UNIVERSIDAD NACIONAL DEL LITORAL

Implementación del método PFEM sobre arquitecturas paralelas

Juan Marcelo Gimenez

FICH

FACULTAD DE INGENIERIA

Y CIENCIAS HIDRICAS

CIMEC

CENTRO DE INVESTIGACION

DE METODOS COMPUTACIONALES

Tesis de Maestría en Computación Aplicada a la Ciencia y la Ingeniería **2014**

UNIVERSIDAD NACIONAL DEL LITORAL

Facultad de Ingeniería y Ciencias Hídricas

Centro de Investigación de Métodos Computacionales

Implementación del método PFEM sobre arquitecturas paralelas

Juan Marcelo Gimenez

Tesis remitida al Comité Académico de MACO

como parte de los requisitos para la obtención

del grado de

MAGISTER en Computación

Aplicada a la Ciencia y la Ingeniería

de la

UNIVERSIDAD NACIONAL DEL LITORAL

2014

Comisión de Posgrado, Facultad de Ingeniería y Ciencias Hídricas, Ciudad Universitaria, Paraje “El Pozo”,

S3000, Santa Fe, Argentina.

And in the end,

the love you take

is equal to the love you make

Agradecimientos

A mi director por su pleno acompañamiento y constante ayuda en la realización de esta tesis.

A mis compañeros del CIMEC, por hacer agradable y ameno el trabajo diario.

A mi novia y familia por estar siempre.

Resumen

El Método de Partículas y Elementos Finitos (PFEM) es un método numérico de discretización espacial híbrida, que utiliza partículas Lagrangianas y malla Euleriana, para la solución numérica de ecuaciones de transporte escritas en la formulación basada en la derivada material, o formulación Lagrangiana. La nueva generación de este método, conocida como PFEM-2, incluye una novedosa estrategia de integración explícita siguiendo las líneas de corriente denominada X-IVAS, que le confiere la capacidad de utilizar grandes pasos de tiempo sin pérdida significante de precisión. Esto permite al método ser competitivo con las clásicas estrategias Eulerianas, aún en su rango óptimo de aplicación como lo son los flujos a una fase. Durante el desarrollo de PFEM-2 surgen diferentes tratamientos para la comunicación de información entre ambas discretizaciones, dando lugar a tres diferentes enfoques que se presentarán y analizarán en este trabajo.

Para evaluar la bondad del método al resolver grandes simulaciones, es imperativo el uso de ambientes paralelos. Las estrategias paralelas para la resolución de problemas de Elementos Finitos han sido ampliamente estudiadas y se encuentran disponibles diversas librerías para resolver las etapas Eulerianas que PFEM-2 contiene. Sin embargo, las etapas Lagrangianas, tales como el mencionado X-IVAS, deben ser desarrolladas atendiendo al caso particular de la discretización híbrida y considerando la estrategia de paralelización seleccionada. Una primera implementación, basada en la arquitectura de memoria compartida, es utilizada para determinar cual es el enfoque PFEM-2 más apropiado para continuar con el desarrollo algorítmico. El principal problema en las simulaciones con este método es la gran cantidad de memoria necesaria, lo que limita su aplicación a grandes problemas cuando se utiliza una sola computadora. Esto conlleva a la necesidad de tener una implementación en la arquitectura de memoria distribuida para poder hacer uso de los denominados clústers de cómputo. A diferencia del enfoque de memoria compartida, en donde es necesario garantizar la seguridad de hilos, en memoria distribuida al utilizar descomposición del dominio se logra aislar automáticamente la

memoria de cada proceso. Sin embargo en ésta última se debe seleccionar una estrategia de particionamiento tanto de la malla como de las partículas que garantice la máxima eficiencia del método, minimizando la comunicación entre procesos y balanceando la carga de trabajo de los mismos. Con el fin de presentar los resultados obtenidos, se analiza el rendimiento de las implementaciones multicore y multinodo resolviendo un conjunto de tests académicos. El número de CFL utilizado influye en la eficiencia de la paralelización y, en algunos casos, debe ser utilizado un particionamiento ponderado para mejorar la escalabilidad del código. Sin embargo, el tiempo total de cómputo de cada uno de los casos presentados en esta tesis, es menor a aquel obtenido al utilizar softwares que implementan las clásicas estrategias Eulerianas.

Finalmente se presentan extensiones del método para la solución de flujos más complejos, como aquellos que requieren modelado de turbulencia o acoplamiento entre las ecuaciones de transporte. Los resultados preservan la capacidad de utilizar grandes pasos de tiempo de forma robusta y sin perder precisión, demostrando así el gran potencial que presenta PFEM-2 al resolver problemas de diversa naturaleza.

Abstract

The Particle Finite Element Method (PFEM) is a numerical method with a hybrid spatial discretization, which uses Lagrangian particles and Eulerian mesh, for the numerical solution of the transport equations written in a formulation based on the material derivative, or Lagrangian formulation. The new generation of this method, known as PFEM-2, includes a novel strategy of explicit integration following the streamlines called X-IVAS, which gives it the capability of using large time-steps without significant accuracy loss. This feature makes the method competitive with the classical Eulerian strategies, even in their range of optimal application as the homogeneous flows. During the development of PFEM-2 different treatments for the communication of data between both discretizations appear. This gives place to three approaches that will be presented and analyzed in this work.

To evaluate the goodness of the method solving large simulations, the use of parallel environments is imperative. Parallel strategies for Finite Element Method have been widely studied and many libraries can be used to solve Eulerian stages of PFEM-2. However, the Lagrangian stages, as the mentioned X-IVAS, must be developed considering the particular case of the hybrid discretization and the parallelization strategy selected. A first implementation, based on the shared-memory architecture, is used to conclude which one is the more appropriate to continue with the algorithmic development. The main drawback of PFEM-2 is the large amount of memory needed, which limits its application to large problems with only one computer. Therefore, a distributed-memory implementation is urgently needed to use the called clusters of computers. Unlike a shared-memory approach where is mandatory to guarantee the thread safety, using domain decomposition the memory of each processor is automatically isolated. However new issues appear due to data distribution over the processes: an optimum partitioning strategy must be selected to minimize the communication between processes and balancing the workload of them. In order to present the results, the performance of multicore and multinode implementations are analyzed by solving a set of academic tests. The Courant–Friedrichs–Lewy

number used influences the efficiency of the parallelization and, in some cases, a weighted partitioning can be used to improve the speed-up. However the total cputime for cases presented is lower than that obtained when using classical Eulerian strategies.

Finally extensions of the method to solve more complex flows, like those which require turbulence modeling or coupling between transport equations, are presented. The results preserve the capability of use large time-step is a stable way and without accuracy loss. This feature shows the great potential offered PFEM-2 when solving problems of various kinds.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Estado del arte y motivación

Las formulaciones estándar para la solución de las ecuaciones de fenómenos de transporte pueden ser agrupadas en dos categorías, dependiendo del enfoque seleccionado para describir los términos inerciales. En el enfoque Euleriano la aceleración es descripta como la suma de una derivada temporal de la velocidad más un término convectivo. El segundo grupo pertenece a las formulaciones Lagrangianas en donde la aceleración es simplemente descripta como una derivada total (o derivada material) de la velocidad.

Durante los últimos treinta años, la simulación de flujo de fluidos incompresibles ha sido principalmente basada en la formulación Euleriana de las ecuaciones fluido-dinámicas sobre dominios fijos [1]. Por otro lado, las formulaciones Lagrangianas justifican su popularidad resolviendo flujos de superficie libre o complicados flujos de fluidos de diferentes propiedades intensivas (multi-fluídos), en los cuales el enfoque Euleriano estándar es impreciso o, en ciertos casos, difíciles de usar.

Los métodos basados en partículas, en los cuales cada partícula fluída es transportada de forma Lagrangiana, han sido utilizados desde que Monaghan [2] propuso las primeras ideas en este enfoque para el tratamiento de problemas de hidrodinámica astrofísica con el método conocido como Smoothed Particle Hydrodynamics Method (SPH). Más tarde, esa estrategia fue generalizada para resolver problemas de la mecánica de fluidos [3]. Una idea similar a SPH fue desarrollada por Koshizuka y otros [4] para la solución numérica de flujos incompresibles y recibió el nombre de Moving Particle Simulation method(MPS). SPH y MPS pertenecen a la familia de los métodos denominados sin malla (meshless). Estas innovadoras ideas de romper con lo concebido hasta el momento y no utilizar mallas tradicionales, fueron generalizadas

por el Meshless Finite Element Method (MFEM) [5]. Este método, que utiliza la Extended Delaunay Tessellation [6] para reconstruir conectividades en tiempo de simulación, toma en cuenta aproximaciones del tipo de elementos finitos para obtener soluciones más precisas.

Una evolución natural de MFEM fue el Particle Finite Element Method (PFEM) [7]. PFEM combina la idea de las partículas Lagrangianas con las funciones de forma del método de los elementos finitos (Finite Element Method - FEM) utilizando una malla auxiliar de este tipo. PFEM ha sido satisfactoriamente utilizado para resolver las ecuaciones de Navier-Stokes con superficies libres [8] [9], problemas de interacción fluido-estructura [10], y problemas de flujos de multi-fluidos [11].

La idea de combinar mallas y partículas móviles fue también utilizada por el denominado Material Point Method (MPM) [12]. Sin embargo, la diferencia más importante entre ambos es que, en PFEM, las partículas no representan una cantidad fija de masa, sino que son solo puntos inmateriales que transportan solo propiedades intrínsecas. Esto último permite incluir una cantidad variable de partículas, lo que simplifica el refinamiento de la malla debido a la posibilidad de utilizar tamaños flexibles de elementos.

Más allá de la evolución presentada en los párrafos anteriores, solo pueden nombrarse unos pocos intentos de resolver flujos de fluidos homogéneos con formulación Lagrangiana. Quizá el trabajo más relevante fue realizado por Joe Stam [13] [14], quien resolvió las ecuaciones de Navier-Stokes en el contexto de los videojuegos dejando el mensaje de que es posible diseñar métodos numéricos simples que puedan ser aplicados en un contexto donde la eficiencia es el punto clave.

Una de las razones de porqué los métodos Lagrangianos no son directamente aplicables en la solución numérica de flujos homogéneos podría ser el importante costo computacional involucrado en la generación y modificación de mallas, grillas o vecindades. Como fue mencionado, los resolvedores Lagrangianos están principalmente basados en mover partículas, y luego de eso algún tipo de conectividad debe ser construída dependiendo de las especificaciones de cada método. Por ejemplo, en PFEM se necesita deformar una malla acorde al movimiento de partículas, pero esto es posible solo hacia cierto límite, más allá del cual hay que remallar. Más allá de que el método ha evolucionado gracias al mallado en paralelo, aparecen otras limitaciones como la impuesta por las no linealidades y otras propias de los esquemas explícitos. Se concuerda entonces que obtener eficiencia con el método PFEM original solo es posible en contadas oportunidades.

Más allá de las limitaciones arriba mencionadas, los esquemas Lagrangianos cuentan con

ciertas características que muestran ciertas ventajas respecto a esquemas Eulerianos. La principal es la ausencia del término convectivo en las ecuaciones de balance, lo que convierte las ecuaciones de no simétricas a simétricas y definidas positivas. Para las ecuaciones de Navier-Stokes este hecho es aún más relevante, debido a que se convierte en lineal la ecuación de conservación de momento. Estas ventajas permiten evitar el uso de términos de estabilización con la fuerte consecuencia de no agregar la típica difusión numérica que se sufre en esos casos, si es que las ecuaciones lineales que quedan son integradas de forma precisa. Es este último punto es donde se ha hecho énfasis para el desarrollo de la nueva generación de métodos PFEM.

Siguiendo la línea antes mencionada, recientemente fue desarrollada una nueva estrategia para la integración de las ecuaciones, la cual es llamada Integración Explícita siguiendo las líneas de corriente de Velocidad y Aceleración (en inglés *eXplicit Integration following the Velocity and Acceleration Streamlines* o simplemente X-IVAS) [15]. Esta forma de integración, que está basada en seguir las líneas de corriente del flujo congeladas en el presente paso de tiempo, es una forma precisa de resolver las no linealidades de la ecuaciones del flujo. Sumando esta estrategia al método original PFEM, se converge en una nueva metodología llamada PFEM-2 [16]. Este método es capaz de resolver flujos complejos pero permitiendo extender de forma significativa el paso de tiempo. Mas aún, preservando la característica de transportar información en las partículas y utilizar la malla para cálculos secundarios, se confiere al método gran precisión.

La discretización espacial híbrida utilizada por PFEM-2 permite seleccionar la estrategia óptima para el cálculo de cada término de las ecuaciones. Como se explicó más arriba, los términos convectivos son resueltos con partículas utilizando el método X-IVAS. Por otro lado, los términos difusivos encuentran mejores resultados al resolverse en una malla, por lo que para estos se utiliza un esquema FEM clásico.

La mencionada dualidad entre malla y partículas requiere enviar información entre cada tipo de discretización. Para ello se han diseñado tres tipos de estrategias, denominadas *Malla Móvil Partícula Móvil* (MMPM), *Malla Fija Partícula Móvil* (MFPM) y *Malla Fija Partícula Fija* (MFPF) respectivamente. A modo de breve resumen, se puede decir que el primer enfoque, siguiendo la idea original de PFEM, genera una nueva malla a cada paso de tiempo con la posición actualizada de las partículas. El segundo utiliza una malla fija de fondo e interpola los estados desde/hacia la nube de partículas viajeras. Finalemente, el tercer enfoque también utiliza una malla fija de fondo, pero las partículas son dispuestas en posiciones fijadas con antelación, sin dejarlas que viajen libremente como en los anteriores casos.

Uno de los principales objetivos del método PFEM-2 es el de resolver problemas de interés

industrial, los cuales usualmente requieren geometrías arbitraria con mallado no estructurado y modelado de la turbulencia, alcanzando números de Reynolds extremos y en donde es necesario obtener buena precisión en el cálculo de las fuerzas que el fluido ejerce sobre cuerpos sólidos. En este marco, se intenta probar que PFEM-2, aún para flujos homogéneos sin superficies libres, es capaz de obtener soluciones precisas y es comparativamente rápido cuando es comparado con otros softwares Eulerianos ampliamente utilizados por ingenieros y científicos en la actualidad, con lo cual el método se transforma en una alternativa real para dichos desarrollos.

Lograr el objetivo de la competitividad del método depende en gran medida del rendimiento o performance de la implementación. Una gran idea algorítmica puede ser desaprovechada si se utiliza una pobre estrategia de programación. Hoy en día, una implementación eficiente debe hacer uso intensivo de las tecnologías de hardware para cómputo paralelo. Un primer paso puede ser el desarrollo de un código que pueda ser ejecutado en computadoras multi-core. El esquema de memoria compartida que se utiliza en estos ambientes de cómputo aparenta ser relativamente simple de programar debido a que todos los procesadores comparten una única visión de los datos y la comunicación entre los procesos es tan rápida como lo permitan los tiempos de acceso a memoria. Sin embargo, en los ambientes multi-core el error más común es la falta de sincronización en el acceso a los datos para escritura: los problemas de concurrencia y reentrancia hacen que su implementación sea un ejercicio no trivial.

Debido a la particular discretización utilizada por PFEM-2, mantener en memoria los datos tanto de la malla como de las partículas representa un importante costo de almacenamiento que limita la operabilidad de la implementación cuando se utiliza una sola computadora. Por lo tanto, para un dado problema y una dada capacidad de memoria, las simulaciones PFEM-2 pueden no caber en RAM mientras que las clásicas utilizando FEM si lo hacen. Por esta razón, rápidamente se hace necesario el desarrollo de una implementación PFEM-2 que sea capaz de resolver dichos problemas de gran escala, que frecuentemente se encuentran en la industria o en la ingeniería. La solución consiste en utilizar la arquitectura de memoria distribuida, pero para ello se debe desarrollar un código que se ejecute de forma eficiente sobre dicha plataforma de cómputo, conformando así el objetivo troncal de esta tesis.

1.2. Objetivos

Como es presentado en el título, el objetivo principal de esta tesis es el desarrollo de código computacional que implemente el método PFEM-2 sobre arquitecturas paralelas de forma

eficiente.

Debido a su reciente y vertiginoso desarrollo, el método PFEM-2 sufre una constante evolución y esta tesis presentará, en primer lugar, el estado actual del método. En esta discusión se analizarán las ventajas y desventajas de cada una de los distintos enfoques desarrollados para la comunicación partícula-nodo.

Las arquitecturas paralelas que se abordarán en esta tesis son la de memoria compartida y la de memoria distribuida. La primera será utilizada para la comparación de enfoques antes mencionada y la segunda, una vez seleccionado el enfoque conveniente, analizará aspectos de cómputo de alto rendimiento, tales como la escalabilidad del código.

Finalmente se espera finalizar con un código eficiente, capaz de simular diversos problemas con grandes requerimientos, en el cual estén aplicadas las últimas tendencias del estado del arte en la materia.

1.3. Esquema de la Tesis

Para favorecer una mejor lectura del contenido de esta tesis, en esta sección se presenta un esquema de la misma.

En el primer capítulo se presenta esta introducción del estado del arte y declaración de objetivos.

En el capítulo 2 se presenta el método PFEM. Partiendo de la novedosa propuesta de integración X-IVAS, se comenta paso a paso el desarrollo de la estrategia, principalmente enfocado a la resolución de problemas de transporte escalar o de flujo incompresible. Aparecen ejemplos que permiten al lector fijar ideas acerca de las diferencias entre los esquemas Lagrangianos y Eulerianos. Al adentrarse en el método PFEM-2, aparecerán los tres enfoques de comunicación nodo-partícula: MMPM, MFPM y MFPF.

El capítulo 3 presenta el núcleo de esta tesis, el cual es la implementación sobre arquitecturas paralelas. El mismo está dividido en dos grandes secciones. En primer lugar, la sección 3.1 presentará la implementación de los tres enfoques PFEM-2 sobre memoria compartida. Esta implementación será utilizada para realizar un análisis de debilidades y fortalezas de cada uno de ellos, decantando en la elección del que se considere más robusto, eficiente y preciso. En la sección 3.2 podrán encontrarse los detalles de la implementación sobre memoria distribuida. Debido a su importancia, se analizarán estrategias de balanceo de carga y de comunicación entre procesos, principalmente orientadas al tratamiento de las partículas. También se presentarán

resultados de típicos problemas de banco de pruebas, tanto de transporte escalar como de flujo incompresible, pero enfocando el análisis en el rendimiento de la implementación. Este rendimiento será notablemente superior dependiendo del hardware utilizado para comunicación entre nodos.

El capítulo 4 presenta ejemplos concretos de las potencialidades del método PFEM-2 para resolver diferentes tipos de problemas. Se seleccionan para esta tesis un caso de flujo externo, un problema de flujo turbulento y un problema de acoplamiento fluido-dinámico y térmico, los cuales son resueltos por medio de la implementación en memoria distribuida presentada en el capítulo previo.

Por último, el capítulo 5 presenta las conclusiones y el trabajo a futuro, haciendo una reseña de las líneas de investigación a seguir hacia la tesis de doctorado por parte del autor.

Capítulo 2

El Método de Partículas y Elementos Finitos

2.1. X-IVAS: Integración temporal de los términos convectivos

Sea $\mathbf{x}_p$ el vector que define la posición de una partícula en el espacio 3D, función del tiempo t. Por simplicidad se usará la notación $\mathbf{x}_p^t$ para denotar dicho punto. Entonces, a tiempo $t = t^n$ se escribirá $\mathbf{x}_p^n$ y a tiempo $t = t^n + \Delta t = t^{n+1}$ se denotará la posición como $\mathbf{x}_p^{n+1}$.

Sea $\mathbf{v}^t(\mathbf{x}_p^t)$ y $\mathbf{a}^t(\mathbf{x}_p^t)$ dos vectores definiendo la velocidad y la aceleración de la partícula $\mathbf{x}_p$ a cualquier tiempo t:

$$\mathbf{v}^t(\mathbf{x}_p^t) = \frac{D\mathbf{x}_p^t}{Dt} \tag{2.1}$$

$$\mathbf{a}^t(\mathbf{x}_p^t) = \frac{D\mathbf{v}^t(\mathbf{x}_p^t)}{Dt} \tag{2.2}$$

en donde $D\phi/Dt$ representa la derivada material (Lagrangiana) para cualquier función ϕ. La derivada material está relacionada con la derivada espacial (Euleriana) por medio de los términos convectivos:

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi \tag{2.3}$$

En todo problema de valores iniciales, como lo son las ecuaciones de Navier-Stokes, la solución temporal de un problema consiste en, dadas todas las variables a tiempo $t = t^n$, encontrar las mismas variables a tiempo $t = t^{n+1}$. En otras palabras, esto equivale a integrar las ecuaciones

2.1 y 2.2:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_n^{n+1} \mathbf{v}^\tau(\mathbf{x}_p^\tau)\ d\tau \tag{2.4}$$

$$\mathbf{v}_p^{n+1}(\mathbf{x}_p^{n+1}) = \mathbf{v}_p^n(\mathbf{x}_p^n) + \int_n^{n+1} \mathbf{a}^\tau(\mathbf{x}_p^\tau)\ d\tau \tag{2.5}$$

La precisión de los resultados dependerá en gran medida de la precisión de la discretización de la velocidad y la aceleración en el espacio, pero también en la aproximación introducida en la integración de las ecuaciones 2.4 y 2.5.

2.1.1. Integración Temporal de la Velocidad

La idea del método X-IVAS es utilizar la velocidad de las líneas de corriente obtenidas en el paso de tiempo t^n para aproximar la posición final de la partícula $\mathbf{x}_p^{n+1}$. Se tiene entonces:

$$\mathbf{x}_p^{n+1} \approx \mathbf{y}_p^{n+1} = \mathbf{x}_p^n + \int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau \tag{2.6}$$

La ecuación 2.6 es explícita ya que solo se utiliza información a tiempo t^n. Cabe destacar que la aproximación del campo de velocidades tiene el mismo alto orden que tiene su solución a tiempo t^n. La única diferencia con la integración exacta 2.4 es que aquí se realiza una integral (dentro de cada paso de tiempo) siguiendo una pseudo-trayectoria calculada con las líneas de corriente congeladas, en vez de utilizar la trayectoria real. Una vez discretizada la ecuación 2.6, puede integrarse analíticamente o utilizando algún esquema estándar como Runge-Kutta o alternativamente con una estrategia de sub-stepping (sub-pasos)[1].

2.1.2. Integración Temporal de la Aceleración

La idea propuesta en la ecuación 2.6, que permite mejorar la integración temporal sin perder la explicitud del cálculo, puede también ser extendida al calculo de la velocidad de la partícula. Es decir, para aproximar 2.5 se hace:

$$\mathbf{v}_p^{n+1} \approx \mathbf{v}_p^n(\mathbf{x}_p^n) + \int_n^{n+1} \mathbf{a}^n(\mathbf{x}_p^\tau)\ d\tau \tag{2.8}$$

[1] El integrador utilizado en este trabajo se encuadra dentro de los algoritmos de sub-stepping. El mismo es heredado del integrador STS [17], el cual adapta su δt dependiendo del número de CFL local.

La expresión utilizada para δt es

$$\delta t_p = \frac{\Delta t}{K \mathrm{CFL}_h} \tag{2.7}$$

en donde $\mathrm{CFL}_h = \dfrac{|\mathbf{v}|\Delta t}{h}$ es el número de Courant del elemento que contiene la partícula, y K es un parámetro para ajustar el mínimo número de sub-pasos requerido para que una partícula atraviese un elemento.

