaces ($magU < 0.8U$) with the background mesh.

Regarding to the computing times a comparison is presented in the Table 2. The value of Δt_{max} on each solver is selected trying not to loose accuracy on the solution. The simulation time was $1[s]$ on each calculation and profiling were performed over a processor Intel i7-2600k with 16Gb RAM memory solving on parallel way using 4 cores.

In this particular case, the time-step used by the Implicit is not too greater than the used by the Explicit. The reason is that if a bigger Δt had been selected in implicit solver, some solution details would have been lost due to overcome the minimum sampling of the frequency of vortex generation. Increasing the mesh refinement, the Fourier number would be more strict and the time step for the explicit solver would be smaller.

On other hand, Figure 10 shows that implicit steps (Poisson step and implicit correction of the viscosity) does not demand too much computing times comparing with for example, streamline integration. However, streamline integration can achieve better speed-up than the other stages of the algorithm (Gimenez and Nigro, 2011).

5 CONCLUSIONS

An exhaustive analysis of the methods to project states from particles to nodes and to update the particle inventory was presented. A new algorithm for updating particles based on regions is proposed, reducing the numerical diffusion and being more efficient than the old method based on sub-elements used for incompressible flows. However, although in scalar transport problems the new algorithm obtains better results, the old updating algorithm has similar results for incompressible flow problems where forces are calculated.

Regarding to the stability of the method, in previous papers a methodology to keep the time integration as explicit independently of the Courant number has been presented, and in this paper a strategy to overcome the Fourier number restriction is reported. The implicit correction of the diffusion extends the stability limit, while reducing the error, and it could also allows the usage of longer time steps. Another feature is that this model does not increase too much the computing cost due to the possibility of factorizing the matrix of the equation system only once.

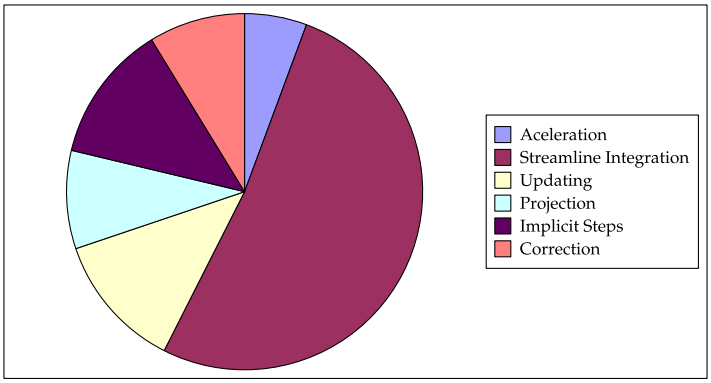


Figure 10: Diagram with the distribution of the computing times of each stage of the algorithm PFEM-2 Implicit Diffusion for the flow Around a Cylinder 3d running over four cores.

The presented case of the flow around a cylinder in three dimensions confirms that the method is able to yield accurate engineering solutions while being competitively fast when compared to state-of-the-art Eulerian solvers.

Being the implicit way the best option (in accuracy and efficiency) to calculate diffusive terms, an algorithm to calculate time-dependent diffusivity is being developed. Using time-dependent diffusivity will allow to solve problems as turbulence modeling where the diffusivity depends on the solution itself ($\alpha = \alpha(\phi)$). The idea is to generate a clever preconditioner for PCG based on a Cholesky decomposition where only a diagonal matrix must be modified at each time-step. This approach will allow, again, to factorize the matrix of the equation system only once at the beginning with minor corrections on the whole preconditioner at each time step attending to the changes in the diffusivity (viscosity) parameters.

It should be pointed out again that the big advantage of the formulation proposed in PFEM-2 is the possibility to use large time steps (as in the implicit solutions) while preserving some sort of explicit strategy. This simplifies the implementation of the formulation in parallel computers, improving considerably the CPU time in complex CFD problems.

REFERENCES

Donea J. and Huerta A. *Finite element method for flow problems*. Springer-Verlag., 1st edition, 2003.

Gimenez J.M. and Nigro N. Parallel implementation of the particle finite element method. *ENIEF*, XXX:3021–3032, 2011.

Idelsohn S., Oñate E., and Del Pin F. The particle finite element method a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *International Journal of Numerical Methods*, 61:964–989, 2004.

Idelsohn S., Nigro N., Gimenez J., Rossi R., and Marti. J. A fast and accurate method to solve the incompressible navier-stokes equations. *Engineering Computations*, XXX:XXX–XXX, 2012a.

Idelsohn S., Nigro N., Limache A., Oñate E. Large time-step explicit integration method for solving problems with dominant convection. *Comp. Meth. in Applied Mechanics and Engineering*, 217-220:168–185, 2012b.

Mittal S. Computation of three-dimensional flows past circular cylinder of low aspect ratio. *Physics of Fluids*, 13 (1), 2001.

Nigro N., Gimenez J., Limache A., Idelsohn S., Oñate E., Calvo N., Novara P., and Morin P. A new approach to solve incompressible navier-stokes equation using a particle method. *ENIEF*, XXX, 2011.

Oñate E., Idelsohn S., Del Pin F., and Aubry R. The particle finite element method, an overview. *International Journal of Computational Methods*, 1:267–307, 2004.

Roshko A. Experiments on the flow past a circular cylinder at very high reynolds numbers. Aeronautical Laboratory, California Institute of Technology, Calfornia, 1960.