RECENT ADVANCES IN THE PARTICLE FINITE ELEMENT METHOD. TOWARDS MORE COMPLEX FLUID FLOW APPLICATIONS.

NORBERTO M. NIGRO[A], PABLO NOVARA[B], JUAN M. GIMENEZ[A], MARTA B. BERGALLO[C], NESTOR A. CALVO[B], PEDRO MORIN[C] AND SERGIO R. IDELSOHN[D]

[A] Centro de Investigación en Métodos Computacionales (CIMEC)
Consejo Nacional de Investigaciones Científicas y Técnicas y la Universidad Nacional del Litoral
Parque Tecnológico Litoral Centro, Colectora Ruta Nacional 168, 3000 Santa Fe, Argentina
e-mail: nnigro@intec.unl.edu.ar, www.cimec.gov.ar

[B] Grupo de Geometría Computacional
Universidad Nacional del Litoral
Ciudad Universitaria, Colectora Ruta Nacional 168, 3000 Santa Fe, Argentina
e-mail: nestor.calvo@gmail.com

[C] Departamento de Matemáticas e Instituto de Matemáticas del Litoral
Consejo Nacional de Investigaciones Científicas y Técnicas y la Universidad Nacional del Litoral
Parque Tecnológico Litoral Centro, Colectora Ruta Nacional 168, 3000 Santa Fe, Argentina
e-mail: pmorin@intec.unl.edu.ar

[D] ICREA Research Professor at the International Center for Numerical Methods in Engineering (CIMNE)
Barcelona, Spain.

e-mail: sergio@cimne.upc.edu

Key words: Particle Finite Element method, Lagrangian, Preconditioner, Real Time, High Performance Computing.

Summary. *One of the main drawbacks of the explicit integration using Eulerian formulations is the restricted stability of the solution with the time steps and with the spatial discretization. For the case of the Navier-Stokes equations, it is well known that the time step to be used in the solution is stable only for time step smaller than two critical values: the Courant-Friedrichs-Lewy (CFL) number and the Fourier number. The first one is concerning with the convective terms and the second one with the diffusive ones. Both numbers must be less than one to have stable algorithms. For convection dominant problems like high Reynolds number flows, the condition CFL<1 becomes crucial and limit the use of explicit method or outdistance it to be efficient. On the other hand, implicit solutions using Eulerian formulations is restricted in the time step size due to the lack of convergence of the convective non-linear terms. Both time integrations, explicit or implicit are, in most cases, limited to CFL no much larger than one. The possibility to perform parallel processing and the recent upcoming of new processors like GPU and GPGPU increase the possibilities of the explicit* *integration in time due to the facility to parallelize explicit methods having results with speed-up closed to one. Although the incompressible condition cannot be solved explicitly, the solution of the momentum conservation equations with an explicit integration of the convective terms together with a parallel processing reduces considerably the computing time to solve the whole problem provided that a large time-step may be preserved independently to the discretization in space. Only to remember the new Particle Finite Element Method, called PFEM 2nd generation (PFEM-2) uses a Lagrangian formulation with an explicit time integrator without the CFL<1 restriction for the convective terms. This allows large time-steps, independent of the spatial discretization, having equal or better precision that an implicit integration. Moreover, PFEM-2 has two versions, one for moving mesh with permanent remeshing and one for fixed mesh [1]. In this lecture we will present some recent advances in the Particle Finite Element Method (PFEM) to solve the incompressible Navier-Stokes equations coupled with another fields like in multiphysics exploiting some nice features found in the fixed version. On the other hand we will also present the moving mesh version applied to multifluids using a parallel remeshing that makes this efficient in terms of cpu time. This updated proposal will be tested numerically and compared in terms of accuracy as in computing cpu time with other more standard Eulerian formulations.*