La ecuación 2.8 representa una integración siguiendo las líneas de corriente de la aceleración obtenidas a tiempo t^n. Esto puede ser resuelto de forma acoplada con el cálculo de la trayectoria.

$$\begin{cases} \mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^\tau)\, d\tau \\ \mathbf{v}_p^{n+1} \approx \mathbf{v}_p^n(\mathbf{x}_p^n) + \int_n^{n+1} \mathbf{a}^n(\mathbf{x}_p^\tau)\, d\tau \end{cases} \tag{2.9}$$

Nuevamente, se debe notar que 2.9 sigue siendo explícito ya que se utilizan las velocidades y aceleraciones a tiempo t^n.

2.1.3. La Integración X-IVAS aplicada a las ecuaciones de Navier-Stokes incompresible

En las ecuaciones de Navier-Stokes, la formulación Lagrangiana de la ecuación de conservación de momento es:

$$\frac{D\mathbf{v}^t}{Dt} = \mathbf{a}^t(\mathbf{x}_p^t) \tag{2.10}$$

en donde la aceleración $\mathbf{a}$ es:

$$\mathbf{a}^t(\mathbf{x}_p^t) = \frac{1}{\rho^t(\mathbf{x}_p^t)} \left[\nabla \cdot \boldsymbol{\sigma}^t(\mathbf{x}_p^t) + \mathbf{b}^t(\mathbf{x}_p^t)\right] \tag{2.11}$$

con el tensor de esfuerzos

$$\boldsymbol{\sigma}^t(\mathbf{x}_p^t) = \boldsymbol{\tau}^t(\mathbf{x}_p^t) - p^t(\mathbf{x}_p^t)\mathbf{I} \tag{2.12}$$

y el tensor deviatórico

$$\boldsymbol{\tau}^t(\mathbf{x}_p^t) = \mu \left[\nabla \mathbf{v}^t(\mathbf{x}_p^t) + \nabla^T \mathbf{v}^t(\mathbf{x}_p^t)\right] \tag{2.13}$$

en donde ρ es la densidad, μ es la viscosidad, p la presión, $\mathbf{b}$ es la fuerza de volumen, $\mathbf{I}$ es el tensor identidad, y ∇^T es el operador gradiente transpuesto.

La ecuación de conservación de masa dicta que:

$$\frac{\partial \rho^t(\mathbf{x}_p^t)}{\partial t} + \nabla \cdot \left[\rho^t(\mathbf{x}_p^t)\mathbf{v}^t(\mathbf{x}_p^t)\right] = 0 \tag{2.14}$$

Utilizando la idealización de flujo incompresible se tiene que $\rho^t(\mathbf{x}_p^t) = \rho(\mathbf{x}_p) = \rho_p = cte > 0$, por lo que la Ecuación 2.14 se transforma en:

$$\nabla \cdot \mathbf{v}^t(\mathbf{x}_p^t) = 0 \tag{2.15}$$

Utilizando el método X-IVAS presentado anteriormente, las ecuaciones de Navier-Stokes entre dos instantes de tiempo t^n y t^{n+1} quedan:

$$\begin{cases} \mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \displaystyle\int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau \\ \mathbf{v}_p^{n+1} \approx \mathbf{v}_p^n(\mathbf{x}_p^n) + \frac{1}{\rho_p} \displaystyle\int_n^{n+1} \left[\nabla \cdot \boldsymbol{\sigma}^n(\mathbf{x}_p^\tau) + \mathbf{b}^n(\mathbf{x}_p^\tau)\right]\ d\tau \end{cases} \tag{2.16}$$

La condición de incompresibilidad de la Ecuación 2.15 no presenta dependencia temporal por lo que necesita ser tratada de forma implícita, a menos que se permita cierto grado de compresibilidad (ver Sección 2.1.6). Por el contrario, es posible encontrar la solución de la Ecuación 2.16 por medio de técnicas explícitas.

El uso del esquema X-IVAS garantiza la utilización de pasos de tiempo largos en problemas donde domina la convección, ya que no se encuentra limitado por el número de Courant-Friedrich-Levy CFL $= |\mathbf{v}|\Delta t/\Delta x$ [15]. Sin embargo, en problemas en donde la difusión es importante, la presencia del término viscoso implica una limitación práctica en el tamaño del paso de tiempo, ya que se debe garantizar que el número de Fourier $Fo = \mu\Delta t/\rho\Delta x^2$ sea menor que una constante orden de la unidad ($Fo < 1/2$).

Finalmente, se presenta la Figura 2.1 a modo de interpretación gráfica de la estrategia X-IVAS. Cada nodo de la malla sobre la cual la partícula viaja tiene su propia velocidad y aceleración a tiempo n. Estos campos se integran a través de la trayectoria de la partícula, la cual comienza con una posición $\mathbf{x}^n$ y velocidad $\mathbf{v}^n$ y finaliza el paso de tiempo con el par $[\mathbf{x}^{n+1}, \mathbf{v}^{n+1}]$.

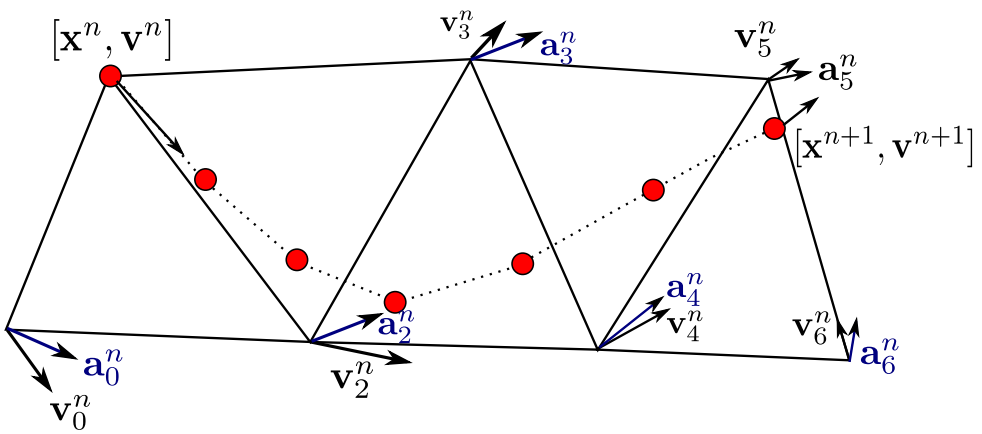


Figura 2.1: Representación gráfica del esquema de integración X-IVAS.

2.1.4. La Integración X-IVAS aplicada a las ecuaciones de transporte escalar

Una de las ventajas del método X-IVAS es que, pese a que fue pensado para la solución de las ecuaciones de Navier-Stokes, su filosofía puede ser aplicada a cualquier ecuación de transporte preservando las ventajas y limitaciones descriptas en la sección anterior.

La ecuación de transporte escalar descripta de forma Lagrangiana es:

$$\frac{D\phi^t}{Dt} = \mathbf{g}^t(\mathbf{x}_p^t) + Q^t(\mathbf{x}_p^t) \tag{2.17}$$

en donde se define la tasa de cambio de la variable $\mathbf{g}$ como:

$$\mathbf{g}^t(\mathbf{x}_p^t) = \nabla \cdot \left(\alpha \nabla \phi^t(\mathbf{x}_p^t)\right) \tag{2.18}$$

siendo ϕ la variable escalar a transportar y α la tasa de difusividad de la misma.

Para encontrar la evolución de la incógnita ϕ se utiliza la misma estrategia X-IVAS quedando a resolver:

$$\begin{cases} \mathbf{x}_p^{n+1} \approx \mathbf{x}_p^n + \displaystyle\int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^\tau)\ d\tau \\ \phi_p^{n+1} \approx \phi_p^n(\mathbf{x}_p^n) + \displaystyle\int_n^{n+1} \mathbf{g}^n(\mathbf{x}_p^\tau) + Q^n(\mathbf{x}_p^\tau)\ d\tau \end{cases} \tag{2.19}$$

A esta altura cabe destacar ciertos puntos:

- En el caso de un transporte escalar puro, el campo de velocidades $\mathbf{v}$ es conocido a cualquier tiempo t.

- En el caso de solución de un problema de flujo incompresible con transporte de un escalar pasivo, el campo de velocidades $\mathbf{v}$ es el campo solución del problema 2.16 asociado.

- En el caso de solución de un problema de flujo incompresible con transporte de un escalar activo, como por ejemplo cuando ϕ es la temperatura, puede utilizarse un término de acoplamiento. Para el caso citado de la temperatura, esto puede lograrse por medio de la aproximación de Boussinesq, que propone:

  $$\mathbf{b}^n = \rho g \beta (\phi^n - \phi_c) \tag{2.20}$$

  donde $\beta \approx \frac{\partial(\log \rho)}{\partial \phi}|_{P=p-\rho g h}$ es la constante de Boussinesq, g es la gravedad y ϕ_c es una temperatura de referencia.

2.1.5. Corrección implícita de los términos difusivos

Como se mencionó, si se resuelve el término difusivo de forma explícita (ya sea utilizando X-IVAS o cualquier otro método) el avance temporal está restringido por $Fo < \text{cte} \approx 1/2$. Esto es una limitación fuerte en el paso de tiempo, especialmente para mallas muy refinadas en la vecindad de cuerpos para capturar capas límites, o de forma más general, en aquellos problemas de difusión localmente dominante ($Fo_h > \text{CFL}_h$).

En esta sección es presentado un nuevo enfoque para tratar los términos difusivos. El mismo está basado en el método θ y consiste en discretizar la variable no estacionaria utilizando un promedio pesado entre una predicción explicita y una corrección implícita. Se propone entonces:

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \theta \mathbf{g}^{n+1} + (1-\theta)\mathbf{g}^n \tag{2.21}$$

Realizando un primer paso predictor de forma explícita

$$\frac{\hat{\phi}^{n+1} - \phi^n}{\Delta t} = (1-\theta)\mathbf{g}^n \tag{2.22}$$

y corrigiendo la estimación inicial de forma implícita, esto es restando (2.22) de (2.21), lo que queda:

$$\frac{\phi^{n+1} - \hat{\phi}^{n+1}}{\Delta t} = \theta \mathbf{g}^{n+1} \tag{2.23}$$

Para el cálculo de la corrección implícita se utiliza una formulación de elementos finitos (FEM) estándar:

$$[\mathbf{M} + \theta\ \Delta t\ \mathbf{K}]\ \phi^{n+1} = \mathbf{M}\ \hat{\phi}^{n+1} + \Delta t\ \mathbf{F}^{n+\frac{1}{2}} \tag{2.24}$$

en donde $\mathbf{M}$ es la matriz de masa, $\mathbf{K}$ es la matriz de rigidez y $\mathbf{F}$ es el vector de cargas asociados a la discretización estándar FEM, el cual incluye $Q(\mathbf{x})$. Se debe notar que la matriz $[\mathbf{M} + \theta\ \Delta t\ \mathbf{K}]$ no depende del tiempo si se cumple que $\alpha \neq \alpha(t)$ y $\Delta t = cte$, por lo que puede ser pre-factorizada al comienzo de la simulación y utilizada como precondicionador durante la misma, lo que representa una reducción significativa de los tiempos de cómputo. Es importante notar la diferencia con el caso Euleriano, ya que en este último enfoque la prefactorización propuesta en los términos difusivos no se puede aprovechar ya que la dependencia temporal con la velocidad no lo permite.

Además, cabe destacar también que estos mismos conceptos pueden ser extendidos al caso de flujo incompresible, en donde la difusión viene dada por el término viscoso $\nabla \cdot \boldsymbol{\tau}$ y el vector de cargas ahora incluye $\mathbf{b}(\mathbf{x})$.

Un ejemplo comparativo: Transporte de una señal Gaussiana

El problema del transporte de una Gaussiana es un test clásico para demostrar las bondades que ofrecen los métodos Lagrangianos, representados en este caso por X-IVAS, para resolver ecuaciones de transporte y particularmente el tratamiento de la advección e integración temporal. Este caso también hace más evidente la patología que sufren los enfoques Eulerianos explícitos al intentar resolver con CFL > 1. El problema considerado en esta sección consiste en una colina Gaussiana utilizada como condición inicial de un problema de transporte con difusión. El campo de velocidades es un flujo rotante alrededor del centro de un dominio cuadrado. La señal Gaussiana está desplazada del centro del dominio por cierto radio y su forma hace que la señal transportada inicialmente solo sea distinta a cero en una región limitada del dominio. La señal debería ser transportada siguiendo una trayectoria circular y su amplitud debe decrecer conforme al coeficiente de difusión utilizado. La Figura 2.2 presenta la definición del problema.

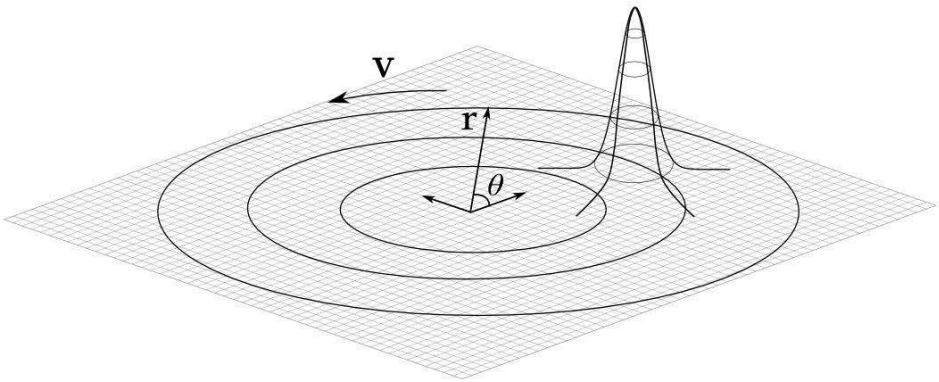


Figura 2.2: Distribución inicial de temperaturas para el problema de transporte sin difusión

El problema se toma del libro de Donea y Huerta [1]. La solución inicial es:

$$\phi(\mathbf{x},0) = \begin{cases} \frac{1}{4}(1+\cos(\pi X))(1+\cos(\pi Y)) & \text{si} \quad X^2+Y^2 \leq 1 \\ \\ 0 & \text{caso contrario} \end{cases} \tag{2.25}$$

donde $\mathbf{X} = (\mathbf{x}-\mathbf{x}_0)/\sigma$ y condición de borde $\phi = 0$. La posición inicial del centro y el radio de la señal son $\mathbf{x}_0$ y σ respectivamente. En el ejemplo se tomó $\mathbf{x}_0 = (\frac{1}{6}, \frac{1}{6})$ y $\sigma = 0.2$. El campo de convección es de rotación pura con velocidad angular $\omega = 2$, esto es $\mathbf{v}(\mathbf{x}) = (-\omega y, \omega x)$. La difusividad elegida es $\alpha = 0.0001$.

Se utilizan dos mallas diferentes, todas sobre el cuadrado unitario $[-\frac{1}{2}, -\frac{1}{2}] \times [\frac{1}{2}, \frac{1}{2}]$. Una

malla gruesa llamada $M1$, de 30×30 elementos cuadrangulares partidos en triángulos. Otra más fina, denominada $M2$, de 100×100.

La Figura 2.3 presenta los resultados utilizando la malla gruesa $M1$. Para el caso de un número de Courant CFL ≈ 0.5, los resultados muestran excesiva difusión en el caso de FEM con esquema temporal de primer orden (Backward Euler), pero buena resolución para el caso de FEM con segundo orden temporal (Crank-Nicholson) y también utilizando X-IVAS. Para el caso de CFL ≈ 5, los resultados muestran excesiva difusión en el caso de FEM con esquema temporal de primer orden. Para FEM a segundo orden se aprecia difusión numérica pronunciada y la aparición de inestabilidades (ripples) debido al integrador temporal. Los resultados de X-IVAS son en general buenos y demuestran que no son sensibles al número de Courant utilizado.

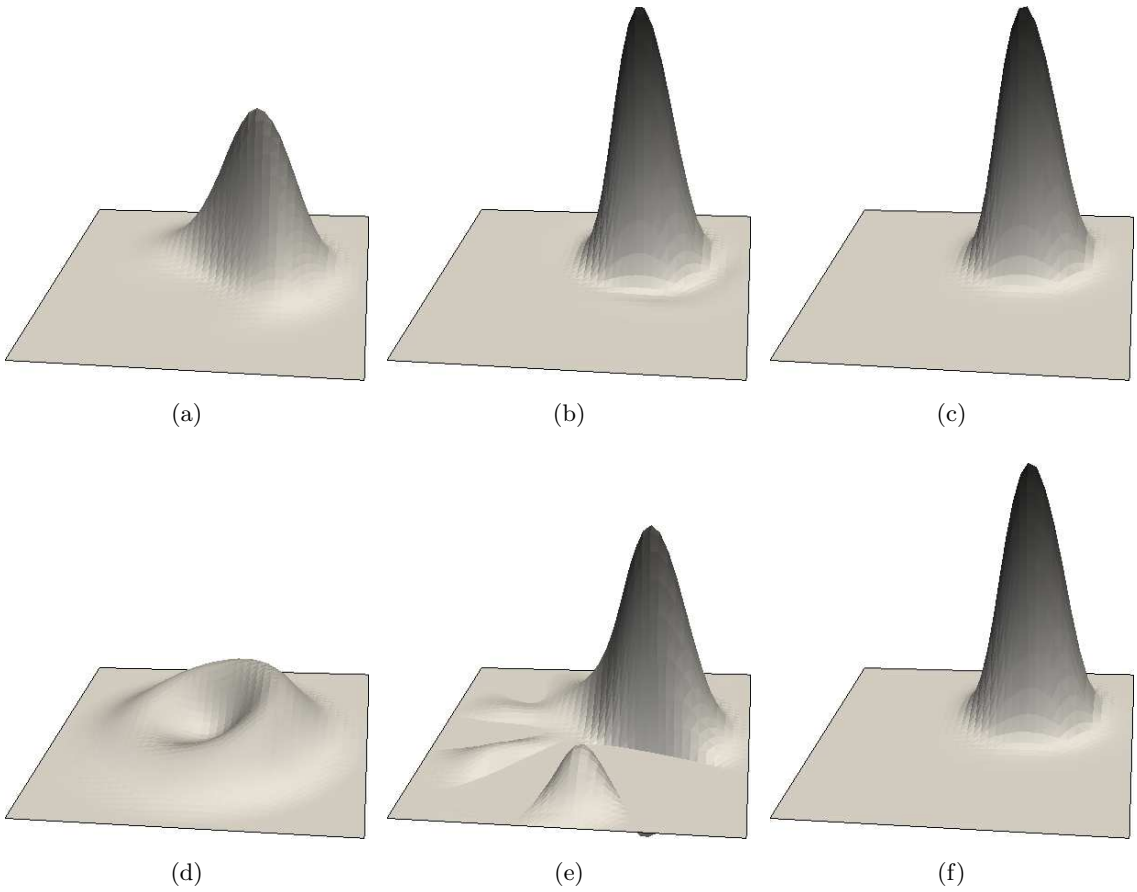


Figura 2.3: Resultados sobre malla $M1$ tras dos giros para `FEM - 1er orden` (Figura 2.3a), `FEM-2do orden` (Figura 2.3b), y `X-IVAS` (Figura 2.3c) utilizando CFL $= 0.5$. En la segunda fila aparecen los resultados sobre la malla $M1$, también tras dos giros, para `FEM - 1er orden` (Figura 2.3d), `FEM - 2do orden` (Figura 2.3e) y `X-IVAS` (Figura 2.3f), pero utilizando CFL $= 5$.

La Figura 2.4 presenta los resultados utilizando la malla más fina $M2$. Para un número de

Courant CFL ≈ 0.5, los resultados muestran nuevamente bastante difusión en el caso de FEM con esquema temporal de primer orden, y otra vez buena resolución para el caso de FEM con segundo orden temporal. En el caso de CFL ≈ 5, los resultados muestran otra vez excesiva difusión en el caso de FEM con esquema temporal de primer orden, pero para FEM a segundo orden temporal ya no se aprecia la difusión numérica que aparecía con malla gruesa y además la aparición de inestabilidades (ripples) debido al integrador temporal ya no es tan pronunciada. Sin embargo los resultados con X-IVAS son en general mejores debido a que siguen presentando menor difusión numérica que FEM.

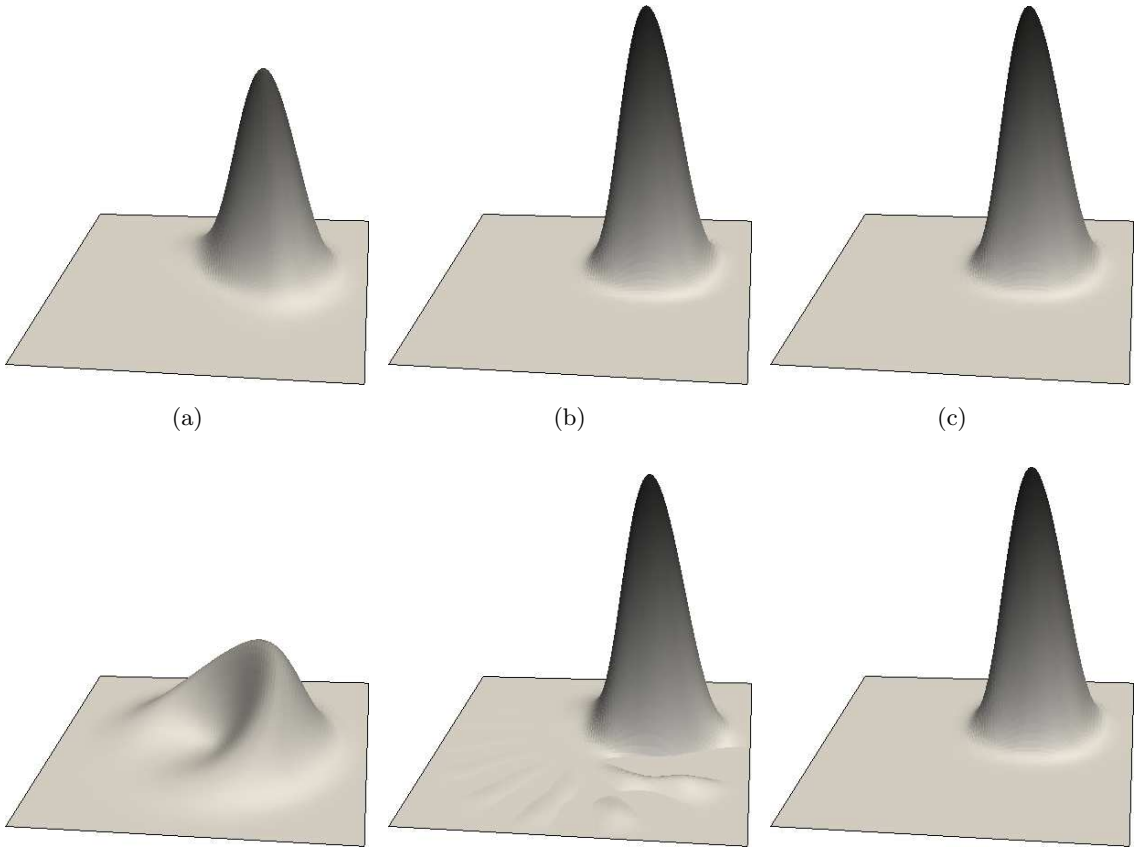
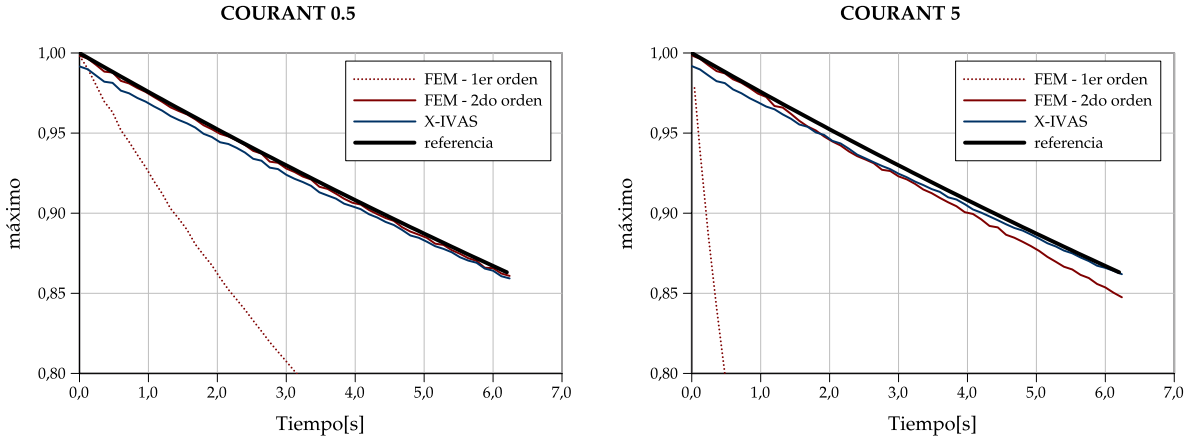


Figura 2.4: Resultados sobre malla $M2$ tras dos giros para `FEM - 1er orden` (Figura 2.4a), `FEM - 2do orden` (Figura 2.4b) y `X-IVAS` (Figura 2.4c) utilizando CFL $= 0.5$. En la siguiente fila aparecen los resultados sobre la malla $M2$, también tras dos giros, para `FEM - 1er orden` (Figura 2.4d), `FEM - 2do orden` (Figura 2.4e) y `X-IVAS` (Figura 2.4f), pero utilizando CFL $= 5$.

En la Figura 2.5 se presenta la evolución del valor máximo de la señal obtenido con las estrategias numéricas de integración y comparados contra una solución de referencia (solución

FEM de segundo orden a CFL bajo en una grilla cartesiana de 1000×1000 nodos). Puede observarse cómo la difusión que introduce el esquema temporal Euleriano es en cierta medida proporcional al CFL; y, aunque utilizar esquemas de segundo orden mejora los resultados, estos no logran tener la misma robustez que al integrar con X-IVAS debido a los ripples que presenta su solución.



(a) (b)

Figura 2.5: Evolución del máximo de la señal sobre la malla $M2$ para CFL $= 0.5$ (Figura 2.5a) y para CFL $= 5$ (Figura 2.5b). X-IVAS muestra la solución proyectada sobre la malla.

2.1.6. Tratamiento de la presión para las ecuaciones de Flujo Incompresible

La Ecuación 2.16 es explícita no sólo por los términos en donde aparece la velocidad, sino también por aquellos en donde está la presión. Como se comentó anteriormente, para flujos incompresibles la presión debe resolverse de forma implícita para que el esquema de integración temporal se preserve incondicionalmente estable. Por esta razón, en la Ecuación 2.26 se incluye el tratamiento implícito para la presión y se considera que el término viscoso se resuelve de forma puramente explícita (aunque, tal como se explicó en secciones anteriores, este último puede resolverse de forma implícita por medio del método θ, pero no se incluye aquí por simplicidad).

$$\begin{cases} \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \displaystyle\int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^\tau)\, d\tau \\ \mathbf{v}^{n+1} \approx \mathbf{v}^n(\mathbf{x}_p^n) + \frac{1}{\rho_p} \displaystyle\int_n^{n+1} \left[\nabla \cdot \boldsymbol{\tau}^n(\mathbf{x}_p^\tau) - \nabla p^\tau(\mathbf{x}_p^\tau) + \mathbf{b}^n(\mathbf{x}_p^\tau)\right]\, d\tau \end{cases} \tag{2.26}$$

en donde el término implícito para la presión puede ser partido de la siguiente manera:

$$\int_n^{n+1} \left[\nabla p^\tau(\mathbf{x}_p^\tau)\right] d\tau = \int_n^{n+1} \left\{\nabla p^n(\mathbf{x}_p^\tau) + \nabla \left[\delta p(\mathbf{x}_p^\tau)\right]\right\} d\tau \tag{2.27}$$

con $\delta p(\mathbf{x}_p^\tau) = p^\tau(\mathbf{x}_p^\tau) - p^n(\mathbf{x}_p^\tau)$.

Con el fin de resolver de forma separada el campo de presiones del campo de velocidades, se utilizará un método de segregación de la presión conocido como el Método de los Pasos Fraccionados, por sus siglas en inglés FSM [18]. Al segregar las ecuaciones se evita tener que resolver un sistema de ecuaciones en donde convivan incógnitas de distintas unidades, lo que mal condiciona la matriz resultante, incrementando el tiempo requerido para la solución de dicho sistema. Por lo tanto, la segunda Ecuación de 2.26 se separa en los siguientes dos pasos:

$$\begin{cases} \hat{\mathbf{v}}^{n+1} = \mathbf{v}^n(\mathbf{x}_p^n) + \frac{1}{\rho_p} \int_n^{n+1} \left[\nabla \cdot \boldsymbol{\tau}^n(\mathbf{x}_p^\tau) - \nabla p^n(\mathbf{x}_p^\tau) + \mathbf{b}^n(\mathbf{x}_p^\tau)\right] \, d\tau \\ \mathbf{v}^{n+1} = \hat{\mathbf{v}}^{n+1}(\mathbf{x}_p^n) - \frac{1}{\rho_p} \int_n^{n+1} \left[\nabla \delta p(\mathbf{x}_p^\tau)\right] \, d\tau \end{cases} \tag{2.28}$$

Aplicando el operador de divergencia a ambos lados de la segunda ecuación de 2.28, y teniendo en cuenta la condición de incompresibilidad (Ecuación 2.15), se tiene una ecuación de Poisson que permite conocer la presión al paso de tiempo actual:

$$\nabla \cdot \hat{\mathbf{v}}^{n+1}(\mathbf{x}_p^{n+1}) = \nabla \cdot \left\{ \frac{\Delta t}{\rho_p} \nabla \left[\delta p(\mathbf{x}_p^{n+1})\right] \right\} \tag{2.29}$$

2.2. Discretización Espacial y los distintos enfoques PFEM-2

El paso de integración explícito de los términos convectivos siguiendo las líneas de corriente (primera Ecuación de 2.28 y Ecuación 2.19) se realiza sobre cada una de las partículas $()_p$ con las cuales se define el muestreo espacial de las propiedades intensivas (viscosidad μ_p, densidad ρ_p, etc) y otras magnitudes físicas ($\mathbf{v}_p$,p_p). Sin embargo, los siguientes pasos implícitos deben realizarse sobre una malla de elementos finitos para poder relacionar el aporte de cada zona del dominio y encontrar una solución global. Esta estrategia de combinar pasos explícitos (X-IVAS) y correcciones implícitas (FEM) utilizando diferentes discretizaciones espaciales para cada uno (partículas y malla respectivamente), recibe el nombre de Particle Finite Element Method Second Generation, o PFEM-2.

La malla utilizada para el cómputo implícito debe tener sus variables actualizadas con los datos calculados por las partículas, para ello es necesario una comunicación entre cada una de estas discretizaciones espaciales, lo que da lugar a las diferentes estrategias o enfoques del

método PFEM-2. La Figura 2.6 presenta un esquema gráfico de cada una de estas opciones, las cuales serán abordadas en las siguientes sub-secciones.

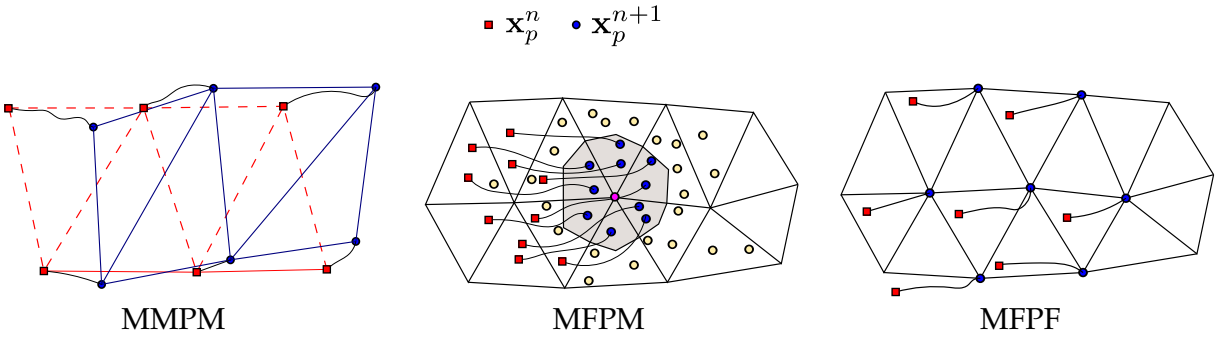


Figura 2.6: Representación gráfica de los tres esquemas propuestos para la comunicación partícula-nodo en PFEM-2. Dependiendo del enfoque, en líneas punteadas se presenta la malla a tiempo n, y en líneas continuas la malla a tiempo $n+1$ (o malla fija de fondo). Al conjunto de partículas de interés se lo presenta con □ a tiempo n y con ◯ a tiempo $n+1$.

2.2.1. PFEM-2 Malla Móvil Partícula Móvil (MMPM)

La primer estrategia consiste en seguir el enfoque del método PFEM original. La idea se basa en que al finalizar la integración de la trayectoria de las partículas siguiendo las líneas de corriente sobre la malla M^n (conectividades a tiempo $t = t^n$), se debe generar una nueva malla M^{n+1} en donde cada partícula pasa a ser un vértice de la nueva malla, la cual utiliza funciones de forma lineales para la aproximación de la solución.

La gran ventaja de esta estrategia es que los vértices adquieren las mismas propiedades físicas que las partículas ya que son la misma entidad geométrica, por lo que se evita la necesidad de realizar algún tipo de promediado o proyección. Sin embargo, esta estrategia tiene puntos negativos a tener en cuenta. Por ejemplo, el remallado es una tarea que acarrea un importante costo computacional y cuyo paralelismo no logra ser demasiado eficiente, además de no poder siempre producir mallas de buena calidad.

Para mejorar la ubicación de los puntos a triangular con el fin de lograr mallas que no contengan elementos demasiado distorsionados o degenerados, como aquellos de área muy cercana a cero, se deben incluir técnicas de remoción y sembrado de partículas. Entonces, para evitar los elementos tipo aguja (sliver), se remueven aquellas partículas que tienen alguna otra partícula muy próxima (dentro de un radio r). Por otro lado, para evitar pérdida de precisión se agregan partículas en aquellas zonas geométricas vacías al final de la integración. La Figura

2.7 presenta un esquema gráfico de los casos aquí citados y la solución propuesta.

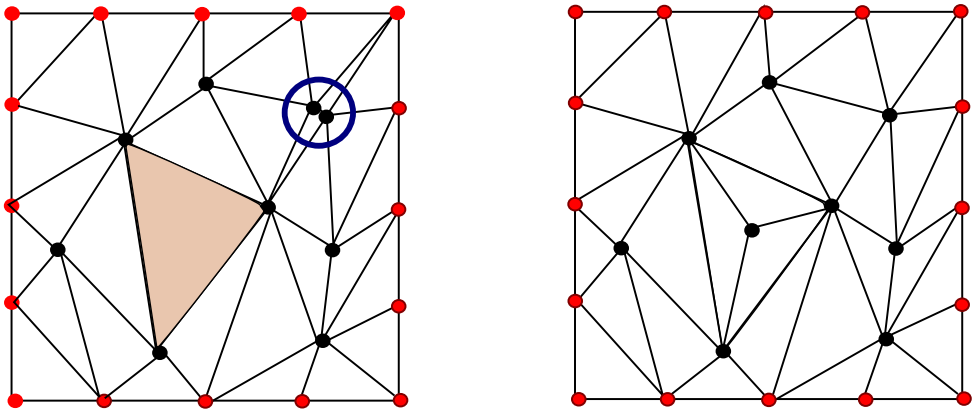


Figura 2.7: Remoción y sembrado de partículas para el enfoque MMPM. A la izquierda (M^n con $\mathbf{x}_p^{n+1}$) se presentan dos problemas: hay una zona geométrica sin partículas y existe un par de partículas con distancia menor a r. A la derecha (M^{n+1} con $\mathbf{x}_p^{n+1}$) se presenta la solución que consiste en sembrar en el primer caso y remover una del par en el segundo. Las partículas rojas no pueden removerse porque son las que identifican las condiciones de borde.

2.2.2. PFEM-2 Malla Fija Partícula Móvil (MFPM)

Con la intención de evitar el constante remallado, este enfoque propone resolver las etapas Eulerianas del algoritmo utilizando una malla fija. La idea es que, luego de la integración de la trayectoria de las partículas, estas proyecten sus variables transportadas a los nodos de la malla de fondo. Con el fin de evitar difusión numérica de los resultados debido a la proyección, es necesario asegurar que alrededor de cada nodo existan suficientes partículas como para obtener una solución correcta.

La gran ventaja de esta estrategia es la de evitar el remallado. Además, al preservar la misma malla toda la simulación, se pueden pre-factorizar las matrices de los sistemas de ecuaciones una única vez (siempre que las propiedades físicas como la difusión y la densidad sean constantes en el tiempo), lo que agiliza la solución de dichos sistemas. Sin embargo deben atenderse nuevos dilemas que aparecen, como lo es el algoritmo de proyección. Para garantizar las suficientes partículas alrededor de cada nodo al momento de proyectar, se requiere una estrategia de sembrado de partículas en donde se necesiten datos. Esto produce, por un lado, difusión al tener que interpolar un estado para la nueva partícula, y además obliga a remover partículas ya que el incremento del número de las mismas puede tornar la simulación inabordable. Estos detalles, un poco más complejos que lo que ocurre en MMPM, son abordados en la secciones

que continúan.

Algoritmos de Proyección

Durante el desarrollo del método se han propuesto diferentes enfoques para realizar las proyecciones de los estados desde las partículas a los nodos. En este trabajo se presentan las estrategias que mejores resultados han logrado. Las ecuaciones son presentadas para la proyección de un valor escalar, pero son aplicables a cada componente de un campo vectorial como la velocidad.

Un operador de proyección $\boldsymbol{\pi}$ es una función que otorga un valor del campo ϕ_j a cada nodo j a partir de los valores del campo ϕ_p sobre las partículas p en la vecindad de dicho nodo, es decir: $\boldsymbol{\pi} \ : \ \phi_j = \boldsymbol{\pi}(\phi_p)$. La formulación genérica es:

$$\phi_j = \frac{\sum_{p=1}^{P} \mathbf{W}_j(\mathbf{x}_p)\phi_p}{\sum_{p=1}^{P} \mathbf{W}_j(\mathbf{x}_p)} \tag{2.30}$$

en donde P es el número de partículas dentro de la región del nodo j. El operador W puede ser una de las funciones kernel como las utilizadas en SPH [19] o las mismas funciones de forma lineales (elevadas a alguna potencia n típicamente $n = 1$) utilizadas para aproximar el campo solución.

Sembrado de Partículas

Como se mencionó, por razones de precisión, los métodos de proyección requieren cierto número de partículas en la región alrededor de cada nodo. La *región alrededor del nodo* debe ser definida de forma precisa, y para un paso de tiempo dado, nada asegura poseer las partículas suficientes en la región de cada nodo, especialmente cuando se utilizan altos números de Courant. Por esta razón, previamente a la proyección, deben crearse nuevas partículas alrededor de dichos nodos.

Una primer idea (S-1) utilizada originalmente en PFEM-2, consiste en asignarle a la nueva partícula una interpolación lineal de los estados nodales del paso de tiempo previo n:

$$\phi_p^{n+1}(\mathbf{x}_p^{n+1}) = \sum_j \lambda_j(\mathbf{x}_p^{n+1})\phi_j^n \tag{2.31}$$

siendo λ_j las coordenadas de área de la partícula en el elemento contenedor.

Una idea mejorada (S-2) busca el estado siguiendo las líneas de corriente, pero en dirección aguas arriba, es decir, buscando la ubicación inicial de la nueva partícula, si es que hubiese

existido al inicio del paso de tiempo, es decir:

$$\phi_p^{n+1}(\mathbf{x}_p^{n+1}) = \phi_p^n(\mathbf{x}_p^n) + \int_n^{n+1} \mathbf{g}^n(\mathbf{x}_p^\tau)\, d\tau \tag{2.32}$$

donde la posición $\mathbf{x}_p^{n+1}$ es elegida al momento de sembrar, pero $\phi_p^n(\mathbf{x}_p^n)$ no es conocida y debe ser interpolada con las coordenadas de área del elemento contenedor una vez hallada la posición $\mathbf{x}_p^n = \mathbf{x}_p^{n+1} - \int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^\tau)\, d\tau$.

Un problema de convección pura, presentado en la Figura 2.8, revela la necesidad inmediata de utilizar la integración hacia atrás. Una condición de borde con una discontinuidad ingresa en el dominio transportada por un campo de velocidades no alineado con la malla de fondo.

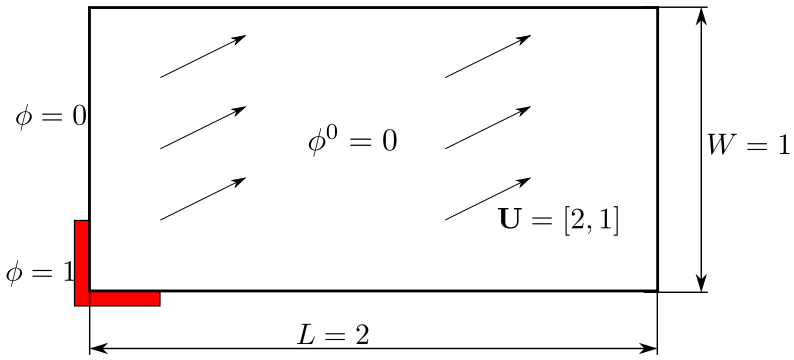


Figura 2.8: Esquema del problema de convección pura para evaluar los valores iniciales asignados a las nuevas partículas.

Las Figuras 2.9 presentan los resultados proyectados sobre una grilla cartesiana de 30×10 nodos mallada con triangulos, para diferentes instantes de tiempo ($T = 0$[s], 0.3[s], 0.7[s] y 1[s]), al utilizar un paso de tiempo tal que CFL > 1 y su comparación utilizando las estrategias (S-1) y (S-2).

Remoción de Partículas

Pese a mejorar la calidad de la proyección, utilizar muchas partículas incrementa los tiempos de cómputo. Si durante la simulación el sembrado es frecuente, es necesario controlar el número de partículas utilizados en la discretización. Por lo tanto es necesario también definir un criterio de remoción.

El caso más simple es cuando las partículas salen del dominio durante la integración X-IVAS: estas deben ser eliminadas. Dentro del dominio se establece el siguiente criterio: cada sub-

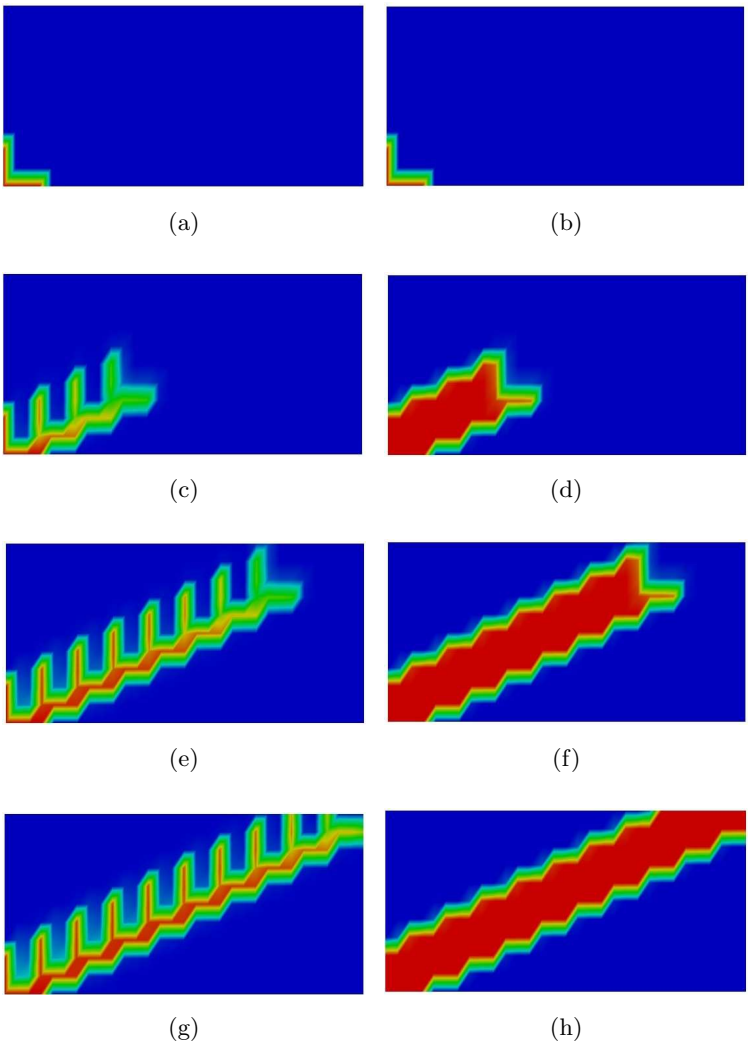
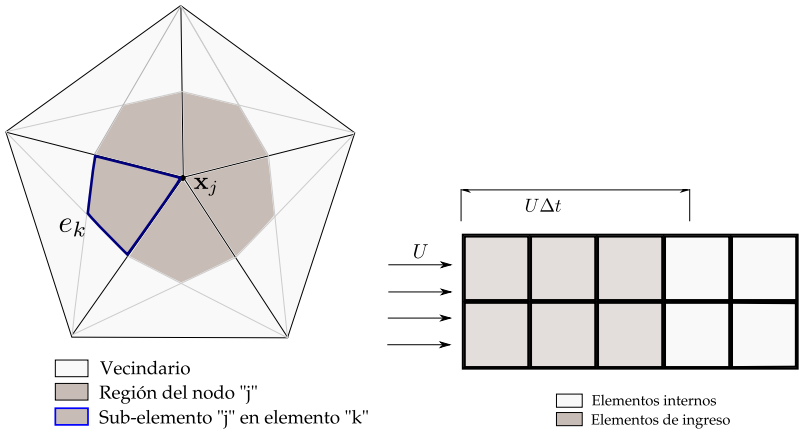


Figura 2.9: Transporte convectivo de un escalón. Resultados para diferentes estrategias de asignación de los nuevos estados a las partículas. A izquierda: S-1. A derecha: S-2.

elemento j (el tercio del elemento -cuarta parte en 3D- en donde la coordenada de área del nodo j es mayor que la del resto de los nodos) debe asegurar al menos $nsubele_{min}$ y poseer a lo más $nsubele_{max}$ partículas. En la Figura 2.10 se presenta un esquema gráfico de lo que se denomina región del nodo j. Un rango pequeño ($nsubele_{max}$-$nsubele_{min}$) produce una continua intrusión en el conjunto de partículas y un rango grande provoca sobrepoblación de las mismas. Debe establecerse un compromiso entonces entre precisión y eficiencia, que en las diversas pruebas se puede especificar en $nsubele_{max} = 3$ y $nsubele_{min} = 1$ en simulaciones de flujo incompresible. A fines prácticos, esta imposición del número de partículas se realiza al final de cada paso de tiempo.



(a) (b)

Figura 2.10: Izquierda: el nodo j y su región. Enmarcado con un trazo más grueso se presenta el subelemento del elemento k correspondiente al nodo j. Derecha: se sombrean los elementos que se quedan sin partículas luego del paso X-IVAS con CFL > 3.