1 INTRODUCTION

During the last years a huge amount of work have been done in order to reduce the computational cost of engineering simulations. Particle based meshless methods like Smooth Particle Hydrodynamics (SPH), explicit cellular automata like Lattice Boltzmann (in particular BGK), enhanced implementations of standard finite element, like edge based FEM , or standard finite volume schemes in graphical processor units (GPU) and particle finite element method arise to be the most predominant alternatives towards this target. These methods and reduced order models also plan to reach real time performance, therefore a big effort should be put in order to fulfill these hard requirements. All these methods have their own advantages and disadvantages. Without enter into details about the reason of selecting the particle finite element method in this paper we try to present the new features of the second generation of this method, called PFEM-2. In [1] the main characteristics of PFEM-2 were introduced where the conclusion was the ability of this novel method to manage very large time steps in a robust way. Here a brief overview of this method is presented, specially the X-IVAS time integration scheme and both, moving and fixed mesh versions are revisited showing their advantages and disadvantages. In [2] some implementation improvements were presented in order to produce a high performance computing. Parallel issues in shared memory architectures with OpenMP were introduced. In [3] an extension to distributed memory architectures is presented. From the analysis carried out in [1] and [2] it is obvious that moving mesh version has the advantage of reducing the numerical diffusion introduced by the projection between mesh and particles and also employs fewer particles for the same level of accuracy. However, mainly the permanent remeshing and also the implicit solution of Poisson equation for the pressure-velocity coupling demand higher cpu time with the drawback of their scalability. Being parallel remeshing currently a big challenge, the development of PFEM-2 was moved towards its fixed mesh version where the remeshing is not needed at all and due to the fact that the mesh remains fixed the Poisson equation matrix is constant for single fluids. A priori factorization used for constant parameter in scalar and NS equations allows to reduce the cost significantly as it is shown in [2,3]. This feature allow to factorize the matrices involved in the computation once at the beginning keeping in memory for future usage reducing the time of assembling and mostly for solving the linear systems. With this nice feature PFEM-2 not only solves the Poisson equation in more efficient way also allows to enlarge the time step drastically solving implicitly the diffusive part of all the equations involved. Up to this point only scalar advection-diffusion problems and Navier-Stokes equations were solved. In [4] a thermally coupled flow solver was presented where the emphasis was put on the accuracy without searching a finer improvement in the perfomance. However the idea is to show how PFEM-2 behaves in problems where several degrees of freedom are solved, as in multi-species flow typical in reactive (combustion) problems. The inclusion of more real industrial applications push the method on a challenge. Turbulent flow in reactive or multi-species problems put a limitation in the usage of a priori factorization because of the changes in the diffusivity of all the fields involved in the mathematical model. Also in multifluids the Poisson equation has their own limitation because the density changes when the different fluids with different physical properties moves over the fixed mesh in background. In the next sections some details about how to solve linear systems with matrices that changes mainly for their physical parameters keeping the mesh fixed are going to be discuss. On the other hand the moving mesh version is also under development. This version has a hard limitation in the lack of scalable parallel remeshing algorithms. Without enter into details in this paper a divide and conquer algorithm is being used for this task with the target in becomes this version competitive.

2 A REVIEW OF PFEM-2

The particle finite element method (PFEM) proposed originally in [8] inspired in the finite point method, a conceptually meshless method uses a lagrangian formulation combined with a fractional step method to solve the pressure with a background mesh that serves to integrate the corresponding equations using a finite element discretization. This method had been applied to solve fluid-structure interaction problems with very success.
However in the opinion of their authors for several problems the time step involved in the computations becomes so small that demands a lot of cpu time losing its attractive feature of being almost a meshless method.
Among the main drawbacks detected that may be responsible for this efficiency decreasing the rough time integration and the moving mesh dynamics may be cited. The former produces a bad integration of the nonlinearities and the later restricts severely the time step.
In order to circumvent these drawbacks the second generation of PFEM was introduced.

Let $\mathbf{x}_p$ be the vector defining the position of a particle in a 3D space, function of the time t that we will write for simplicity $\mathbf{x}_p^t$. 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}

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

where $\dfrac{D\phi}{Dt}$ 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: knowing 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 in 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). In this paper we will be concern with the time integration only, being possible to use any space discretization to achieve analogous results.

2.1 Time integration of the velocity

Equation (2.3) may be approximated in different ways. The simplest one is the constant velocity explicit integration in which the velocity is considered constant in the whole time interval with the value of the velocity at the initial position:

$$\mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \mathbf{V}^n(\mathbf{x}_p^n)\, \Delta t \tag{2.5}$$

Another possibility is the linear velocity implicit integration in which the velocity is considered with a linear variation between t^n and t^{n+1}:

$$\mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \mathbf{V}^n(\mathbf{x}_p^n)\,(1-\theta)\Delta t + \mathbf{V}^{n+1}(\mathbf{x}_p^{n+1})\,\theta\,\Delta t \tag{2.6}$$

with θ being a parameter varying between 0 (explicit) and 1 (fully implicit).

The difficulty of (2.6) is the fact that velocity $\mathbf{V}^{n+1}$ is unknown. This means that is part of the variables to be solved. For this reason implicit methods introduce a non-linear system of equation.

The question is: are there other explicit formulations better than the constant velocity approximation used in (2.5)? Of course there are. We can use previous time steps, like t^{n-1}, t^{n-2}, etc. to approximate high order time curves. We propose here a different way to improve the explicit integration. The idea 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

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

Equation (2.7) is explicit because we are using only information at time step t^n. In this case we are not using a constant none 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 (inside each time step) following a pseudo trajectory of the particles calculated with the velocity streamline, instead of following the true trajectory (Fig. 2.1). It must be noted that in the stationary case, the particle position evaluated with the velocity streamlines and the trajectory are coincident. This time integration will be named Explicit Integration following the Velocity Streamlines (X-IVS).

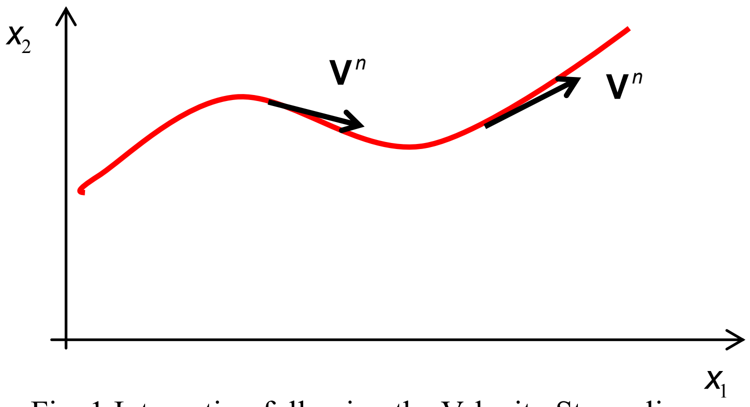

Fig. 1 Integration following the Velocity Streamlines

2.2 Time integration acceleration

In classical explicit integration, equation (2.4) is replaced by:

$$\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \mathbf{A}^n(\mathbf{x}_p^n)\Delta t \tag{2.8}$$

The value of $\mathbf{x}_p^{n+1}$ may be evaluated with any of the possibility described before:

a) the fully explicit case, that is using (2.5), reduce to:

$$\begin{cases} \mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \mathbf{V}^n(\mathbf{x}_p^n)\,\Delta t \\ \mathbf{V}^{n+1}(\mathbf{x}_p^{n+1}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \mathbf{A}^n(\mathbf{x}_p^n)\Delta t \end{cases} \tag{2.9}$$

b) the X-IVS case, that is using (2.7), remains:

$$\begin{cases} \mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \int_n^{n+1} \mathbf{V}^n(\mathbf{x}_p^t)\,dt \\ \mathbf{V}^{n+1}(\mathbf{x}_p^{n+1}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \mathbf{A}^n(\mathbf{x}_p^n)\Delta t \end{cases} \tag{2.10}$$

In implicit (linear) integration equation (2.4) is replaced by

$$\mathbf{V}^{n+1}(\mathbf{x}_p^{n+1}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \mathbf{A}^n(\mathbf{x}_p^n)(1-\theta)\Delta t + \mathbf{A}^{n+1}(\mathbf{x}_p^{n+1})\theta\,\Delta t \tag{2.11}$$

which also may be used with any of the previous time integrations for the particle position.

However, we will propose something new for evaluating the velocity: the idea proposed in (2.7) 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:

$$\mathbf{V}^{n+t}(\mathbf{x}_p^{n+t}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+t} \mathbf{A}^n(\mathbf{x}_p^\tau)\,d\tau \tag{2.12}$$

Equation (2.12) 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. For consistence, we will use the X-IVS method described in (2.7):

$$\begin{cases} \mathbf{x}_p^{n+t} \approx \mathbf{x}_p^n + \int_n^{n+t} \mathbf{V}^n(\mathbf{x}_p^\tau)\,d\tau \\ \mathbf{V}^{n+t}(\mathbf{x}_p^{n+t}) \approx \mathbf{V}^n(\mathbf{x}_p^n) + \int_n^{n+t} \mathbf{A}^n(\mathbf{x}_p^\tau)\,d\tau \end{cases} \tag{2.13}$$

We must note that equations (2.13) are still explicit because they are using the velocity and acceleration at time t^n (Fig.2.2). Nevertheless, they do not assume any constant or linear variation in time neither the velocity nor the acceleration, as is the standard assumption in classical explicit or implicit integration scheme. This approach will be named Explicit Integration following the Velocity and Acceleration Streamlines (X-IVAS).

The position of the particles x_p^{n+1} is used for remeshing in the moving mesh version or for the projection between particles and mesh in the fixed mesh version. As it was mentioned in the introduction both versions present some advantages and disadvantages. However the fixed version seems to be much more efficient than the first one. In this version a fixed mesh is used with particles that move over the mesh in background. One of the nice features of this version is the removing of the remeshing. Another one is the efficiency in the resolution of the linear system based on the fact that the matrix involved in the linear system that being constant may be factorized once at the beginning and used along the simulation in an efficient way. This feature has shown this advantage in [2] for constant physical parameters. In the next sections this feature is revisited in cases where the mesh remains fixed but the physical coefficients changes with space and time, typical in real engineering situations.