2.2.3. PFEM-2 Malla Fija Partícula Fija (MFPF)

Una de las desventajas que se le puede atribuir a la estrategia MFPM es que requiere muchas partículas para que la proyección hacia los nodos resulte precisa. En la estrategia MFPF se intenta preservar la ventaja de MFPM de mantener la misma malla toda la simulación, pero se desean utilizar menos partículas y evitar así los errores por proyección y todo lo derivado en cuanto a la administración del conjunto de partículas.

Para esto, y teniendo en cuenta la buena calidad de resultados obtenidos cuando se utiliza S-2 en MFPM, se propone ubicar las partículas a tiempo t^{n+1} exactamente en la posición de los nodos de la malla de fondo y encontrar su estado integrando aguas arriba con el método X-IVAS. Esta estrategia toma el nombre del hecho de que las partículas ocupen siempre las mismas posiciones (fijas) al final de cada paso de tiempo.

Es sabido que al buscar el valor ϕ_p^n de la partícula se debe realizar una interpolación con los datos de la malla de fondo. Para mejorar los problemas de difusión introducidos por esta interpolación se propone generar una grilla más fina de partículas para utilizar en dicho cálculo. En la Figura 2.11 se presentan las diferentes posibilidades para la posición de las partículas. El parámetro N puede incrementarse tanto como uno desee, pero se debe tener un compromiso entre la precisión y el costo computacional, ya que el número de partículas crece rápidamente, y era justamente el inconveniente de MFPM que se buscaba evitar. Además, el incremento de N solo se justifica al momento de la interpolación, ya que la precisión en los resultados responde

principalmente al h de la malla, tal como cualquier cálculo que involucre resolución por elementos finitos.

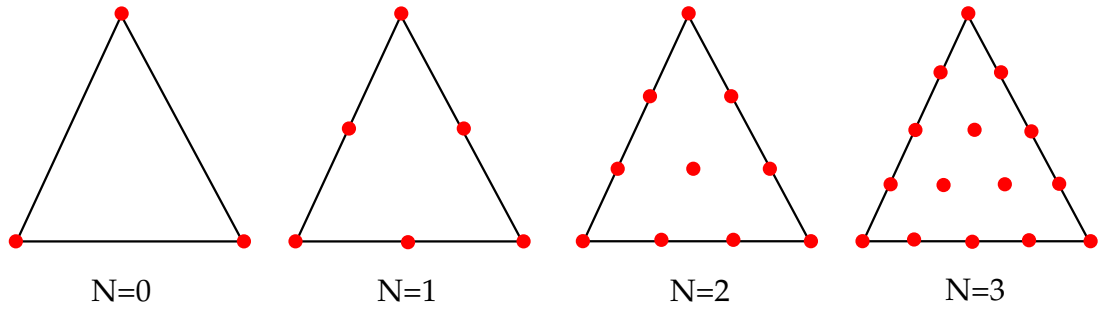


Figura 2.11: Ubicación de las partículas en un elemento dependiendo del parámetro N. Dicha posición se mantiene fija durante toda la simulación para el enfoque MFPF.

Este enfoque MFPF tiene un gran parentesco con el método presentado por Jos Stam en [13]. La estabilidad, simpleza en el cálculo y rapidez de cómputo son los puntos fuertes de estas estrategias. Ambas utilizan la integración hacia atrás siguiendo las líneas de corriente, conocida cómo *método de las características*, como estrategia de cálculo de la advección, lo que brinda gran robustez. La diferencia entre ambas radica en que MFPF permite realizar el paso difusivo de forma explícita en la propia integración de la trayectoria (aunque para garantizar independencia del paso de tiempo de deba realizarlo de forma implícita, tal como Stam), y que MFPF permite incluir una grilla de partículas que puede diferir de la grilla nodal de la malla de fondo (al seleccionar $N > 0$). Sin embargo, ambas estrategias presentan grandes problemas por la importante difusión numérica, lo que encasilla estos algoritmos dentro de una aplicación más asociada a la computación gráfica que al diseño ingenieril.

2.3. Algoritmo PFEM-2

En esta sección se integra todo lo comentado en las secciones anteriores y se presenta el algoritmo PFEM-2.

En primer lugar, en el Algoritmo 1, se exhibe la estrategia para resolver la ecuación de transporte de un escalar pasivo, convectado por un campo de velocidades conocido (Ecuación 2.17). Una gran ventaja de realizar el transporte por medio de partículas Lagrangianas es que pueden transportar diferentes campos ϕ simultáneamente, por lo que el algoritmo puede ser expandido a cuantas especies se deseen.

En segundo lugar se presenta el Algoritmo 2 para la solución de las ecuaciones de Navier-

Stokes para flujo incompresible 2.10 y 2.15. En este caso se aplica el FSM para el cálculo segregado de la presión y la velocidad. Los pasos 1 a 4 pueden agruparse dentro del paso *predictor* de la estrategia FSM, en los cuales se busca una velocidad predictora aproximada, pero que no es necesario que cumpla divergencia nula. En el paso 5 se calcula la presión por medio de una ecuación de *Poisson*, de tal manera de garantizar la conservación de masa. Por último en el paso 6 se realiza la *corrección* del campo de velocidades, al proyectarlo sobre un campo de divergencia nula.

Algoritmo 1 - Paso de tiempo en PFEM-2 para el transporte escalar.

1. Etapa de *Aceleración*: Calcular la tasa de cambio nodal:
$$\int_\Omega \boldsymbol{N}\ \mathbf{g}^n\ d\Omega \approx \int_\Omega \boldsymbol{N}\ \nabla\cdot(\alpha\nabla\phi^n)\ d\Omega = -\int_\Omega \alpha\nabla\boldsymbol{N}\cdot\nabla\phi^n\ d\Omega + \int_\Gamma \alpha\ \mathbf{N}\ \nabla\phi\cdot\boldsymbol{\eta}\ d\Gamma$$
2. Etapa *X-IVAS*: Evaluar la nueva posición y estados de las partículas:
$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_n^{n+1} \mathbf{v}^t(\mathbf{x}_p^t)\ dt$$
$$\hat{\phi}_p^{n+1} = \phi_p^n + \int_n^{n+1} (1-\theta)\mathbf{g}^n(\mathbf{x}_p^t) + \mathbf{Q}^t(\mathbf{x}_p^t)\ dt$$
3. Transferencia desde partículas a nodos:
   3.1. **MMPM**: Generar una nueva malla de fondo M^{n+1} con la posición actual de las partículas:
   $$\mathbf{x}_j^{n+1} = \mathbf{x}_p^{n+1}$$
   $$\hat{\phi}_j^{n+1} = \hat{\phi}_p^{n+1}$$
   $$M^{n+1} = \text{DELAUNAY}\left(\mathbf{x}_j^{n+1}\right)$$
   3.2. **MFPM**: Proyectar los valores desde las partículas a los nodos de la malla de fondo:
   $$\hat{\phi}_j^{n+1} = \boldsymbol{\pi}\left(\hat{\phi}_p^{n+1}\right)$$
   3.3. **MFPF**: Copiar los valores de las partículas a los nodos de la malla de fondo:
   $$\hat{\phi}_j^{n+1} = \hat{\phi}_p^{n+1}$$
4. Etapa de *Difusión Implícita*: Corrección implícita de la difusión con FEM:
$$\phi_j^{n+1} = \hat{\phi}_j^{n+1} + \Delta t\ \theta\ \mathbf{g}^{n+1}$$
$$\phi_p^{n+1}\ = \hat{\phi}_p^{n+1} + \Delta t\ \theta\ \boldsymbol{\pi}^{-1}(\mathbf{g}^{n+1})$$

Algoritmo 2 - Paso de tiempo en PFEM-2 para flujo incompresible.

1. Etapa de *Aceleración*: Calcular la aceleración nodal:
$$\int_\Omega \boldsymbol{N}\ \mathbf{a}_\tau^n\ d\Omega \approx \frac{1}{\rho}\int_\Omega \boldsymbol{N}\ \nabla\cdot\boldsymbol{\tau}^n\ d\Omega = \frac{1}{\rho}\left[-\int_\Omega \mu\nabla\boldsymbol{N}\cdot\nabla\mathbf{v}^n\ d\Omega + \int_\Gamma \mu\mathbf{N}\nabla\mathbf{v}^n\cdot\boldsymbol{\eta}\ d\Gamma\right]$$
$$\int_\Omega \boldsymbol{N}\ \mathbf{a}_p^n\ d\Omega \approx \frac{1}{\rho}\int_\Omega \boldsymbol{N}\ \nabla p^n\ d\Omega = \frac{1}{\rho}\left[-\int_\Omega \nabla\boldsymbol{N}\ p^n\ d\Omega + \int_\Gamma \boldsymbol{N}\ p^n\cdot\boldsymbol{\eta}\ d\Gamma\right]$$
$$\mathbf{a}^n = -\mathbf{a}_p^n + (1-\theta)\ \mathbf{a}_\tau^n$$

2. Etapa *X-IVAS*: Evaluar la nueva posición y velocidad de las partículas siguiendo las líneas de corriente:
$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \int_n^{n+1} \mathbf{v}^n(\mathbf{x}_p^t)\ dt$$
$$\hat{\mathbf{v}}_p^{n+1} = \mathbf{v}_p^n + \int_n^{n+1} \mathbf{a}^n(\mathbf{x}_p^t) + \mathbf{f}^t(\mathbf{x}_p^t)\ dt$$

3. Transferencia desde partículas a nodos:

   3.1. **MMPM**: Generar una nueva malla de fondo M^{n+1} con la posición actual de las partículas:
   $$\mathbf{x}_j^{n+1} = \mathbf{x}_p^{n+1}$$
   $$\hat{\mathbf{v}}_j^{n+1} = \hat{\mathbf{v}}_p^{n+1}$$
   $$M^{n+1} = \text{DELAUNAY}\left(\mathbf{x}_j^{n+1}\right)$$

   3.2. **MFPM**: Proyectar los valores desde las partículas a los nodos de la malla de fondo:
   $$\hat{\mathbf{v}}_j^{n+1} = \boldsymbol{\pi}\left(\hat{\mathbf{v}}_p^{n+1}\right)$$

   3.3. **MFPF**: Copiar los valores de las partículas a los nodos de la malla de fondo:
   $$\hat{\mathbf{v}}_j^{n+1} = \hat{\mathbf{v}}_p^{n+1}$$

4. Etapa de *Difusión implícita*: Corrección implícita de la difusión viscosa con FEM:
$$\rho\hat{\mathbf{v}}_j^{n+1} = \rho\hat{\mathbf{v}}_j^{n+1} + \Delta t\ \theta\ \mathbf{a}_\tau^{n+1}$$

5. Etapa de *Poisson*: Encontrar el valor de la presión resolviendo un sistema de Poisson con FEM:
$$\rho\nabla\cdot\hat{\mathbf{v}}_j^{n+1} = \Delta t\ \nabla\cdot\left[\nabla(\delta p^{n+1})\right]$$

6. Etapa de *Corrección*: Corregir la velocidad de la malla y de las partículas con el nuevo gradiente de presiones:
$$\rho\mathbf{v}_j^{n+1} = \rho\hat{\mathbf{v}}_j^{n+1} - \Delta t\ (\mathbf{a}_p^{n+1} - \mathbf{a}_p^n)$$
$$\rho\mathbf{v}_p^{n+1} = \rho\hat{\mathbf{v}}_p^{n+1} - \Delta t\ \boldsymbol{\pi}^{-1}(\mathbf{a}_p^{n+1} - \mathbf{a}_p^n) + \Delta t\ \theta\ \boldsymbol{\pi}^{-1}(\mathbf{a}_\tau^{n+1})$$

Capítulo 3

Implementación en Arquitecturas paralelas

El principal objetivo del método PFEM-2 es encontrar algoritmos para simular de forma precisa problemas de CFD tan rápido como sea posible. Esta metodología intenta reducir los tiempos de cálculo que hoy en día presentan tanto códigos comerciales como códigos abiertos en éste área. Esta meta no significa aún alcanzar simulaciones en tiempo real (Real Time CFD), pero es esperable cambiar días de simulación por horas o horas por minutos, haciendo posible satisfacer las actuales necesidades del diseño ingenieril. Para ello es vital contar con un código eficiente y que sea desarrollado utilizando las mejores características del software y hardware de hoy en día.

Históricamente los programas de cálculo numérico han sido desarrollados utilizando el paradigma de programación por procedimientos a través de lenguajes de programación como `Fortran` o `C`. Estos códigos se basan en estructuras de datos simples y una reducida lógica de operación, y en su ámbito de aplicación estos alcanzan una alta eficiencia computacional al optimizar tiempo de cómputo y recursos de memoria. Sin embargo, esta bien lograda optimización es demasiado dependiente del problema en particular para el que fue diseñada, y cada vez que las condiciones cambian en el programa o alguna de sus extensiones, se requiere no solo saber acerca del aspecto a ser mejorado, sino también un profundo conocimiento de todo el código. Esto último decrementa la escalabilidad del código a resolver varios tipos de problemas, sin contar la dificultades que aparecen para un grupo de desarrollo de código. Todo esto incrementa el esfuerzo inicial para obtener resultados con este paradigma [20].

Utilizando el Paradigma Orientado a Objetos (POO) a través de lenguajes como `C++`,

la información es almacenada en objetos con una estructura más compleja que es capaz de representar las abstracciones del problema y permite programar una solución por medio de una sucesión de instrucciones de alto nivel. La alta escalabilidad de este paradigma permite extender el número de problemas diferentes que se pueden resolver, preservando un costo computacional similar al paradigma anterior, ya que el núcleo del lenguaje es el mismo que en el caso procedural.

Respecto al hardware, hoy en día un procesador tipo incluye dos o más núcleos de cómputo y estos procesadores pueden agruparse de forma sencilla en clústers de varios nodos. Esto, sumado a la nueva tecnología de unidades de procesamiento gráfico de propósito general (General Purpose Graphics Processing Units - GPGPU), otorga al cómputo científico nuevas posibilidades y hacer uso de ellas es vital para obtener un código competitivo. Este trabajo se centrará en las tecnologías sobre CPUs ya que el algoritmo PFEM-2 es memoria demandante, algo que no es compatible hoy en día con los recursos GPGPU, típicamente limitados por el tamaño y la velocidad de lectura de memoria.

En el ámbito de las CPUs existen dos arquitecturas para ejecutar programas en paralelo, a saber: memoria compartida y memoria distribuida. La primera está pensada para su utilización en CPUs multi-core y las nuevas CPUs many-core, como el Intel Xeon-Phi. La segunda arquitectura esta diseñada para su uso en computadores multi-node, es decir, un conjunto de CPUs conectadas a través de una red física denominado clúster. Existe un enfoque híbrido que utiliza memoria distribuida para la comunicación entre nodos y dentro de cada nodo realiza tareas de memoria compartida para aprovechar la arquitectura multi-core del mismo.

Las siguientes secciones se organizan de la siguiente manera: En la sección 3.1 se presenta la implementación del método PFEM-2 sobre memoria compartida (SM -Shared Memory-). Partiendo de un código base serial desarrollado por el autor que implementa PFEM-2 MMPM, se lo extiende a SM y se presentan puntos claves para evitar los típicos inconvenientes de estas arquitecturas. Se implementan las tres estrategias de PFEM-2: MMPM, MFPM y MFPF y se utiliza esta plataforma para comparar entre ellas en la sección 3.1.2 y concluir en cual de ellas es la que obtiene la mejor relación entre calidad de los resultados y tiempos de cómputo.

En la sección 3.2 se detalla la implementación del método PFEM-2 MFPM sobre memoria distribuida (DM -Distributed Memory-), haciendo hincapié en las estrategias de distribución del dominio y balance de carga. Luego, en la sección 3.2.2, se presentan resultados de rendimiento sobre clústers de cómputo y comparando estos con aquellos obtenidos con otros software.

3.1. Implementación de PFEM-2 sobre memoria compartida

El modelo de ejecución de memoria compartida esta basado en el modelo bifurcación-unión (fork-join), en el cual un hilo de ejecución maestro reparte tareas a un grupo de hilos esclavos y a sí mismo (ver Figura 3.1).

Los datos en memoria son compartidos entre los hilos (de allí el nombre de la arquitectura) por lo que una modificación de algún dato por parte de un hilo influye en la próxima lectura por parte de cualquier otro hilo. Si los resultados dependen del orden de ejecución o modificación de datos, se dice que se está en presencia de una condición de carrera (race conditions). La existencia o no de race conditions depende del algoritmo a implementar. En casos positivos, es necesario analizar y utilizar estrategias de control de concurrencia para asegurar la seguridad de hilos (thread-safety) del código.

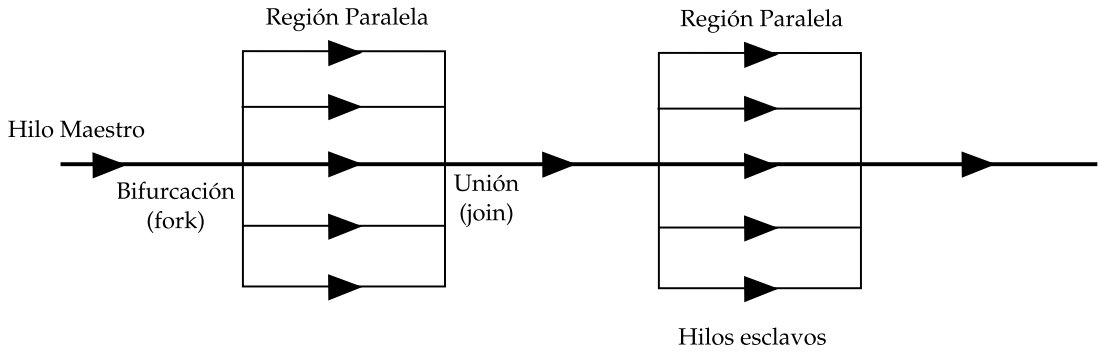


Figura 3.1: Modelo de ejecución en memoria compartida *fork-join*. Un hilo maestro (Master Thread) reparte tareas a hilos esclavos. Existe sincronización al momento de la bifurcación (fork) y unión (join). En las regiones paralelas debe prestarse especial atención en la seguridad de los hilos y evitar problemas de concurrencia.

El problema del balance de carga en memoria compartida suele ser menos severo que en memoria distribuida, pero puede deteriorar con facilidad la eficiencia, obteniendo códigos no escalables.

3.1.1. Detalles de la implementación

La implementación sobre la arquitectura de memoria compartida se realiza para las tres variantes del método. Dado que se posee un código base para el enfoque MMPM (Tesis de grado del autor [21]), se parte del mismo y se lo reorganiza para, en primer lugar, poder adaptarlo a los otros enfoques, y en segundo lugar para paralelizar las tareas. El diseño POO del código

base simplifica las tareas de extensión a las otras variantes, mientras que el algoritmo numérico, principalmente aquellas etapas explícitas, resultan compatibles y adecuadas con la paralelización de trabajo.

En la etapa X-IVAS no hay problemas de concurrencia ya que durante el bucle sobre todas las partículas, cada una de ellas actualiza su información de forma independiente. Utilizando el integrador de sub-stepping adaptativo presentado anteriormente, los tiempos de cómputo se ven reducidos ya que el algoritmo gestiona los recursos computacionales de forma óptima garantizando una correcta integración. Esto es, para un valor grande de CFL_p se calculan varios sub-pasos de tiempo para esa partícula p, lo que típicamente incluye varios cambios de elemento, y esto es algo que no sucede para las partículas con CFL_p pequeño. Aunque se reduzcan los tiempos de cómputo, utilizando este integrador aparecen problemas de balance de carga, especialmente en aquellos problemas de alta variabilidad de CFL_p. Es por esto que es necesario una distribución de partículas por hilos inteligente. Si se utiliza un esquema de distribución (o scheduling) estático, en donde se reparte el mismo número de partículas por nodos, es probable que estos problemas aparezcan. Utilizando un esquema dinámico para entregar trabajo a cada uno de los nodos puede ayudar a balancear mejor la carga.

En la presente implementación, es utilizada la especificación API para programación multi-plataforma paralela en memoria compartida llamada `openMP`. Para scheduling dinámico se selecciona la opción `schedule(guided,chunk)` (en donde *chunk* depende del número de partículas del problema) de la directiva `for`.

En la Figura 3.2 se presenta una comparación de los *speed-up* obtenidos en la etapa X-IVAS del algoritmo con las diferentes estrategias de scheduling para diferentes números de Courant en un problema de flujo incompresible. El *speed-up* S_n es calculado como $S_n = \frac{T_1}{T_n}$, donde T_1 y T_n son los tiempos de cómputo con uno y n hilos respectivamente. El lector puede observar que el desempeño de la implementación varía, en primer lugar, de acuerdo a la estrategia de scheduling elegida y, luego, por el CFL_{max} utilizado. El hecho de que en memoria compartida se puedan optar por enfoques dinámicos para la repartición de tareas, es una ventaja respecto a los enfoques de memoria distribuida, los cuales, como se verá en los próximos capítulos, exhibirán un desempeño similar al modo estático.

Respecto a la gestión de la memoria, se seleccionan contenedores secuenciales tanto para los datos elementales, nodales y de partículas, los cuales resultan óptimos para tornar más eficientes las distintas etapas del algoritmo. Por ejemplo, en el loop sobre las partículas de la etapa X-IVAS, esta disposición de los datos permite tener almacenada en memoria caché una copia

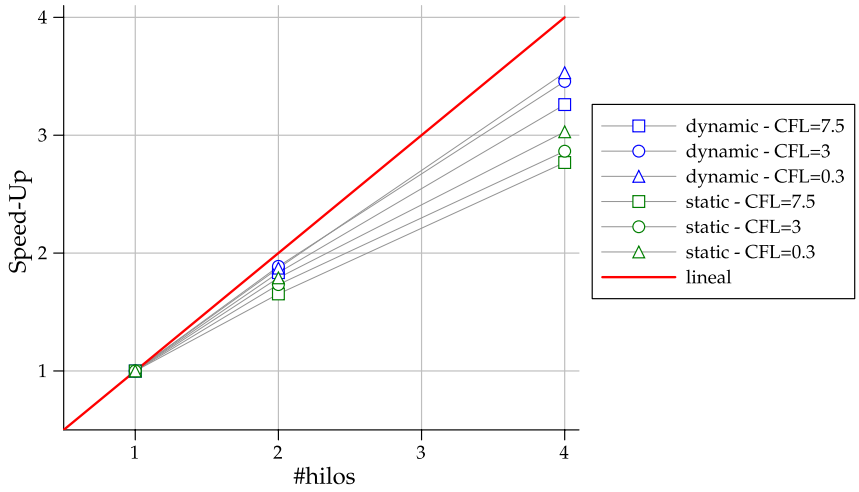


Figura 3.2: Comparación de Speed-Up obtenido para la implementación en memoria compartida de la etapa X-IVAS del Algoritmo 2 utilizando diferentes CFL.

de la partícula hasta que su movimiento finalice. Es cierto que cuando una partícula cambia de elemento deben leerse de RAM los datos del nuevo, acción que no es *cache-friendly*. Sin embargo, utilizando los contenedores secuenciales, junto a la lectura previa de los elementos vecinos, reduce la ocurrencia de *cache-misses*. Una posible solución a este problema consiste en realizar el bucle externo sobre los elementos (moviendo las partículas que contiene), pero esta alternativa no es implementada en este trabajo porque se considera ineficiente cuando se utilizan número de Courant altos, ya que esto requeriría repetir varias veces los bucles por elemento.