3 A PRIORI FACTORIZATION METHOD TO SOLVE POISSON EQUATION FOR VELOCITY-PRESSURE COUPLING

As it was shown in the above section the velocity-pressure coupling in PFEM-2 is carried out via a Poisson like pressure equation. For single phase problems, where the density is constant, this equation may be written as:

$$\Delta t * \nabla \bullet \left(\frac{1}{\rho} \nabla \left(\theta * p^{n+1} + (1-\theta) * p^n \right) \right) = \nabla \bullet v \tag{3.1}$$

In multifluids or multiphase problems the density becomes a function of another variable to be solved, the void fraction, i.e:

$$\rho = \rho(vof(x,t)) \tag{3.2}$$

$$\frac{D(vof)}{Dt} = 0 \tag{3.3}$$

Therefore, the density is not more constant, neither in space nor in time avoiding the usage of the a priori factorization. Moreover in mutiphase flows where the resolution scale produces a strong separation of the fluids through a very thin interface the density jumps. This feature has an impact on the stiffness of the linear system to be solved.

In order to keep the advantage of an a priori factorization instead of solving the linear system with a direct method we change for an iterative one using a preconditioner based on an a priori factorization of a matrix that remains constant and another simple diagonal matrix that change at each time step. The former is built from the topological mesh with a constant diffusivity (density), remember that the left side of the Poisson equation behaves like a diffusion, and the later contains the inverse of the square root of the density and apply both at left and right of the former, i.e:

$$K_\rho * U = F \tag{3.4}$$

$$K_\rho \approx P = D * K_{\rho=1} * D = D * (R^T * R) * D \tag{3.5}$$

$$D = diag\left(\sqrt{\frac{1}{\rho}}\right) \tag{3.6}$$

$$R = IC\left(K_{\rho=1}\right) \tag{3.7}$$

This preconditioning matrix *P* is used inside a preconditioned conjugate gradient iterative method to solve the Poisson equation and it is based on the incomplete Cholesky factorization of the topological part of the matrix that remaining fixed during the computation is only computed once at the beginning.

4 A PRIORI FACTORIZATION METHOD TO SOLVE THE DIFFUSION OF TRANSPORT EQUATIONS

This section extends the above ideas of preconditioning the iterative solver using a preconditioner that contains at one side the topology of the mesh solved as good as possible via a incomplete Cholesky factorization corrected by the variable in space and time physical diffusivity via a very simple diagonal matrix. In this case it is applied to the diffusive part of the transport equation that for efficiency reasons is solved implicitly. Explicit schemes for this kind of problems seem to be expensive in terms of cpu time. Even though enhanced schemes may be written to enlarge the time step for heat equation [5], it may be proved that this strategy does not produce advantages in terms of cpu time spent in the computation.

Using a lagrangian formulation and a fractional step methodology for the velocity pressure coupling PFEM-2 may solve the momentum predictor in an explicit way or implicitly. In order to enlarge the time step drastically the last case is preferred but only if the linear system may take advantage of the a priori factorization of the matrix involved. First the momentum equation is split in two steps, first a lagrangian X-IVAS integration is carried out without the diffusive terms and afterwards the diffusive part is solved implicitly. For this second step the following discrete equation is found:

$$\left(M + \Delta t * K_\mu\right) * U = F \tag{4.1}$$

$$\left(M + \Delta t * K_\mu\right) \approx P = Q^T * Q \tag{4.2}$$

$$Q = \sqrt{M} + \sqrt{\Delta t} * R * D$$

$$D = diag\left(\sqrt{\mu}\right) \tag{4.3}$$

$$R = IC\left(K_{\rho=1}\right) \tag{4.4}$$

in an iterative way using again a preconditioned conjugate gradient (PCG) due to the fact that the linear system is symmetric and positive definite.

In this way it is possible to compute the a priori factorization only once at the beginning and afterwards using it with some matrix-vector products for getting a preconditioner for the PCG that behaves in a very good manner for large time steps. If the time step is short, i.e. Fourier number is low, this preconditioner is worser than the standard solution of the linear system but only by a constant. Normally PFEM-2 looks for enlarge the time steps as much as possible in order to accelerate the computation.

5 SOME RESULTS

In this section some results obtained with the preconditioners introduced in the last two sections are presented. In all the cases the domain is a unitary square discretized with different number of degree of freedom with a fixed and constant right hand side.

5.1 STEADY CASES

Three cases were solved, the first one with a constant diffusivity, the second one with the diffusivity parameter varying smoothly in space and the last one with a sharp diffusivity variation through the interface. The first two cases are not included here for brevity reasons. These results show a very good behavior of the preconditioner when compared with the standard diagonal preconditioner, both in iteration numbers and in cpu time. For the last case a parameter P is used for the sharpness. Different ways of factorization were assessed. In terms of cpu time it may be concluded that the incomplete Cholesky (LU) factorization with some tolerance (4 levels) for the filling results a very good choice in general. The case B presents a big ratio between the maximum and the minimum value of the diffusivity but with the minimum value not so close to zero while the case C is similar to case B but with the minimum value close to zero.

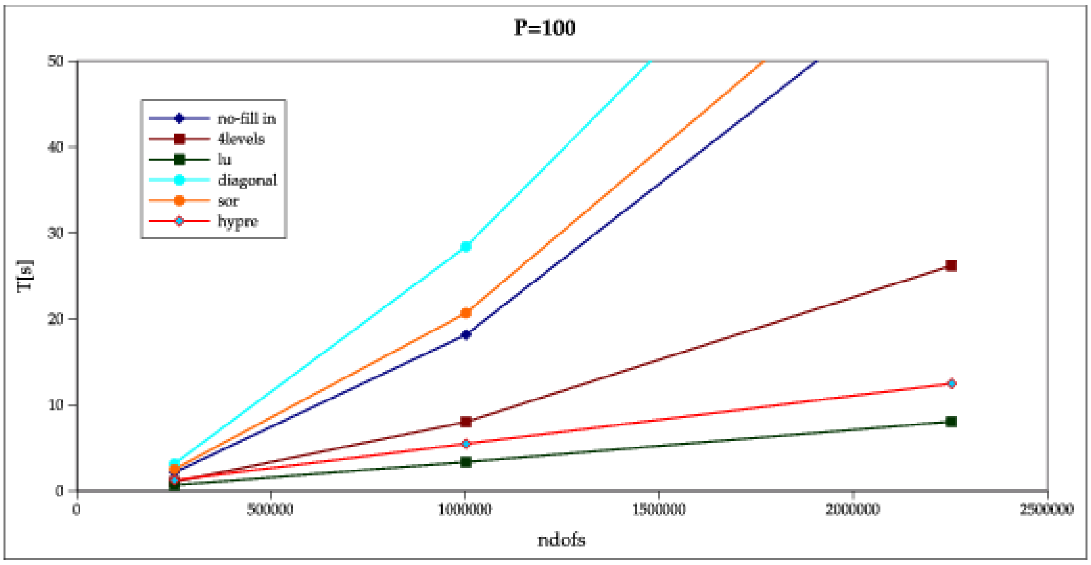

Fig. 2: Case B – Steady – P=100 – CPU time vs #of dofs

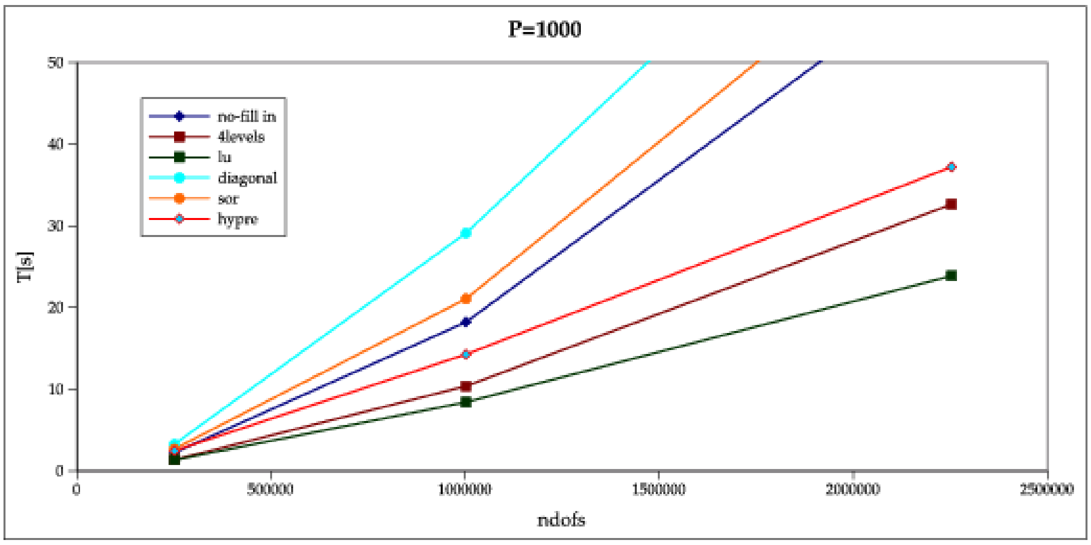

Fig. 3: Case B – Steady – P=1000 – CPU time vs #of dofs

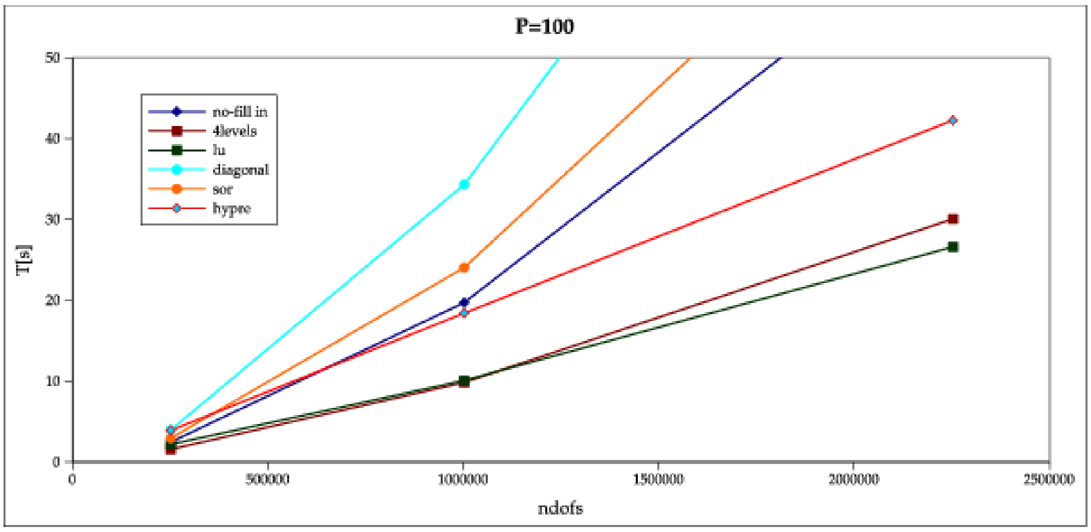

Fig. 4: Case C – Steady – P=100 – CPU time vs #of dofs

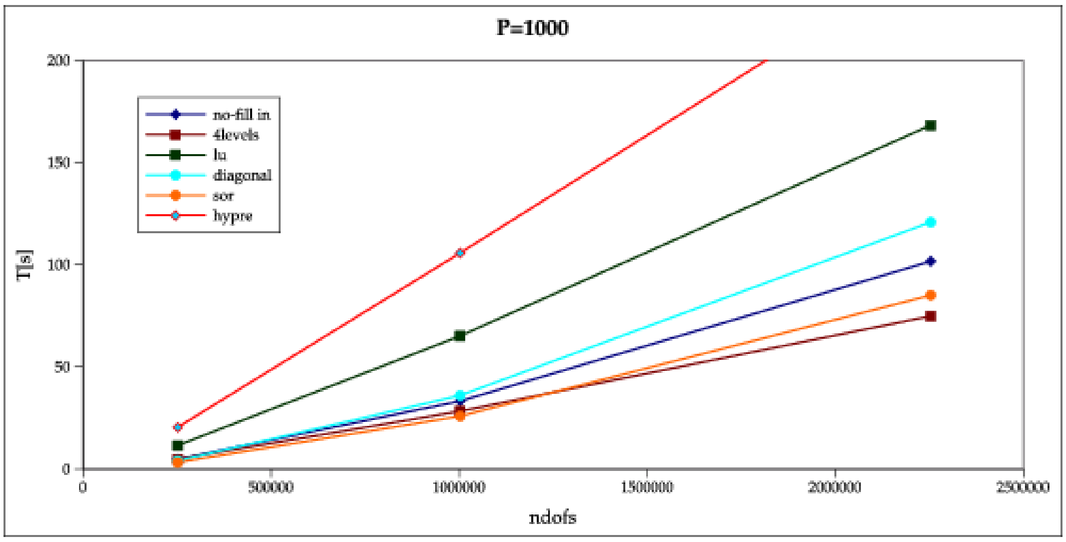

Fig. 5: Case C – Steady – P=1000 – CPU time vs #of dofs

5.2 UNSTEADY CASES

For the unsteady case we have proved the same diffusivity field as before but now in a transient context. As in this case a new parameter is added, the Fourier number, then we plot the cpu time vs Fourier number for a fixed number of dofs.

The results show more or less the same conclusion where it is shown that as large the time step better the new preconditioner and as short the time step better the standard diagonal preconditioner.