En el enfoque MFPM se tiene otra etapa crítica respecto a la seguridad de hilos: la etapa de proyección de los estados desde las partículas ϕ_p a los nodos ϕ_j. Cada nodo recibe información de P partículas, las cuales pertenecen a su región, y debe promediar y actualizar su estado. Por lo tanto, la función de proyección $\boldsymbol{\pi}$ puede verse como un promedio múltiple, en los cuales se utilizan diferentes factores de peso $\mathbf{W}$. En la versión implementada en memoria compartida, la función de forma N_j de cada nodo evaluada en la posición de la partícula es utilizada como factor de peso (ver Ecuación 2.30). Si se implementa esta etapa de proyección por medio de un loop sobre las partículas, podrían introducirse bloqueos mutuos entre hilos al querer actualizar el acumulador nodal. Una posible solución es utilizar estrategias de coloreo [22] que permitiría independencia en cada kernel de interpolación. Sin embargo, en el presente trabajo, tomando ventaja de la redundancia de datos de la implementación, se realiza un bucle sobre los nodos en lugar de hacerlo sobre las partículas. Esta última opción evita de forma natural los bloqueos de hilo y no introduce costo extra, ya que cada partícula solo contribuye a los nodos de su elemento, debido a la definición de región de un nodo.

Tal vez, la tarea que presenta mayor dificultad para la paralelización es el remallado en el enfoque MMPM. En el presente trabajo se utiliza la implementación de la triangulación de puntos siguiendo un algoritmo tipo Delaunay presentada por Novara en [23] y [24]. El dialogo con la librería, llamada *DeWall*, es directo ya que ésta solo requiere un conjunto de puntos (la posición de las partículas luego de remover/sembrar $\mathbf{x}_p^{n+1}$) y retorna un arreglo de conectividades que representa la nueva malla M^{n+1}.

A modo de resumen, la Figura 3.3 presenta las librerías utilizadas en la presente implementación. Para operaciones tensoriales (de rango 1 a 3 y de pequeño tamaño), típicas para el ensamblado elemental en FEM, se utiliza la librería LTensor [25]. Para la solución de los sistemas de ecuaciones lineales de utiliza la librería MKL, particularmente la aplicación paralela Pardiso [26], la cual implementa un resolvedor (solver) directo basado en la descomposición Cholesky. En aquellos enfoques en donde la matriz de los sistemas de ecuaciones puede ser reutilizada (MFPM y MFPF), se almacenan en memoria los resultados de la factorización paralela y a cada paso de tiempo solo se realiza substitución hacia atrás (también paralela, pero con baja eficiencia) para hallar las incógnitas. Por último, cabe destacar que para el pre y post-proceso se utiliza la API *Engine* del software de cálculo científico Matlab, la cual permite importar y exportar datos y ejecutar scripts en tiempo de ejecución para obtener gráficas en tiempo real de la simulación. La librería de código abierto Octave-Engine podría ser utilizada en su lugar, pero actualmente está en fase de desarrollo y no cuenta con funcionalidades como la comunicación en tiempo real de vectores de datos.

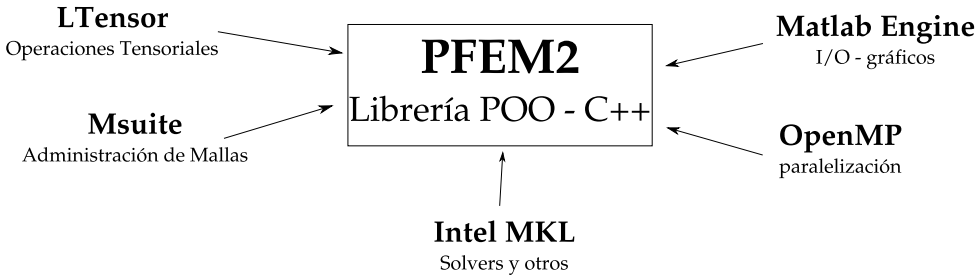


Figura 3.3: Librerías utilizadas en la implementación en memoria compartida.

3.1.2. Comparación entre los distintos enfoques

Cada uno de los enfoques del método PFEM-2 presentados en las secciones anteriores poseen sus propias ventajas y desventajas. Sin embargo, es necesario efectuar un análisis exhaustivo resolviendo un conjunto de problemas en concreto para determinar realmente las fortalezas y

debilidades de cada uno, a fin de realizar la selección de uno de ellos. De esta manera se podría enfocar los nuevos desarrollos del método, ya sea numéricos o informáticos, a la estrategia que posea mejor perspectiva a futuro.

Se analizan dos casos clásicos de flujo incompresible: la cavidad cuadrada y el flujo alrededor de un cilindro. Estos problemas poseen datos de referencia bien definidos y validados contra los cuales contrastar. El primero es una geometría sencilla con condiciones de borde también simples y se busca imitar numéricamente perfiles de velocidad reportados por Ghia [27]. Utilizando grillas gruesas es posible determinar rápidamente la difusión numérica que induce cada enfoque. En el segundo aparece una geometría un poco más compleja, en donde es necesario refinar la malla de cálculo cerca del obstáculo (cilindro) de manera tal de encontrar la evolución correcta de las fuerzas que ejerce el fluido sobre el objeto, representadas por los coeficientes adimensionales de arrastre (Drag) y de sustentación (Lift), definidos como:

$$Cd = \frac{2F_d}{\rho U^2 A} \tag{3.1}$$

$$Cl = \frac{2F_l}{\rho U^2 A} \tag{3.2}$$

donde F_d y F_l son las fuerzas paralela y perpendicular a la dirección de la velocidad U del flujo y A es un área de referencia. Además de este análisis, reviste de un gran interés el captar correctamente la frecuencia de emisión de vórtices. El refinamiento alrededor del obstáculo puede especificarse con anterioridad en los enfoques de malla fija, pero en el enfoque con malla móvil se debe realizar un sembrado de partículas más exhaustivo en la zona de interés buscando preservar el grado de refinamiento de la malla original.

Cavidad Cuadrada con Tapa Impulsada

Este clásico test posee las características que se presentan en la Figura 3.4. La simpleza de la geometría, sumado a las amplias referencias numéricas bien validadas, permiten que este ejemplo presente las primeras conclusiones acerca de cual de los enfoques PFEM-2 obtiene los mejores resultados. El Reynolds seleccionado para el caso es $Re = 1000$, con un paso de tiempo conservador de $\Delta t = 0.01$ y una malla de 50×50 nodos conformando una grilla de 2500 elementos triangulares. En la Figura 3.5 se presentan las componentes de velocidad obtenidas con cada enfoque para los cortes en los planos medios $x = 0.5$ y $y = 0.5$ a tiempo $T = 30$[s] en donde se alcanza el estado estacionario. Además se muestran gráficos que amplían los resultados sobre las zonas críticas en donde existen mayores gradientes de velocidad. Los resultados con MMPM

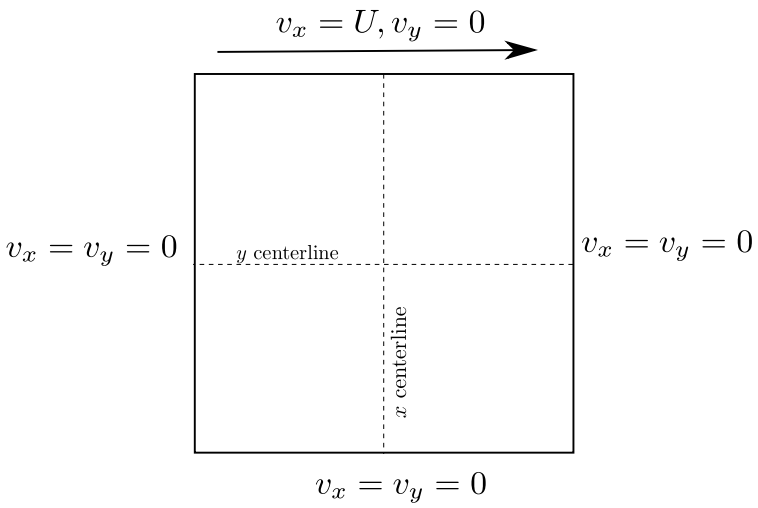


Figura 3.4: Geometría y condiciones de borde del caso de la cavidad cuadrada con la tapa impulsada.

y MFPM se destacan por sobre MFPF, y esta última solo logra no ser exageradamente difusiva si se utiliza un número alto de partículas $N >= 2$.

La Figura 3.6 muestra capturas de diferentes estados de la malla para la simulación MMPM. Partiendo desde la misma malla que con el resto de los enfoques, se remalla a cada paso de tiempo aplicando correcciones en el conjunto de partículas de tal manera de evitar la degradación de la malla resultante.

Flujo alrededor de un cilindro

La geometría de este problema, junto con las condiciones de borde e iniciales se presentan en la Figura 3.7. La malla utilizada consiste en 88000 elementos triangulares con refinamiento hacia el cilindro. Para el caso MMPM, la malla mencionada es la inicial, mientras que en los sucesivos remallados se trata de preservar, mediante sembrado de partículas, el h (tamaño de elemento) inicial de cada zona.

La Figura 3.8 presenta la evolución de los coeficientes de arrastre y sustentación para cada enfoque PFEM-2. A nivel general, todos obtienen resultados similares entre sí y similares también respecto a la simulación numérica de referencia realizada por Mittal [28]. Sin embargo, marcando las diferencias puntuales entre cada una de las soluciones, en primer lugar se percibe que los coeficientes obtenidos por MMPM presentan un aspecto más ruidoso que el del resto de los enfoques. Este ruido se debe a la continua variación de la malla y de la disposición de las partículas alrededor del obstáculo, y es un problema inherente de los métodos que

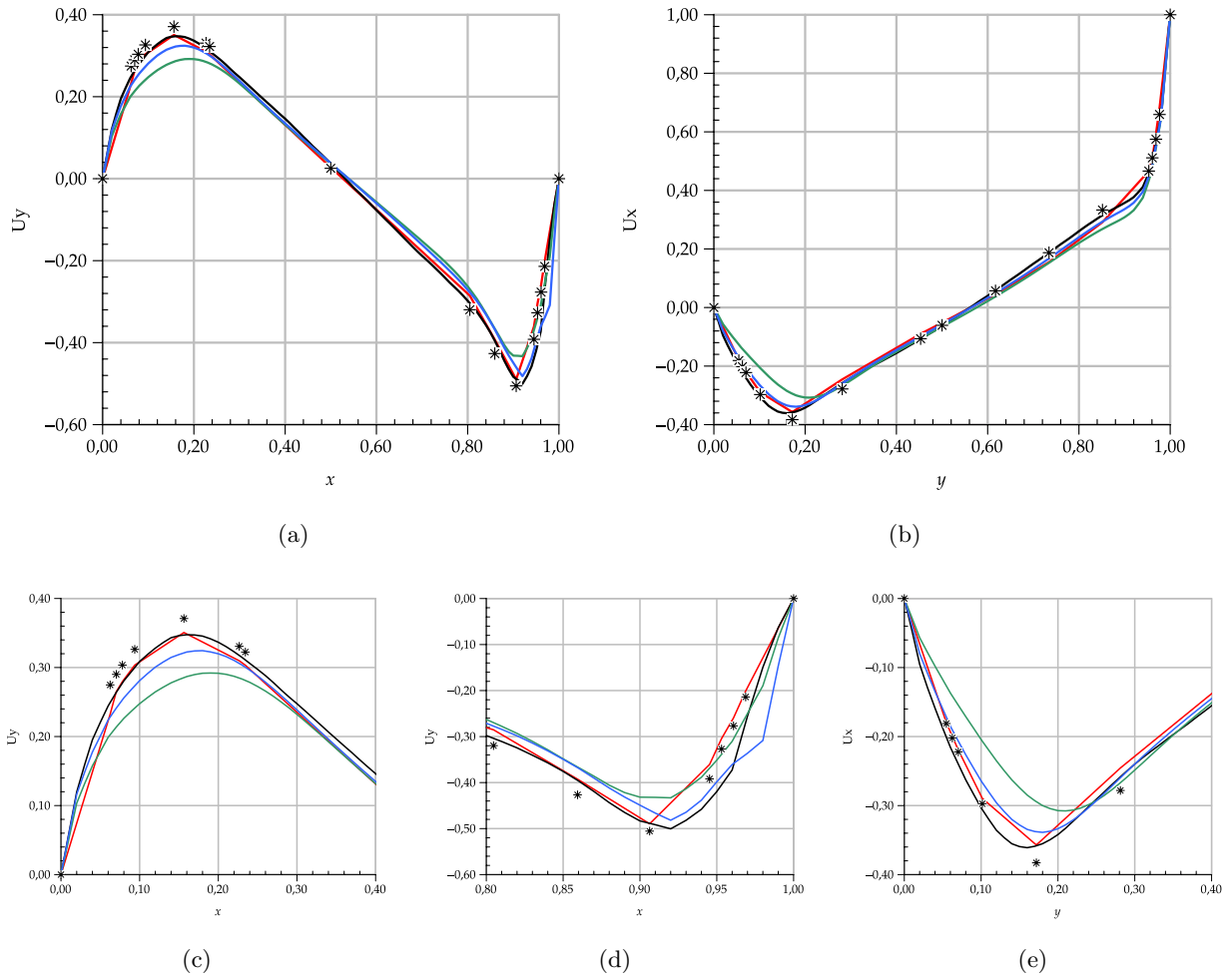
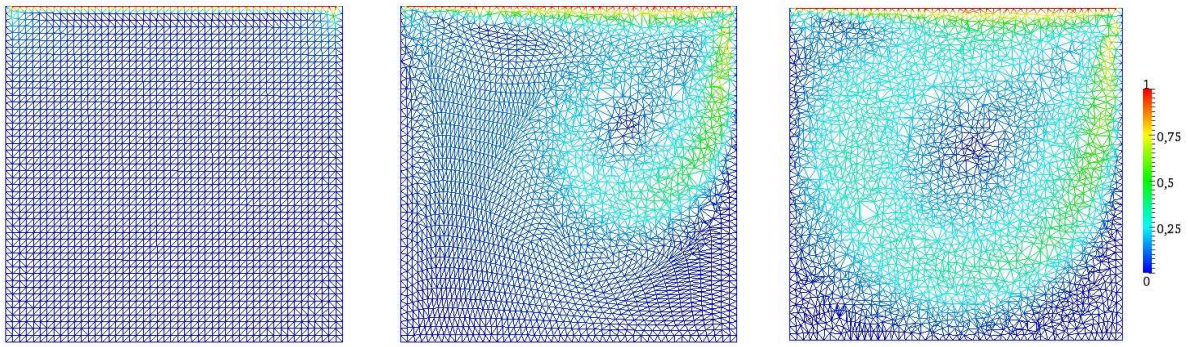


Figura 3.5: Componente y de la velocidad sobre un corte en $y = 0.5$ (Figura 3.5a) y componente x de la velocidad sobre el corte $x = 0.5$ (Figura 3.5b). Resultados para los diferentes enfoques PFEM-2. Referencias: — MMPM, — MFPM, — MFPF-1, — MFPF-3.

no utilizan una malla fija de fondo para el cálculo fuerzas. Los esquemas MFPM y MFPF presentan una evolución mucho más suave de los coeficientes, pero aparecen diferencias en su amplitudes, mostrando una vez más que MFPF posee difusión numérica exagerada. Aunque este inconveniente podría recuperarse utilizando una grilla aún más fina de partículas, los tiempos de cómputo se incrementarían demasiado, transformando el enfoque en no competitivo.

Continuando con el análisis, un valor de gran importancia en este tipo de flujos es el número adimensional de Strouhal, el cual describe los mecanismos de flujo oscilante, y obtener soluciones numéricas correctas es de vital importancia para determinar la calidad de los resultados de esta



(a) (b) (c)

Figura 3.6: Diferentes estados de la malla móvil en el enfoque MMPM. La escala de colores se corresponde a la magnitud de la velocidad. Tiempos de simulación: $0[s]$, $8[s]$ y $24[s]$.

simulación. Su valor teórico se calcula como:

$$St = \frac{fd}{U} = 0.198(1 - \frac{19.7}{Re}) \tag{3.3}$$

donde f es la frecuencia de generación de vórtices, d es una longitud característica y U es una velocidad característica. Teóricamente, con $d = 1$, $V = 1$ y $Re = 1000$, el número de Strouhal que se obtiene es $St = 0.194$. Sin embargo, Mittal explica en su trabajo [28] que en las simulaciones 2D se sobrestiman los valores de los coeficientes de arrastre y con esto la frecuencia de generación de vórtices, mientras que en una simulación 3D, en donde se captan todos los fenómenos que aparecen en este flujo, se está más cerca de las observaciones experimentales (es bien conocido por la comunidad que superando cierto Reynolds crítico de aproximadamente $Re = 300$, los efectos tridimensionales en la estela del cilindro de tornan significativos y no pueden ser omitidos). Es por esto que para comparar los resultados obtenidos en esta sección, el presente trabajo lo hace contra aquellos obtenidos numéricamente por Mittal.

La Tabla 3.1 presenta un resumen de los resultados con cada uno de los enfoques, comparando con aquellos resultados publicados por Mittal y con los obtenidos con el software OpenFOAM [29], que implementa el método de los volúmenes finitos. Respecto al Strouhal, MFPM obtiene los mejores resultados, en contraste a MFPF que, por su solución sobredisipativa, ralentiza la generación de vórtices subestimando el valor final del número de Strouhal. Los resultados de los coeficientes adimensionales obtenidos con MFPM tienen una gran similitud con los obtenidos con OpenFOAM.

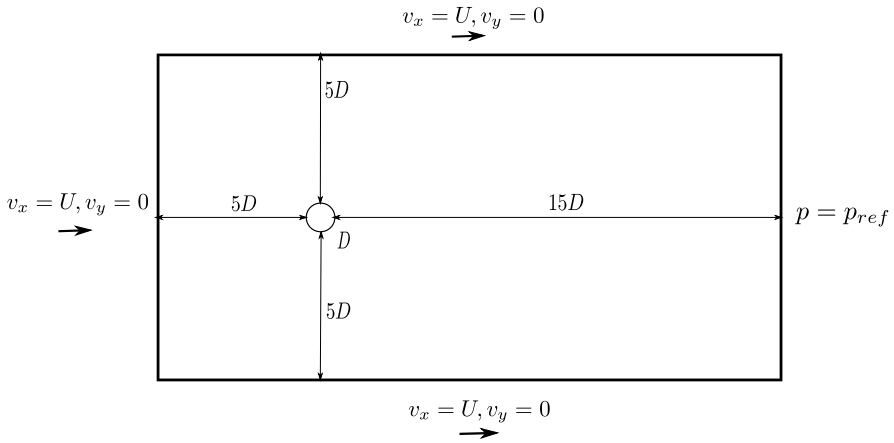
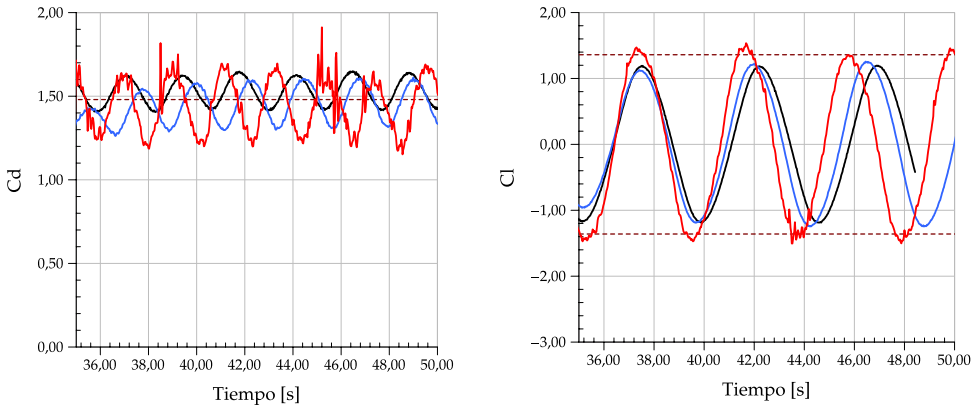


Figura 3.7: Geometría y condiciones de borde del caso del flujo alrededor de un cilindro en dos dimensiones. En el ingreso se impone un perfil plano, mientras que a la salida se imponen presión fijada a un valor de referencia y $\boldsymbol{\sigma} \cdot \mathbf{n} = 0$.

	Strouhal	$\overline{C_d}$	C_d amplitude	C_l amplitude	$T_f = 5[s]$	$S_n 4x$
Mittal	0.25	1.48	0.21	1.36	-	-
OpenFOAM	0.26	1.69	0.245	1.62	336[s]	3.01x
MMPM	0.24	1.45	0.3	1.48	195[s]	2.12x
MFPM	0.25	1.65	0.25	1.63	155[s]	2.97x
MFPF-2	0.21	1.42	0.13	1.31	119[s]	2.99x

Tabla 3.1: Resumen de las simulaciones del flujo alrededor del cilindro en 2D con los tres enfoques PFEM-2.

También se debe analizar la escalabilidad de los códigos de los diferentes enfoques. Si bien la mayoría de las etapas del algoritmo son idénticas en cada uno de ellos, la distribución del tiempo total varía, resultando más importante escalar correctamente una u otra etapa dependiendo el enfoque elegido. La Figura 3.9 presenta un esquema con los tiempos de cómputo totales, y por etapa, obtenidos por cada uno de los enfoques al simular $T_f = 5[s]$ de tiempo real en el caso del flujo alrededor del cilindro de esta sección. Los tiempos de cómputo totales son similares, mostrando el menor tiempo la estrategia MFPF-2. La etapa que mejor escalabilidad presenta es la X-IVAS, mientras que la resolución de sistemas lineales presenta el peor rendimiento. Un gran porcentaje del tiempo de cómputo en MFPM es dedicado a la integración por líneas de corriente, lo mismo que en MFPF-2, por lo que globalmente también consiguen la mejor eficiencia cuando se utilizan varios hilos de cómputo. Por otro lado, en MMPM, al poseer pocas



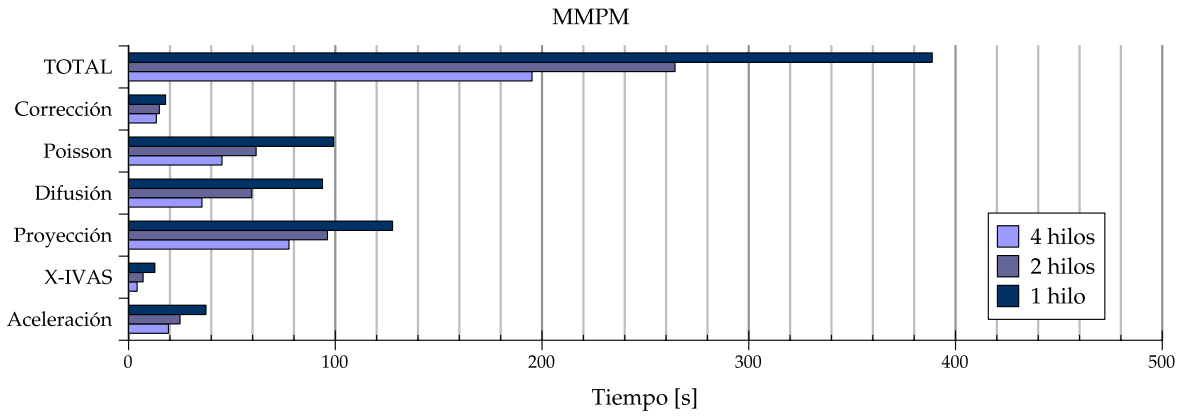
(a) (b)

Figura 3.8: Coeficientes de arrastre y de sustentación (Figuras 3.8a y 3.8b respectivamente) en una ventana temporal para los diferentes esquemas propuestos para PFEM-2. Referencias: — MMPM, — MFPM, — MFPF-3 y - - - solución numérica de Mittal.

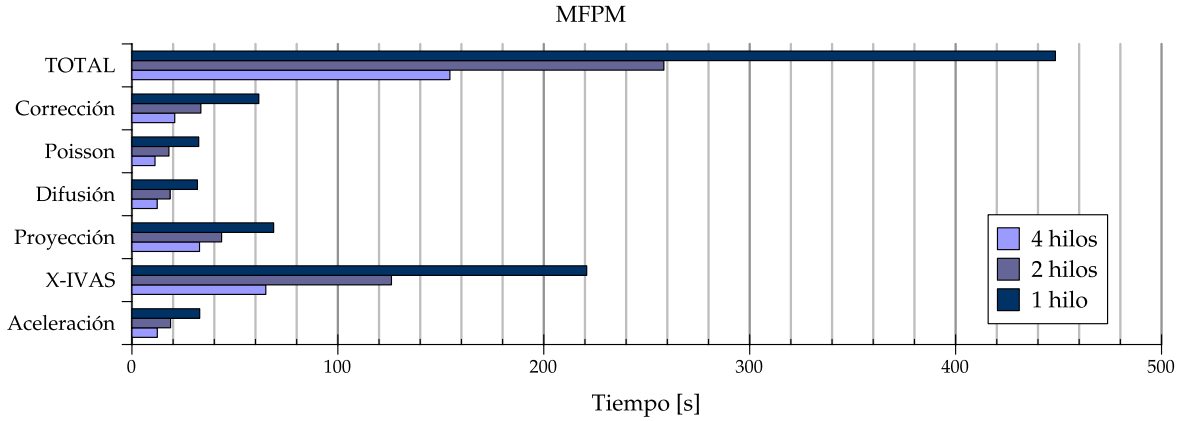
partículas (una por nodo), la etapa X-IVAS no es relevante, siendo lo más costoso de la misma los cómputos implícitos, que además de no escalar tan bien como las estrategias explícitas, tienen la particularidad en este enfoque de requerir reensamblado de las matrices de los sistemas de ecuaciones a cada paso de tiempo. Esto, sumado a que la etapa de remallado no garantiza tampoco alta eficiencia, deriva en que MMPM posea la menor eficiencia entre los tres enfoques.

Selección para implementación en Memoria Distribuida

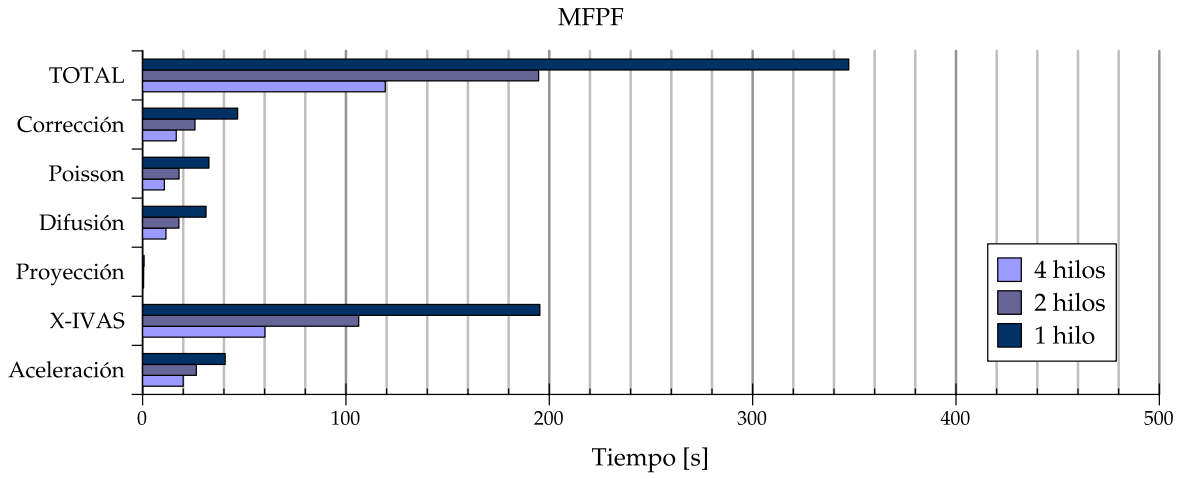
De acuerdo a los resultados presentados en las secciones anteriores, se toma la determinación de elegir el enfoque MFPM para su implementación en memoria distribuida y posterior desarrollo algorítmico en la presente tesis. El enfoque seleccionado posee las mejores tasas de speed-up ya que el esfuerzo computacional está volcado sobre las estrategias explícitas, principalmente la etapa X-IVAS. Además la necesidad de simular en clústers de cómputo con muchos nodos hace necesario garantizar alta escalabilidad, algo que no es posible con el enfoque MMPM. El rendimiento también es bueno utilizando el enfoque MFPF, pero este obtiene resultados demasiado difusivos para el interés del grupo de investigación. Para otros objetivos, tales como animaciones gráficas realistas y rápidas, esta última estrategia sería ideal, pero para la búsqueda de precisión ingenieril en problemas industriales, MFPF requiere la utilización de muchas partículas, más aún que con MFPM, lo que conlleva a que esta última opción sea la seleccionada para el resto de este trabajo.



(a)



(b)



(c)

Figura 3.9: Tiempos de cómputo para cada etapa del algoritmo PFEM-2 para los diferentes enfoques propuestos. El caso es el flujo alrededor de un cilindro en 2D. $Tf = 5[s]$, $\Delta t = 0.025$.

3.2. Implementación de PFEM-2 sobre memoria distribuida

Las librerías para elementos finitos deben almacenar en memoria varios datos acerca de la malla utilizada. Cada elemento requiere conocer su conectividades (índices o punteros de los nodos que lo componen), información de elementos vecinos y otras propiedades precalculadas. Además, los nodos deben contener su posición, estado, punteros a elementos, propiedades físicas y otros. Una rápida estimación muestra que son necesarios al menos 100 bytes por elemento y 200 bytes por nodo. Por ejemplo, para resolver un problema de flujo incompresible en 3D con una malla de 10^7 tetraedros, con aproximadamente $2.5\text{x}10^6$ nodos, una simulación típica de elementos finitos requiere al menos $1.5Gb$ de memoria RAM.

Por otro lado, las simulaciones PFEM-2 además de tener que almacenar los datos de la malla utilizada, debe tener presente en memoria los datos de las partículas (posición, estados y otros), que se estima en 100 bytes por partícula. A partir de los resultados presentados en la sección anterior, se elige implementar la versión MFPM de PFEM-2, la cual requiere aproximadamente 10 partículas por elemento en un problema 3D. Entonces, si se quiere resolver el problema presentado en el párrafo anterior, son necesarios $10Gb$ solo para almacenar la información de las partículas. Por lo tanto, cuando solo se utiliza una computadora, la restricción en términos del tamaño del problema a resolver es mucho más severa en PFEM-2 que en los solvers FEM. En consecuencia, es inmediata la necesidad de tener una implementación en memoria distribuida para poder ejecutar problemas de un tamaño importante sobre arquitecturas multi-nodo.

3.2.1. Detalles de la implementación

Librerías Base y Estrategias POO

La implementación presentada en este trabajo utiliza como base la librería `libMesh` [30]. `libMesh` es una librería paralela orientada a objetos escrita en C++ para resolver ecuaciones diferenciales mediante el método de elementos finitos. Además, incluye la posibilidad de utilizar refinamiento adaptativo (en inglés Adaptive Mesh Refinement o AMC). La comunicación entre nodos se realiza a través del estándar MPI (Message Passing Interface). Se incluyen también otras librerías, entre las cuales se pueden nombrar a `PETSc` [31], `METIS` [32] y `ParMETIS` [33]. La primera es utilizada para la resolución de los sistemas lineales, tanto en serial como en paralelo, mientras que la segunda y tercera implementan una estrategia de descomposición de dominio basada en el particionamiento de grafos, para mallas seriales y distribuidas respectivamente.

En cuanto al diseño original de `libMesh`, dos nuevas clases han sido agregadas: `Particle` y

`PFEM2`. La primera, que deriva de la clase `Point`, representa individualmente a cada partícula de la discretización, mientras que la segunda encapsula todos los datos ya que tiene dos atributos principales: la nube de partículas (un vector secuencial de tipo `Particle`) y una instancia de la clase original `EquationSystem`, que contiene los datos de la malla y los sistemas FEM a resolver.

Descomposición del dominio

Crítico para cualquier implementación paralela es una correcta elección de la metodología a utilizar para la distribución de tareas a cada proceso. En una simulación numérica, las tareas están generalmente alineadas con la integración de puntos de un cuerpo en el espacio, por lo que dividir el espacio es una solución para paralelizar el problema. Esta estrategia es adoptada en la mayoría de las implementaciones FEM a través de la metodología denominada descomposición del dominio (domain-decomposition). En esta, el dominio del problema es geométricamente dividido en subdominios que deben ser distribuidos entre los recursos computacionales disponibles. A su vez, cada subdominio intercambia datos con los otros a través de sus fronteras.

Por otro lado, la mayoría de los métodos de partículas, incluyendo el presentado en éste trabajo, poseen un paralelismo natural ya que los cálculos de fuerza y actualización de posiciones pueden realizarse de forma simultánea para todas las partículas. Desde hace años se utilizan dos ideas principales para lograr el paralelismo [34]. El primer método es denominado *descomposición atómica* de la carga de trabajo, ya que aquí el procesador computa siempre las mismas partículas sin importar en que zona del dominio se encuentren, ya que la distribución se mantiene fija. El segundo método consiste simplemente en la descomposición del dominio arriba mencionada, es decir, cada procesador actualiza las partículas que se encuentren en su zona asignada. Respecto a PFEM-2, la primera estrategia ha presentado mejor desempeño en cómputos de memoria compartida, tal como se presentó en secciones anteriores. Sin embargo, si es que se quiere imitar dicho comportamiento en memoria distribuida, el carácter global de los algoritmos empleados para la gestión de la cola de trabajos produce sobrecargas en la comunicación entre procesos, ya que es necesaria una copia de la malla en cada procesador. Por lo tanto, el segundo método, el de descomposición del dominio, es el elegido tanto para la malla como para las partículas. En la Figura 3.10 se presenta un esquema con las diferentes opciones para la repartición de tareas explicadas en éste párrafo.

La descomposición de dominio requiere cierta cantidad de comunicación entre los subdominios y/o cierto grado de duplicación de datos, para asegurar precisión en el cómputo.

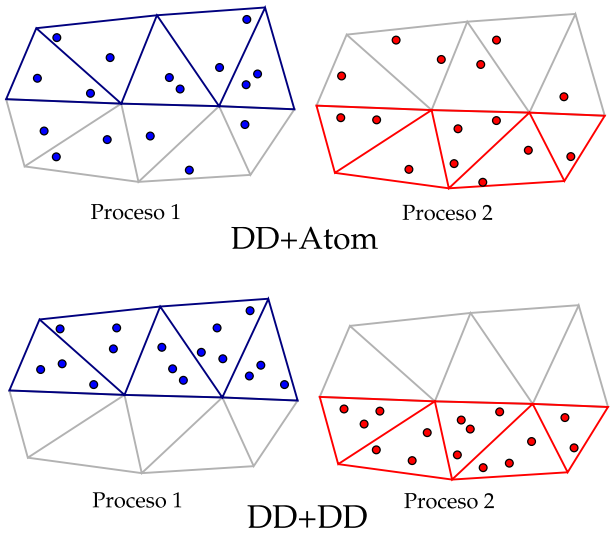


Figura 3.10: Posibilidades para la repartición de tareas en PFEM-2. DD+Atom sufre del problema de requerir una copia de la malla en cada procesador. DD+DD no requiere dicha redundancia, pero debe comunicar datos a través de las fronteras de los subdominios.

Estas zonas a lo largo de los planos de corte son llamadas *ghost* o *virtual*. Para el caso típico de una estrategia de primer orden en FEM utilizada por PFEM-2, esta zona simplemente refiere a los nodos inmediatamente adyacentes al plano de corte. Sin embargo, para calcular la trayectoria de las partículas utilizando grandes pasos de tiempo, esta zona *ghost* podría ser extendida a través de varias capas de elementos, tantos como la partícula pueda llegar a atravesar. Esta idea tiene graves problemas: existe una incerteza acerca del número de capas necesarias para garantizar la integración de la partícula en la etapa X-IVAS (además, en caso de conocerla, representaría un costo importante de comunicación para mantener los datos actualizados), por otro lado, en caso de utilizar mallas no estructuradas ya no es simple definir *capas* de elementos. Para evitar todos estos inconvenientes, existe un enfoque diferente, el cual integra la partícula hasta los límites del subdominio y en vez de continuar en los elementos ghost, el procesador responsable de la partícula la envía al procesador con los datos reales del elemento para que éste continúe con la integración (ver Figura 3.11).

Consideraciones para X-IVAS

Como se menciona en la sección anterior, el principio básico de paralelización por medio de la descomposición del dominio del algoritmo X-IVAS es que cada CPU calcule las trayectorias de su set de partículas (aquellas que residen en su memoria al inicio del paso de tiempo). Cuando

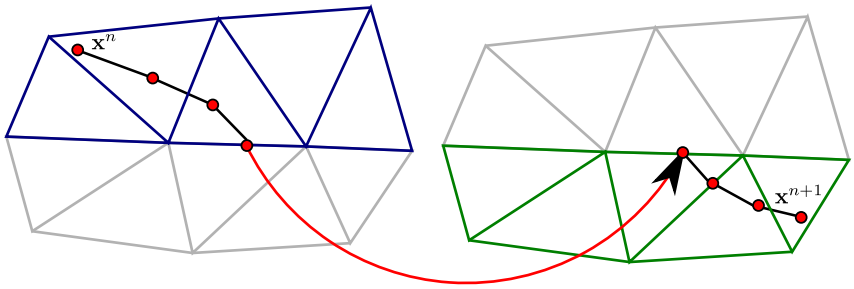


Figura 3.11: Integración de la trayectoria de una partícula con intercambio de subdominio.

una partícula cambia la partición a la que pertenece debido a su propio movimiento siguiendo las líneas de corriente, los datos de dicha partícula deben transferirse al CPU en control de la nueva partición a la cual la partícula en cuestión ha ingresado. De esta manera, el control de la partícula ahora es propiedad del CPU receptor [22].

Una opción para implementar la comunicación de partículas podría ser la transferencia asíncrona, en la cual los datos de la partícula son empaquetados en un buffer de datos continuo y enviados inmediatamente a la CPU que está en control de la partición al otro lado de la frontera entre subdominios. Cuando la partición ya ha integrado todas sus partículas, el algoritmo abandona el bucle de integración e ingresa a un bucle dummy en donde chequea si algún otro procesador le ha enviado datos que representen partículas que han ingresado a su dominio. En caso positivo, se acepta el mensaje, se desempaquetan los datos e inicia la integración para la nueva partícula adquirida. Este enfoque tiene el problema de requerir de un punto de sincronización para todas las CPUs que permita determinar el final de la integración [35]. Atendiendo a la necesidad de un punto de sincronización y buscando minimizar la comunicación, una idea mejorada realiza lo siguiente: se empaquetan las partículas a enviar, pero no se las transfiere hasta que el bucle sobre todas las partículas del dominio haya finalizado. Esta estrategia entonces permite una transferencia de partículas de forma síncrona, que también actúa como fin de integración (cuando ningún proceso recibe datos), y es por ello que es seleccionado para la presente implementación.

Para realizar las tareas de paralelización de partículas, se implementa una clase llamada `ParticleCommunication`, la cual a través del método `Alltoallv` de *MPI*, permite a cada subdominio enviar y recibir un grupo de partículas. De esta manera se opta por la estrategia más simple de implementar, sin embargo, dado que es una operación colectiva, solo obtiene buen rendimiento cuando no se utilizan gran número de procesos[36]. A futuro deben ser analizadas otras posibilidades, tratando de reducir la comunicación innecesaria entre procesadores no

vecinos, por ejemplo, intercambiando información a través de comunicaciones punto a punto con los procesadores designados por vecindad.

Dado que una partícula puede cruzar más de dos subdominios en un paso de tiempo, se necesita un loop externo cuya condición de finalización sea que la partición no tenga más partículas por mover. El Algoritmo 3 presenta una transcripción del código, en donde se observa el bucle externo, junto al bucle sobre las partículas y el criterio de parada (cuando ningún procesador tiene partículas para enviar). El bucle `for`, cuya iteración comienza desde la última partícula analizada del arreglo (con el fin de evitar repetir el cálculo), integra la trayectoria de cada partícula computando cada sub-paso de tiempo dentro de un bucle `while`. Para cada sub-paso de tiempo de la partícula existen cuatro posibilidades: el primer caso es que la partícula ha completado satisfactoriamente el sub-paso de tiempo y debe continuar con el resto de la trayectoria, en el segundo caso la partícula ha dejado el dominio global por alguna frontera geométrica, su cálculo finaliza y la partícula se desecha, en tercer lugar la partícula puede cruzar a otro sub-dominio (en cuyo caso es encolada para enviarse posteriormente); y la última posibilidad es que la partícula haya finalizado completamente el paso de tiempo.

3.2.2. Análisis de la implementación

La implementación en memoria distribuida ha sido evaluada en nuestro clúster Beowulf local en el Centro de Investigación de Métodos Computacionales (CIMEC) [37]. El clúster tiene un servidor Intel i7-2600K 8Gb RAM y seis nodos single socket con procesadores i7-3930K hexacores (16Gb RAM) conectados a través de Gigabit Ethernet. Para no introducir perturbaciones en los resultados (particularmente la reducción del speed-up observado), se desactivan de los procesadores las tecnologías Intel Turbo Boost, la cual incrementa la frecuencia de reloj de/los cores ocupados cuando el resto está en reposo, e Intel Hyperthreading, la cual procesa múltiples hilos en paralelo dentro de un único core. Finalmente se tiene un total de 36 núcleos de cómputo de 3.4 Ghz. El código fue compilado con g++ 4.7.2, y utiliza las librerías mpich2-1.4, petsc-3.3-p7 y libmesh-0.8.0.

Transporte Advectivo de una señal Gaussiana

En la sección 2.1.5 se presentó el problema de una señal Gaussiana transportada para demostrar las ventajas del esquema de integración X-IVAS como estrategia Lagrangiana en comparación de estrategias clásicas Eulerianas. En esta sección se aborda el mismo problema, pero se lo analiza desde la eficiencia computacional de la implementación en memoria distribuida.

Algoritmo 3 - Implementación del algoritmo X-IVAS sobre memoria distribuida

```
int ini_ip = 0;
while(1){
  std::map<int,std::vector<int> >particles2send;
  np = vP.size();
  for(unsigned int ip=ini_ip;ip<np;ip++){
    int c=0, pid_send;
    bool next_ddt = true;
    while(next_ddt){
      c = integrateSubStep(vP[ip],pid_send);
      switch(c){
        case 0://substep complete
          break;
       case 1://out of domain
          next_ddt = false;
          break;
       case 2://out of sub-domain
          next_ddt = false;
          particles2send[pid_send].push_back(ip);
          break;
       case 3://time-step complete
          next_ddt = false;
          break;
      }
    }
  }
  ini_ip=np;
  next = particles2send.size();
  Parallel::max(next); //max All_reduce
  if(!next) break; //no particles to send
  ParticleCommunication().interchangeParticles(particles2send,vP); //Alltoallv
}
```

Para este caso, se utiliza una malla 2D estructurada, compuesta por un millón de nodos y dos millones de elementos triangulares en los cuales se representan las velocidades y temperaturas por medio de funciones de forma lineales. Dado que la velocidad es proporcional al radio ($\mathbf{v} = \mathbf{v}(\mathbf{r})$), si se utiliza una malla no alineada con la velocidad, como sucede en este caso, se obtiene un campo no constante de CFL_h que depende de $\mathbf{r}$. Por lo tanto $\text{CFL}_h = \text{CFL}_h(r)$. Fijando $\mathbf{v}$ y δx, la única variable libre en CFL_h es Δt; por lo tanto, se eligen diferentes pasos de tiempo para obtener diferentes magnitudes locales del número de Courant.

El vector de pesos seleccionado para cada vértice del grafo de particionamiento es $\eta_n(v) = \#gdl$ (número de grados de libertad del nodo), el cual asegura una correcta división del total de elementos y partículas sobre cada procesador. La Figura 3.12 muestra un esquema de las particiones obtenidas con 1 a 16 sub-dominios. Sin embargo, como fue mencionado más arriba, esta estrategia puede introducir problemas de desbalanceo de carga en la etapa de integración por líneas de corriente, que son más evidentes cuando se utilizan grandes CFL. Debido a la independencia de CFL_h con el ángulo $\text{CFL}_h \neq \text{CFL}_h(\theta)$, la mejor opción para balancear la carga de trabajo en X-IVAS es particionar por el ángulo, pero, tal como se observa en la Figura, esto sólo ocurre con 2 y 4 procesadores. Este comportamiento es confirmado en la Figura 3.13, que muestra el speed-up S_n obtenido por la etapa X-IVAS eligiendo diferentes CFL (el máximo local CFL se reporta en la Figura). Para $\#processors <= 4$ el speed-up es prácticamente independiente de CFL_h; sin embargo cuando el número de sub-dominios se incrementa, aparecen los desbalances de carga, que empeoran la escalabilidad cuando se utilizan mayores CFL_h. Las otras etapas del algoritmo (el cálculo de la tasa de cambio y la proyección) no se presentan en el gráfico debido a que su escalabilidad no depende de CFL_h (se obtiene aproximadamente $S_{16} = 12x$ y $S_{16} = 11x$ respectivamente) y además, su relevancia en el tiempo total de cómputo es baja, alcanzando aproximadamente el 20 % del mismo.

Flujo Alrededor de un Cilindro

El flujo alrededor de un cilindro es un típico test para flujos incompresibles. La Figura 3.14 presenta la geometría utilizada para el caso tridimensional, donde $D = 1$ es el diámetro del cilindro, los límites de la geometría van desde $[-2.5D;\ -5.5D;\ -5.5D]$ a $[2.5D;\ 15.5D;\ 5.5D]$, con la dirección axial del cilindro sobre el eje x, y centrado en $[0;\ 0;\ 0]$. El caso bidimensional utiliza la misma geometría pero sin la extrusión en la dirección axial.

Respecto a las condiciones iniciales y de borde, se simula el caso para $Re = 1000$, por lo que se impone $U = [0;\ 1;\ 0]$ en la superficie de ingreso de flujo, presión fijada $p = p_{atm} = 0$ y

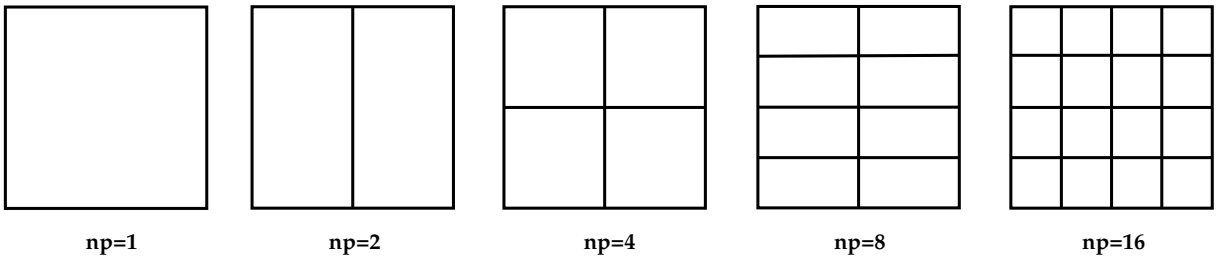


Figura 3.12: Particionamiento del dominio con 1 a 16 sub-dominios. Caso: Problema de transporte advectivo de una señal Gaussiana. La carga de trabajo es balanceada solo con 2 o 4 procesadores. Con mayor número sólo es posible balancear el número de elementos y partículas en cada procesador.

tracción nula $\boldsymbol{\sigma} \cdot \mathbf{n} = 0$ en la superficie de salida, deslizamiento libre en las superficies anterior y posterior (en el caso 3D), sin deslizamiento sobre el cilindro, y $U = [0;\ 1;\ 0]$ en las superficies superior e inferior. Los campos iniciales son $U = [0;\ 1;\ 0]$ (excepto sobre el cilindro en donde la velocidad es nula) y $p = 0$ con las propiedades del fluido constantes en todo el dominio, siendo la viscosidad $\mu = 10^{-3}$ y la densidad $\rho = 1$.