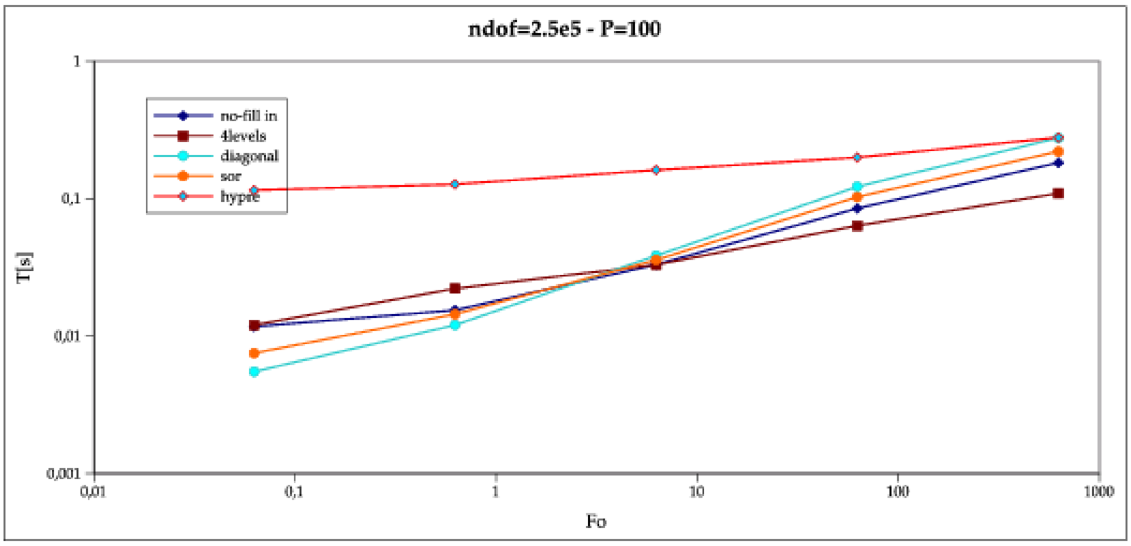

Fig. 6: Case B – Unsteady – P=100 – CPU time vs #of dofs

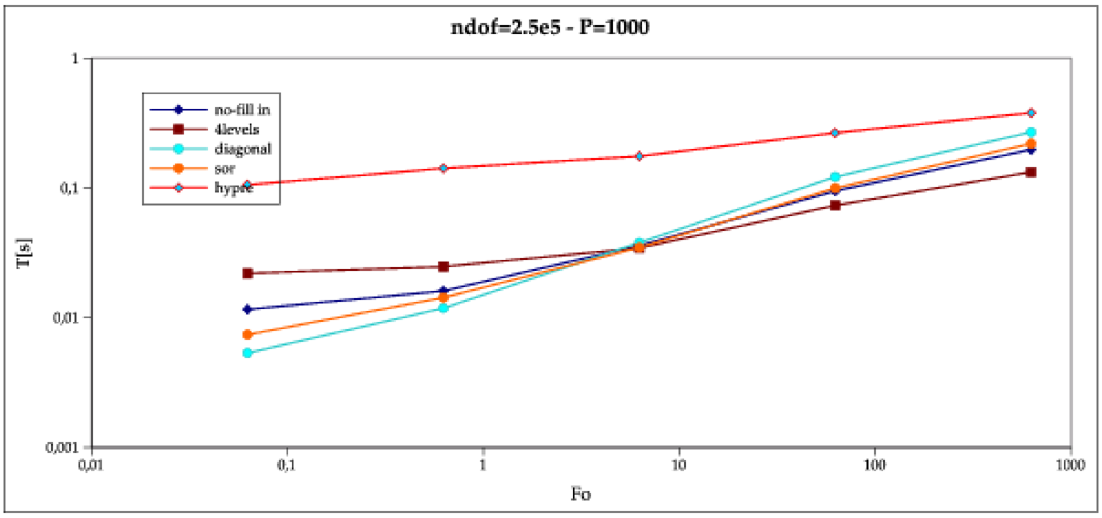

Fig. 7: Case B – Unsteady – P=1000 – CPU time vs #of dofs

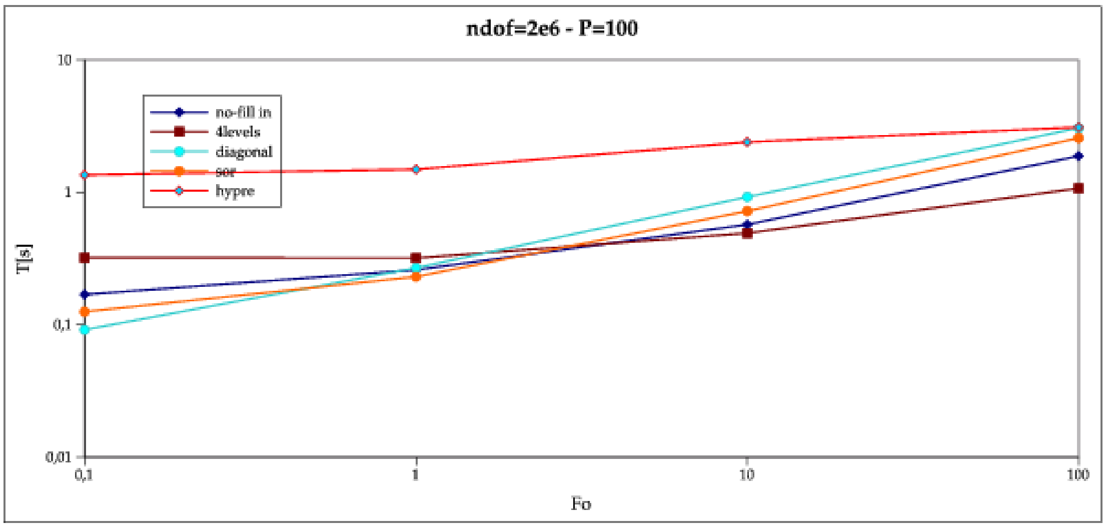

Fig. 8: Case C – Unsteady – P=100 – CPU time vs #of dofs

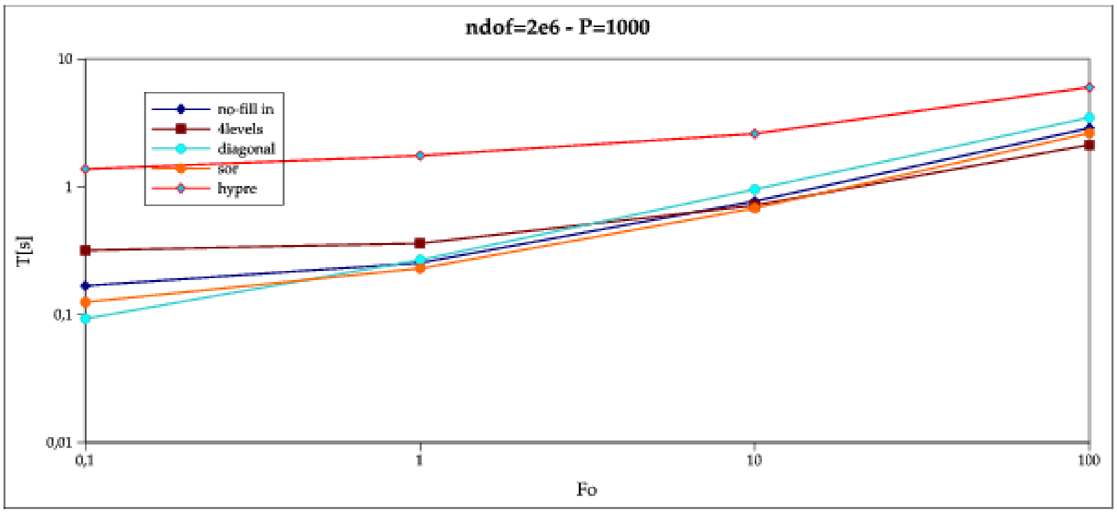

Fig. 9: Case C – Unsteady – P=1000 – CPU time vs #of dofs

6 CONCLUSIONS

- PFEM-2 has shown to be a competitive method to reduce drastically the cpu time involved in the computations due to its inherent stability.
- Also it has an accuracy level acceptable for engineering applications even though the time-steps used are high enough. A better accuracy may be reach reducing the time step arriving at the same level of that reported in scientific papers. The advantage here is the possibility of enlarging or reducing the time-steps at demands in an stable way.
- A good preconditioner for discrete steady and unsteady elliptic equations with variable in space and time physical coefficients is presented in this paper that may be extended for more general applications. This preconditioner allows PFEM-2 to include turbulent effects, reactive systems where multi-species are produced and also multifluids with sharp interfaces.

REFERENCES

[1] Idelsohn, S. R., Nigro N.M., Limache A. and Oñate E. Large time-step explicit integration method for solving problem with dominant convection. *Comput. Methods Appl. Mech. Engrg*. (2012) **217-220**:168–185.