Los cómputos en 2D se realizan utilizando una malla de 88 mil elementos triangulares con 43 mil nodos y refinado hacia el cilindro. La malla tridimensional tiene 1.6 millones de tetraedros y 356 mil nodos, también refinando hacia el cilindro.

El refinamiento de la malla es diseñado con el fin de calcular con mayor precisión las fuerzas que ejerce el fluido sobre el cilindro. Dado que el número CFL depende de la inversa del tamaño del elemento h, este valor se incrementa en la vecindad del cilindro. Tal como se explicó exhaustivamente en secciones anteriores, los elementos con grandes CFL tendrán mayor carga de trabajo en la etapa X-IVAS; entonces, para balancear esta carga se debe realizar un particionamiento ponderado con un factor proporcional al CFL_h. Sin embargo, esta estrategia genera desbalance de la carga de trabajo en las restantes etapas del algoritmo, en donde el número de elementos (en los cálculos FEM) o el número de partículas (en la proyección y en la corrección) deben ser balanceados en cada procesador para optimizar el rendimiento.

La fórmula utilizada para calcular el peso de un vértice v_j del grafo de particionamiento (es decir, el peso del elemento e_j) es la misma que se utiliza para determinar el número de sub-pasos necesarios por cada partícula en la integración sobre las líneas de corriente:

$$\eta_w(v_j) = min\{N_{max}, max\{N_{min}, K \times (\mathrm{CFL}_h)_j\}\} \tag{3.4}$$

donde N_{max} y N_{min} son el número máximo y mínimo de sub-pasos requeridos para mover una

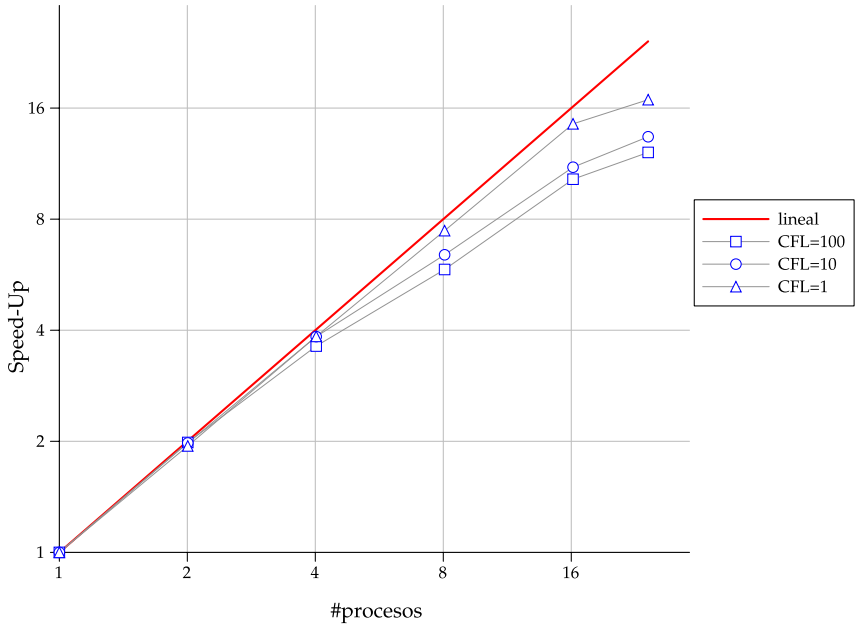


Figura 3.13: SpeedUp X-IVAS. Caso: transporte de una señal Gaussiana. En el caso explícito ($\theta = 0$) de PFEM-2 para transporte escalar, la etapa X-IVAS representa el 80 % del tiempo total de cómputo.

partícula a lo largo del total del paso de tiempo y K es un parámetro para ajustar el número mínimo de sub-pasos de tiempo requeridos para cruzar un elemento. Cabe destacar que este vector de pesos requiere de un conocimiento *a priori* de la solución, por ejemplo, realizando una corrida precursora que estime valores aproximados de CFL. Otra característica es que puede ser utilizado para particionamiento dinámico.

En la Figura 3.15 se presenta el rendimiento de la paralelización resolviendo el flujo alrededor del cilindro en 2D con $\Delta t = 0.025$ y $\text{CFL}_{max} \approx 15$. Los resultados muestran que el tipo de ponderación seleccionada para el particionamiento incide sobre el speed-up obtenido para cada etapa del algoritmo. La Tabla 3.2 presenta un resumen con los tiempos de CPU para cada etapa. Se debe notar que aunque $\eta_w(v)$ mejora la escalabilidad de la etapa X-IVAS alrededor de $4\times$ comparado con $\eta(v)$ (de $8\times$ a $12\times$), la influencia de esta etapa sobre el tiempo total no es tan relevante como en el caso escalar. De este modo, la estrategia con $\eta_n(v)$ alcanza los mejores resultados generales. Se debe observar también que el tamaño del problema no es lo suficientemente grande como para obtener buen rendimiento cuando se utilizan aproximadamente más de 10 procesadores.

En el caso tridimensional, se comparan la actual implementación del método PFEM-2 y

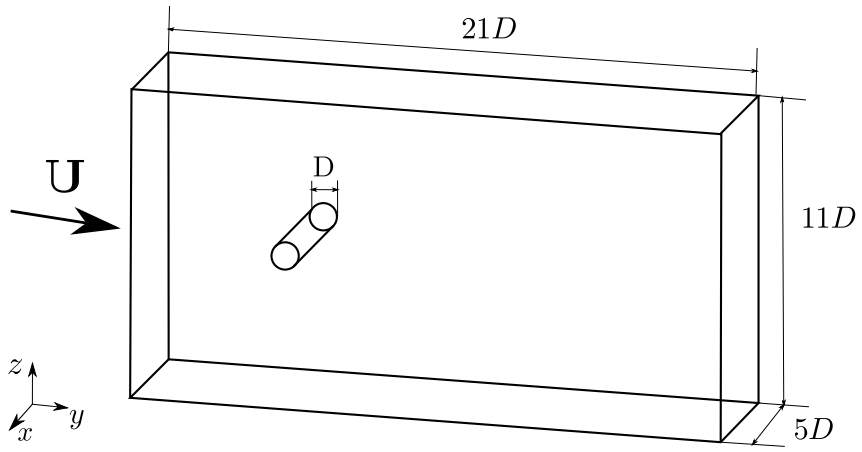
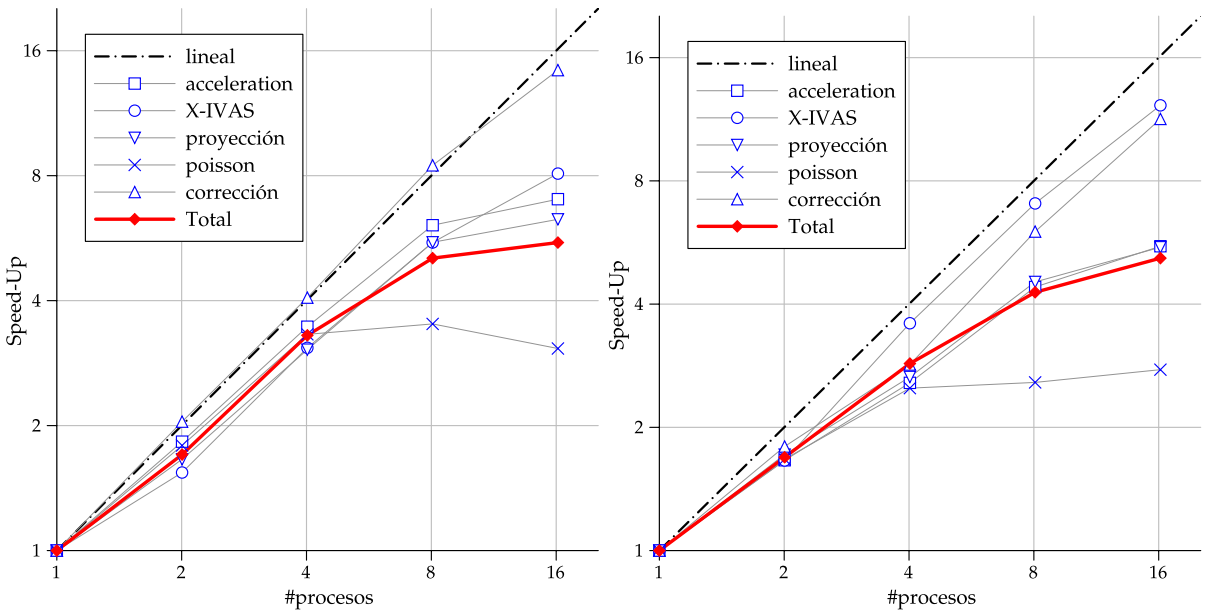


Figura 3.14: Geometría del flujo alrededor de un cilindro en tres dimensiones. La geometría bidimensional no posee la propagación axial sobre el eje x.

Etapa	1×	16× $\eta_n(v)$	16× $\eta_w(v)$
Aceleración	40.5[s]	5.81[s]	7.36[s]
X-IVAS	88.55[s]	11.02[s]	7.3[s]
Proyección	42.87[s]	6.87[s]	7.83[s]
Difusión Implícita	33.71[s]	11.58[s]	12.84[s]
Poisson	50.23[s]	16.44[s]	18.22[s]
Corrección	34.73[s]	2.44[s]	3.09[s]
TOTAL	290.5[s]	52.93[s]	56.48[s]

Tabla 3.2: Tabla de comparación de los tiempos de CPU para las diferentes etapas del algoritmo PFEM-2. Caso: flujo alrededor de un cilindro en 2D.



(a) (b)

Figura 3.15: Comparación de Speed-Up para el flujo alrededor de un cilindro en 2D. El caso de particionamiento ponderado con el número de grados de libertad $\eta_n(v)$ se observa en la Figura 3.15a. El particionamiento ponderado con la carga de trabajo de las partículas, $\eta_w(v_j)$ se presenta en la Figura 3.15b.

del ampliamente utilizado software CFD llamado OpenFOAM. Éste es un código abierto cuyo uso en ambientes ingenieriles, industriales y científicos se ha incrementado en los últimos años debido a su fiabilidad, extensión y rapidez de cómputo [38] [39]. Esta comparación servirá para determinar el estado actual del método al compararlo con una herramienta bien validada en el ambiente, cuya eficiencia es considerada de las mejores del rubro.

La idea principal de esta comparación es, utilizando la misma malla, forzar el paso de tiempo Δt a tomar el máximo valor tal que el solver permanezca estable y preciso. En los resultados de PFEM-2, se reportan los tiempos de CPU al calcular con el tratamiento implícito de la viscosidad (anulando la limitación por Fo), y para los resultados con OpenFOAM se selecciona el solver segregado `pimpleFoam`, el cual es el más rápido y robusto para flujo incompresible que ofrece la suite, ya que permite pasos de tiempo más largos que otros solvers segregados, como `icoFoam`, al incluír un bucle externo que recalcula el predictor de momento. Para resolver los sistemas de ecuaciones lineales se utiliza `pcg`, seleccionado tolerancias absolutas de 10^{-6}.

La Tabla 3.3 presenta los pasos de tiempo (Δt) que fue capaz de utilizar cada solver y muestra además los valores de CFL que cada simulación logra alcanzar. En `pimpleFoam` se

impone $max\text{CFL} = 10$ y el paso de tiempo es un promedio de los valores instantáneos de los pasos de tiempo utilizados por el solver. Ambos resolvedores son capaces de resolver la simulación con grandes pasos de tiempo, pero cuando los resultados se tornan muy difusivos (para esto se chequean los coeficientes de drag y lift) se los consideran imprecisos y dicha simulación es inválida. Por último, en la Tabla mencionada, se reporta el Speed-Up y el tiempo total de CPU para computar $1[s]$ de tiempo real, ambos con 16 procesadores. A partir de los resultados se puede concluir que, preservando una eficiencia paralela similar, PFEM-2 es aproximadamente tres veces más rápido que el más rápido resolvedor de flujo incompresible de OpenFOAM.

Solver	Re	Δt	CFL_{mean}	CFL_{max}	S_{16}	Tiempo de CPU
PFEM-2	1000	$0.05[s]$	≈ 0.75	≈ 8	10.45x	202.56[s]
OpenFOAM	1000	$\approx 0.025[s]$	≈ 0.5	≈ 10	9.41x	613.98[s]

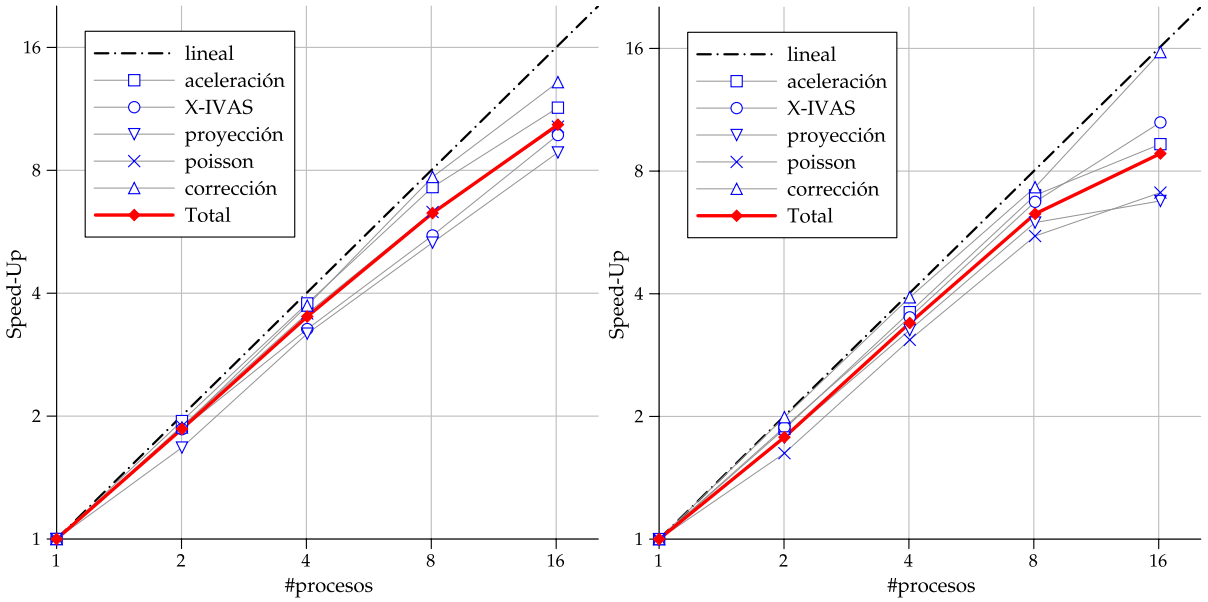
Tabla 3.3: Comparación entre PFEM-2 y OpenFOAMpara el caso del flujo alrededor de un cilindro en tres dimensiones. Ambos particionamientos del dominio utilizan $\eta_n(v)$.

El rendimiento de la paralelización se presenta en la Figura 3.16, en donde se muestra el obtenido por cada etapa del método considerando la estrategia de ponderación seleccionada. En este caso, el problema es lo suficientemente grande como para alcanzar buen speed-up con más de 10 procesos. La corrección de la velocidad por el gradiente de presiones es la etapa más eficiente debido a su simplicidad y localidad de los datos, y alcanza aproximadamente $S_{16} \approx 14x - 15x$.

La escalabilidad de las etapas implícitas resueltas con FEM, la cual es heredada de la implementación subyacente en `libMesh`, obtiene valores desde $S_{16} \approx 10x$ a $S_{16} \approx 12x$ utilizando la fórmula de ponderación $\eta_n(v)$, mientas que con $\eta_w(v)$ solo se alcanzan valores desde $S_{16} \approx 7x$ a $S_{16} \approx 9x$, indicando de esta manera un desbalance del número de elementos. El rendimiento de la etapa X-IVAS es mejorado al utilizar $\eta_w(v)$, pero esta ganancia no compensa el empeoramiento de las demás etapas del algoritmo.

Pruebas sobre un clúster Infiniband

Se debe enfatizar que una importante razón de la perdida de eficiencia al utilizar un número grande de procesadores en todos los tests, ya sea resolviendo con PFEM-2 o con OpenFOAM, es debido a la red de interconexión utilizada: Gigabit Ethernet es una arquitectura multi-propósito, que introduce severos retardos en redes congestionadas, tal como sucede cuando los nodos están computando y enviando datos. La utilización de arquitecturas dedicadas debería tener un gran



(a) (b)

Figura 3.16: Comparación del Speed-Up entre el particionamiento pesado con el número de grados de libertad $\eta_n(v)$ (Figura 3.16a) y pesado proporcional al Courant elemental $\eta_w(v)$ (Figura 3.16b). Caso: flujo alrededor de un cilindro en 3D.

impacto sobre la calidad de los resultados, por lo que en esta sección las mismas simulaciones del flujo alrededor del cilindro en tres dimensiones son re-ejecutadas pero sobre un clúster con interconexión Infiniband.

El clúster mencionado[1], llamado *Bull*, cuenta con nodos dual-socket Intel Xeon E5-2600 CPUs con 64Gb RAM, interconectados con IB-QDR 40 Gbps (las librerías utilizadas fueron las mismas que se mencionaron anteriormente). Aunque el clúster ofrece mayores prestaciones que el presentado en las anteriores secciones, el principal objetivo aquí es mostrar la dependencia de los resultados con la red de interconexión, dejando de lado para futuros trabajos el diseño y solución de problemas aún mayores que requieran utilizar toda la capacidad de este potente clúster. Para la comparación se utilizan la misma malla y configuración presentadas anteriormente. Además, la partición de dominios es realizada de la misma manera: esto es, cuatro procesos por nodo. La Figura 3.17 presenta la escalabilidad de cada etapa PFEM-2 y del algoritmo completo utilizando la fórmula de ponderación del particionamiento $\eta_n(v)$. Las diferencias son notables: con dieciséis núcleos se logra un speed-up de $S_{16} \approx 14.5x$ comparado con $S_{16} \approx 10.45x$ obtenido con el clúster Gigabit Ethernet. Mas aún, la eficiencia en el clúster Infiniband es muy buena también

[1]Cortesía del grupo de investigación dirigido por el profesor Xavier Oliver, CIMNE-UPC, Barcelona - España

corriendo con 32 procesos, alcanzando un speed-up global de $S_{32} \approx 26x$. Utilizando mayor número de procesos la eficiencia decae ya que el tiempo de comunicación se torna significante al no tener suficiente trabajo cada proceso. Pero debe notarse que este límite mínimo de número de elementos por partición disminuye gracias a utilizar la red de interconexión Infiniband.

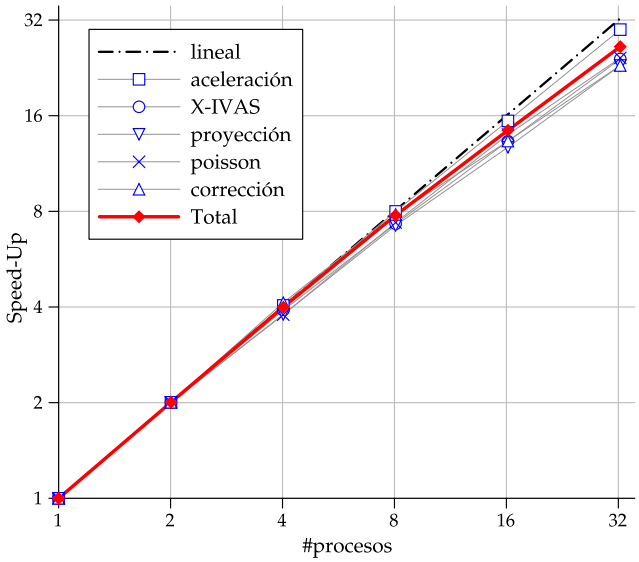


Figura 3.17: Speed-Up en un clúster Infiniband. Caso: flujo alrededor de un cilindro en 3D.

Capítulo 4

Aplicaciones del método PFEM-2

4.1. Flujos externos: perfil NACA0012

El ejemplo elegido es el popular perfil alar NACA 0012, el cual fue seleccionado para hacer uso de la gran cantidad de datos experimentales disponibles. El primer caso de estudio corresponde al perfil con un ángulo de ataque (AoA) de 0° y un número de Reynolds de 6 millones. El segundo caso es el mismo perfil pero con un $AoA = 4°$ y $Re = 10000$. En las simulaciones no se incluyen modelos de turbulencia.

La cuerda del perfil c es de una unidad de largo y el dominio computacional es lo suficientemente largo para no interferir con las condiciones de borde en el perfil. En la Figura 4.1 se presenta el dominio computacional y un detalle de la malla utilizada en ambos casos. La misma consiste en 37.568 triángulos lineales y 18.989 nodos. Se utilizan aproximadamente 200.000 partículas en promedio durante la simulación.

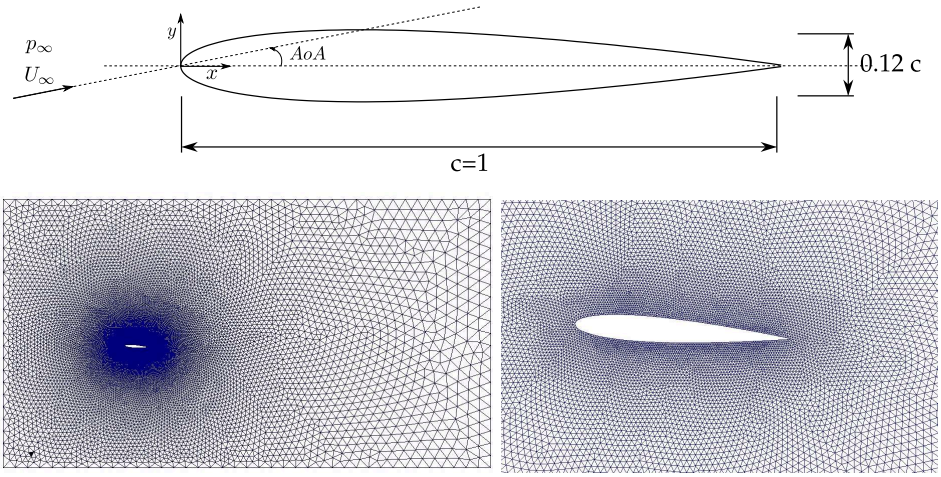


Figura 4.1: Configuración y malla utilizada para el flujo alrededor de un perfil NACA0012.

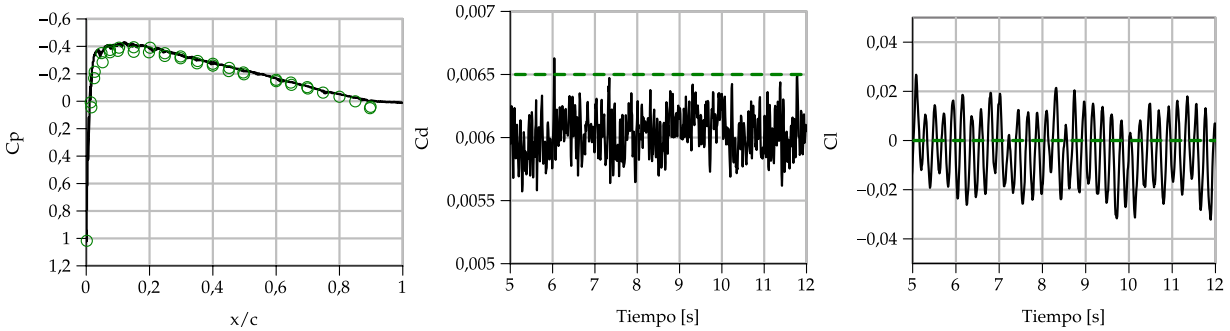
4.1.1. NACA 0012 AoA = 0° y Re=6 millones

Para tener una idea de lo que representa este valor de Reynolds, se puede tomar como pensar en un perfil alar de $D = 1$[m] viajando en aire a $T = 0°C$ a una velocidad de 160[km/h] $\approx$ 45[m/s]. Los datos de referencia fueron tomados de un reporte de experimentos realizados por Sheldhal [40]. El coeficiente de presión promedio, comparado con el valor de referencia se presenta en la Figura 4.2a. Este coeficiente es obtenido como el promedio temporal de la razón entre la presión relativa y la presión dinámica de referencia:

$$\overline{C_p} = \frac{\overline{p} - p_\infty}{\frac{1}{2} p_\infty U_\infty^2} \tag{4.1}$$

donde p_∞ y U_∞ son una presión y una velocidad de referencia lejos del perfil.