[2] Idelsohn, S. R., Nigro N.M., Gimenez J. M., Rossi R. and Marti J. A fast and accurate method to solve the incompressible Navier-Stokes equations. *Enginnering Computations*. (2013) **30-2**:197-222.

[3] Sklar D., Gimenez J. M., Nigro N.M.and Idelsohn, S. R. Thermal coupling in particle finite element method-second generation. *Mecánica Computacional*. (2012) **31**:4143–4152.

[4] Becker P., Nigro N.M. and Idelsohn, S. R., E. Integración temporal explícita con grandes pasos de tiempo de la ecuación de transmisión del calor. Revista *Comput.Revista Internacional de Métodos Numéricos en Ingeniería* (2012) **28**:187–197.

[5] Gimenez J.M., Nigro N.M. and Idelsohn S.R. Improvements to solve diffusion-dominant problems with PFEM-2, *Mecánica Computacional* (2012), **31:**137-155

[6] Nigro N.M., Gimenez J.M., Limache A., Idelsohn S.R., Oñate E., Calvo N.A., Novara P and Morin P. A new approach to solve incompressible Navier-Stokes equation using a particle method, *Mecánica Computacional,* (2011) **30**

[7] Gimenez J.M. and Nigro N.M., Parallel Implementation of the Particle Finite Element Method, *Mecánica Computacional,* (2011), **30:** 3021-3032

[8] 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, 964-989.

[9] Zienkiewicz, O.C. and Taylor, R.L. *The finite element method*. McGraw Hill, Vol. I., (1989), Vol. II, (1991).

[10] Idelsohn, S.R. and Oñate, E. Finite element and finite volumes. Two good friends. *Int. J. Num. Meth. Engng* (1994) **37**:3323-3341.