PFEM-2 logra una buena concordancia con los resultados experimentales. La curva del coeficiente de arrastre (Ecuación 3.1) se muestra en la Figura 4.2b en donde los resultados numéricos tienen una diferencia de un 8 por ciento respecto a los experimentales. El coeficiente de sustentación (Ecuación 3.2), presentado en la Figura 4.2c, oscila alrededor de cero, como también es observado experimentalmente.



(a) (b) (c)

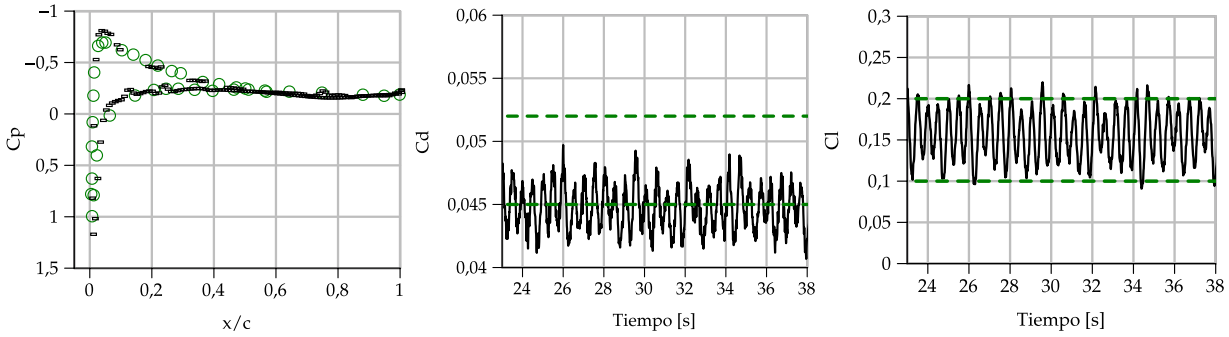
Figura 4.2: Coeficiente de presión sobre el perfil NACA 0012 (Figura 4.2a), siendo Sheldhal las marcas ∘. Evolución de los coeficientes de arrastre (Figura 4.2b) y de sustentación (Figura 4.2c). Caso NACA 0012 con $AoA = 0$ y $Re = 6e6$. Resultados PFEM-2 en líneas continuas, comparados con los datos experimentales de Sheldhal (guiones).

4.1.2. NACA 0012 AoA = 4° y Re=10 mil

Este ejemplo refuerza los resultados obtenidos en la sub-sección anterior, especialmente cuando el ángulo de ataque es diferente de cero como es normalmente esperado en modelado

aeronáutico o aplicaciones a turbinas eólicas. Los datos de referencia en este caso son tomados de las simulaciones numéricas realizadas por Srinath y Mittal [41].

Los resultados en términos de la precisión muestran que el coeficiente de presión promedio, tanto en el extradós (curva inferior) como en el intradós (curva superior), sigue los datos de referencia de forma aceptable (ver Figura 4.3a), alcanzandosé un máximo, un poco sobreestimado por la simulación, en la zona de impacto en el intradós. La amplitud y valor medio del coeficiente de sustentación coincide con los de referencia, esto es 0.15 de media y 0.05 de amplitud (Figura 4.3c). El coeficiente de arrastre es un poco subestimado: tiene una amplitud similar a la publicada en la referencia (0.07) pero su valor medio es aproximadamente un 10 por ciento menor (ver Figura 4.3b).



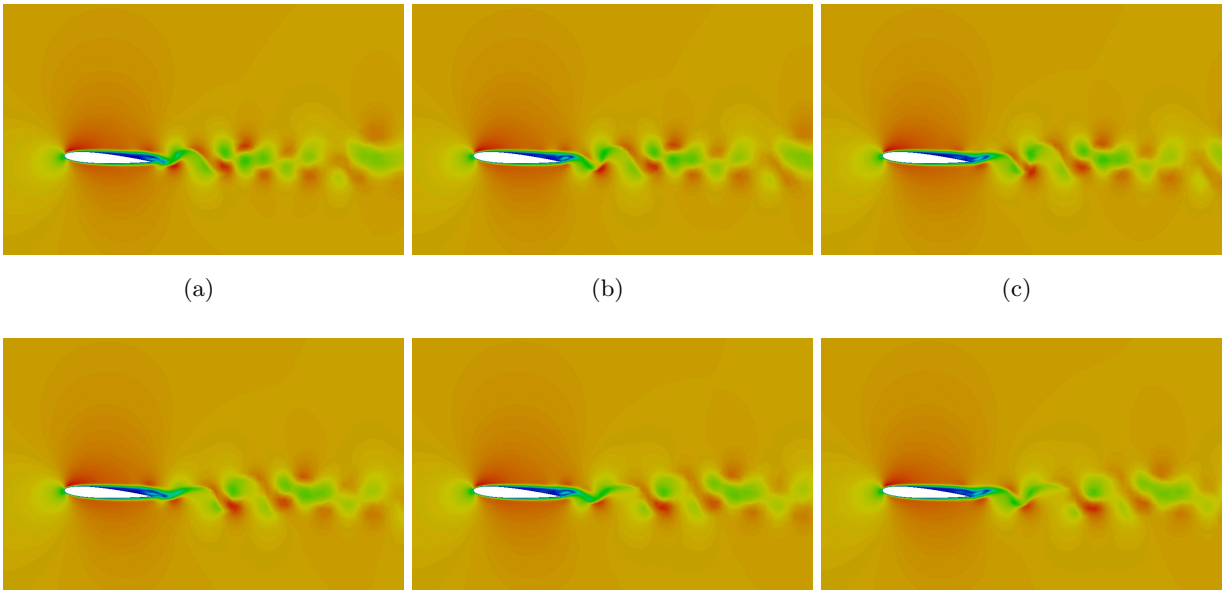
(a) (b) (c)

Figura 4.3: Coeficiente de presión sobre el perfil NACA 0012 (Figura 4.3a), siendo Mittal las marcas ∘. Evolución de los coeficientes de arrastre (Figura 4.3b) y de sustentación (Figura 4.3c). Caso NACA 0012 con $AoA = 4$ y $Re = 1e4$. Resultados PFEM-2 en líneas continuas, comparados con los datos experimentales de Mittal (guiones).

En la Figura 4.4 se presentan algunas capturas en donde se puede apreciar la generación de vórtices en la estela del perfil. La frecuencia de la generación de vórtices en la simulación con PFEM-2 es de $2.1[Hz]$, que comparada con los $2.3[Hz]$ reportados en la referencia, representa un 10 por ciento de diferencia.

Los resultados obtenidos parecen proveer la suficiente precisión para aplicaciones ingenieriles, particularmente cuando el Re no es tan alto. Para mayores Re es necesario incluir modelado de la turbulencia.

Finalmente, cabe destacar que la suite implementada y presentada en esta tesis fue utilizada en la tesis de Máster de un alumno de la Universidad Politécnica de Catalunia [42]. La misma se centró en el análisis del flujo alrededor del perfil aerodinámico NACA0012, fijando un



(a) (b) (c)

(d) (e) (f)

Figura 4.4: Capturas de la producción de vórtices para el perfil NACA 0012 con un ángulo de ataque de cuatro grados y un Reynolds de diez mil. La escala de colores representa la magnitud de la velocidad para los tiempos $T = 20$[s], 20.2[s], 20.4[s], 20.6[s], 20.8[s] y 21[s] en las Figuras 4.4a, 4.4b, 4.4c, 4.4d, 4.4e y 4.4f respectivamente.

determinado Reynolds y barriendo un rango de AoA. Los resultados presentados en dicha tesis muestran que se obtiene una buena aproximación a los valores experimentales con tiempos de simulación relativamente bajos, siendo necesario tener controlados los parámetros del modelo (principalmente el paso de tiempo y la calidad de la malla alrededor del perfil) para evitar una dispersión en los resultados.

4.2. Flujo turbulento: Cubo Montado en el Piso

Los flujos turbulentos alrededor de obstáculos tridimensionales son relativamente comunes en la naturaleza y están presentes en diversa aplicaciones como el flujo alrededor de edificios, vehículos y hasta chips de computadoras, entre muchos otros. Entender y predecir las propiedades de estos flujos es necesario por seguridad, eficiencia y economía de los diseños. Las técnicas experimentales son costosas y suelen proveer información que no es lo suficientemente detallada. Con el incremento del poder de cómputo de los últimos años, se ha tornado posible investigar este tipo de flujos a través de simulaciones numéricas.

En esta sección se presenta un reporte de la simulación del flujo turbulento alrededor de un

obstáculo cúbico. Este problema ha sido analizado experimentalmente por Martinuzzi y Tropea [43] y numéricamente por Sha y Ferziger [44] entre otros. El flujo alrededor del cubo exhibe características como la tridimensionalidad del flujo medio, separación y la no estacionariedad de las escalas grandes. Son escasos los resultados cuantitativos en este problema, por lo que solo se suelen analizar los patrones de flujo.

La geometría de este problema se presenta en la Figura 4.5.

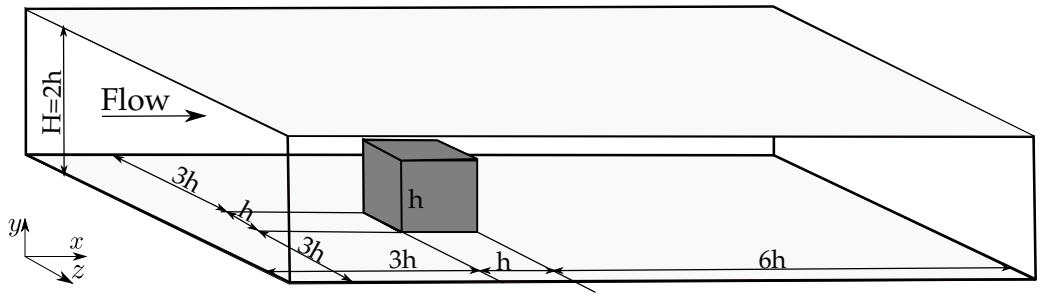


Figura 4.5: Geometría del flujo turbulento alrededor de un obstáculo cúbico

Modelado computacional

Para incluir el modelado de la turbulencia en el método PFEM-2 se opta por el enfoque de simulación de grandes vórtices (LES, por sus siglas en inglés) [45]. En LES, la malla determina la mínima escala de turbulencia que se puede resolver con el esquema numérico, el resto debe ser modelado. Al filtrar las ecuaciones de Navier-Stokes (filtrando las escalas que la malla no puede capturar), aparece un tensor $\boldsymbol{\tau}^s_{ij} = -\rho(u_i\hat{u}_j - \hat{u}_i\hat{u}_j)$, conocido como tensor de esfuerzos de sub-grilla de Reynolds (SGS, por sus siglas en inglés), el cual debe ser modelado. El modelo más simple es el denominado modelo estático de Smagorinsky, el cual define una viscosidad de remolino, sugiriendo que el tensor resultante del filtrado actúa como un término disipativo de energía. Matemáticamente se propone

$$\boldsymbol{\tau}^s_{ij} = 2\mu_T\overline{S}_{ij} \tag{4.2}$$

donde μ_T es la viscosidad de remolino y $\overline{S}_{ij}$ es el tensor de esfuerzos de las grandes escalas que se resuelven con la malla. La viscosidad de remolino puede encontrarse con

$$\mu_T = C_s^2\rho\Delta^2|\overline{S}| \tag{4.3}$$

siendo $|\overline{S}| = (\overline{S}_{ij}\overline{S}_{ij})^{1/2}$, Δ la escala de tamaño del filtro y C_s un parámetro constante del modelo estático de Smagorinsky y debe ser determinado dependiendo del problema a resolver (en

modelos de turbulencia isotrópica se utiliza $C_s \approx 0.2$). Otros modelos, denominados dinámicos, permiten variar C_s en el espacio y el tiempo.

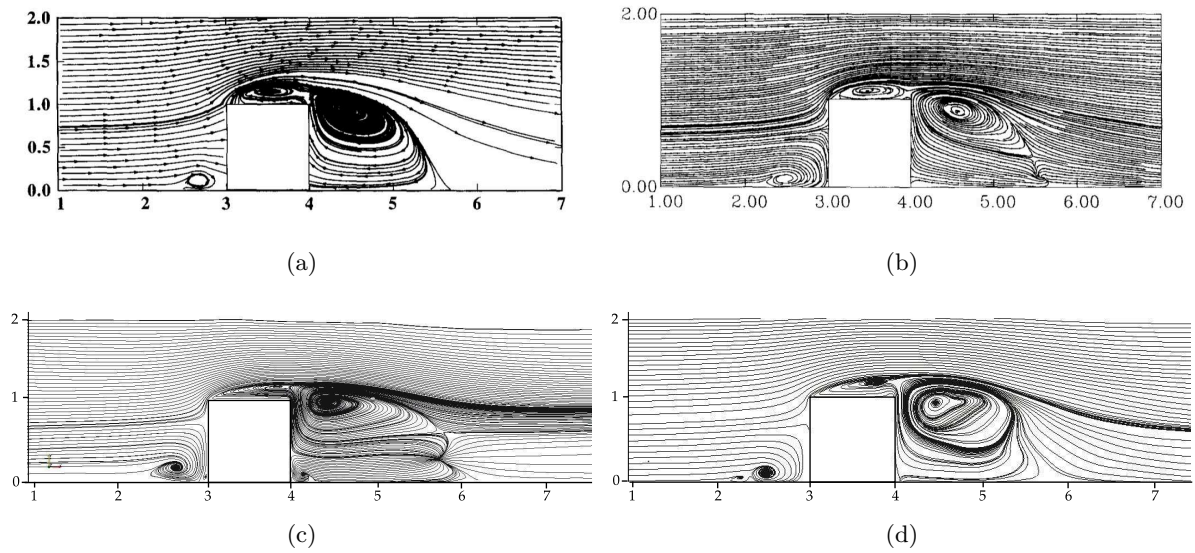
Respecto al dominio computacional, el problema fue resuelto utilizando dos grillas: la primera es relativamente gruesa y posee un millón de tetraedros refinando hacia el cubo y en su estela, con un paso de malla de $\delta_h = h/25$ sobre el cubo. Por otro lado, la segunda malla utilizada posee el mismo tipo de refinamiento pero totaliza cuatro millones de elementos, con $\delta_h = h/40$ sobre el cubo.

En las paredes anterior y posterior se impone condición de borde slip, siendo el ancho del canal $7h$, lo que asegura que los efectos de bloqueo por las paredes son despreciables. En la dirección del flujo, su utilizan condiciones inflow (ingreso de flujo) y outflow (egreso de flujo): un perfil parabólico con ciertas perturbaciones se utiliza como condición de entrada y se imponen presión fija y tracción nula $\boldsymbol{\sigma} \cdot \mathbf{n} = 0$ a la salida. La longitud del dominio es $10h$. Sobre el cubo, el piso y el techo se impone condición no-slip.

Resumen de resultados

Las simulaciones LES se realizaron para un número de Reynolds $Re = 40000$. La Figura 4.6 muestra una comparación de las líneas de corriente promediadas en el tiempo en el plano de simetría, formado por la dirección del flujo y la vertical. La predicción global de la región de separación en el techo y detrás del obstáculo es acertada aún utilizando la malla gruesa. Shah and Ferziger comentan en su trabajo que el punto de estancamiento está localizado arriba en la cara frontal, y con los resultados PFEM-2 podemos llegar a la misma conclusión. Cuando el fluido golpea el cuerpo y utilizando la malla fina, PFEM-2 logra arribar a una solución en donde el flujo se vuelve a unir en el techo, algo que la malla gruesa no puede lograr. Utilizando la malla fina, la zona de recirculación trasera no es cerrada, pero las líneas de corriente que se originan aguas arriba del obstáculo no ingresan a esta región: el fluido ingresa a la zona trasera de recirculación por los flancos. Cerca del tope de la región de recirculación se puede encontrar la cabeza del arco de vórtice. Los resultados utilizando la malla gruesa no son precisos, principalmente detrás del obstáculo.

La Figura presenta las líneas de corriente promediadas en el tiempo en un plano cerca del piso del canal. Los patrones de líneas de corriente son consistentes con los observados experimentalmente por Martinuzzi y Tropea. Estas líneas de corriente, que pueden ser vistas como líneas de fricción de corte, muestran la complejidad de este flujo tridimensional. En la referencia [44], donde se utiliza una grilla refinada hacia el cubo de $[192 \times 64 \times 96]$



(a) (b)

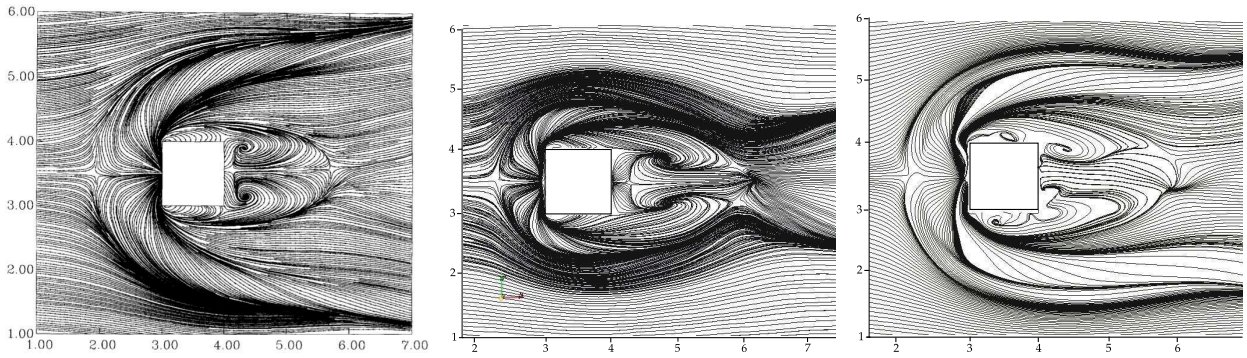
(c) (d)

Figura 4.6: Lineas de corriente en el plano de simetría para $Re = 40000$. La Figura 4.6a muestra los resultados experimentales de Martinuzzi y Tropea [43], la Figura 4.6b los resultados de la simulación LES de [44], y las Figuras 4.6c y 4.6d presenta los resultados de PFEM-2 utilizando mallas gruesa y fina respectivamente.

nodos, la separación primaria ocurre en el punto de ensilladura localizado a una distancia de alrededor de $1.05h$ adelante del obstáculo, siendo el valor experimental $1.026h$, mientras que la simulación PFEM-2 estima $0.89h$ con la malla gruesa y $0.92h$ con la malla fina. La región de separación envuelve el obstáculo y forma un fuerte vórtice de tipo herradura. La convergencia y divergencia de las líneas de corriente, que marcan el alcance de este vórtice, son fuertes zonas de upwash (corriente ascendente) y downwash (corriente descendente). Esta herradura es mejor representada por PFEM-2 utilizando la malla fina, ya que con la malla gruesa las líneas de corriente se cierran demasiado detrás del cubo. Las capturas instantáneas del flujo (que aquí no se presentan) muestran que de hecho la herradura es altamente intermitente ya que la estructura correctamente formada casi nunca aparece en estas capturas. El flujo medio en las caras laterales está completamente invertido. En Shah, el punto de unión primario se encuentra a $1.65h$ y concuerda muy bien con el valor experimental de $1.61h$, mientras que en PFEM-2, este punto se halla en $1.8h$.

En el trabajo de Shah y Ferziger [44], se dice que el punto de separación primaria delante del obstáculo y el punto de unión detrás de la zona trasera de recirculación son puntos singulares (fricción de corte nula) en donde las llamadas líneas de separación comienzan y terminan. También, se comenta que las líneas de corriente con forma de cara de búho (owl face) en la

zona trasera de recirculación se corresponden a la base del vórtice de arco, el cual está formado por una generación de vórtices cuasi-periódica que genera estructuras del tipo de calle de vórtices de Von Karman. Este arco de vórtice, que sólo existe en la solución promediada, es reproducido de forma aproximada en PFEM-2, y extrañamente se logran mejores soluciones con la malla gruesa que con la fina. Sin embargo, debe notarse que ninguna de las mallas utilizadas en este trabajo posee el refinamiento suficiente cerca del piso del canal para capturar correctamente la capa límite, lo que imposibilita obtener resultados de la misma calidad que Shah y Ferziger.



(a) (b) (c)

Figura 4.7: Lineas de corriente en un plano cerca del piso con $Re = 40000$. La Figura 4.7a muestra los resultados numéricos de Shah y Ferziger [44], y las Figuras 4.7b y 4.7c presentan los resultados de PFEM-2 utilizando mallas gruesa y fina respectivamente.

Para discutir el rendimiento de la implementación, se comentan resultados obtenidos en la simulación con la malla más fina. La misma fue llevada a cabo en el clúster Bull presentado en secciones anteriores. Utilizando una estrategia Euleriana para la ponderación del particionamiento ($\eta_n(v)$) se logra una escalabilidad de $S_{32} = 26x$, con tiempos absolutos de aproximadamente 12 minutos de cómputo por segundo de tiempo real simulado.

4.3. Problemas acoplados: Flujo Térmico

El objetivo de esta sección es presentar otra extensión del método PFEM-2, en este caso para resolver el flujo incompresible acoplado con el transporte térmico. Una de las estrategias para lograr este cometido es a través de la aproximación propuesta por Boussinesq (ver Ecuación 2.20), la cual agrega una fuerza de cuerpo de flotación a la ecuación de momento para relacionar los cambios de densidad con la temperatura. La aproximación estima que la diferencia de densidades es lo suficientemente pequeña para considerarla nula, excepto en aquellos términos en donde

aparece multiplicada por la aceleración debido a la gravedad $\rho\mathbf{g}$. La esencia de la aproximación de Boussinesq es que la diferencia de inercia es despreciable, pero la gravedad es lo suficientemente fuerte para que el peso específico del fluido sea significativamente diferente a lo largo del dominio.

En PFEM-2 se toma ventaja de que las partículas pueden transportar las variables físicas que se deseen. Además, habiendo implementado satisfactoriamente los algoritmos para flujo incompresible y para la ecuación del calor, no resulta complicado extender la estrategia a resolver el acople entre ambas, agregando el término de Boussinesq en la etapa X-IVAS.

4.3.1. Convección Natural en cavidades cerradas

Un caso particular de aplicación es el de convección natural en cavidades cerradas, en donde las diferencias de temperatura son las que inducen momento. Este tipo de problemas son de gran importancia en varios campos de ingeniería, como calderas, reactores nucleares, dispositivos de almacenamiento de energía y otros. Existen varios trabajos con resultados reportados de simulaciones de convección natural en régimen laminar, tanto en dos como en tres dimensiones. En esta sección se presentan resultados para cuatro valores del número de Rayleigh: $Ra = 10^3$, 10^4, 10^5 and 10^6, el cual se define como

$$Ra = \frac{g\beta H^3(T_h - T_c)}{\alpha\nu} \tag{4.4}$$

siendo α la difusividad térmica correspondiente al aire, con un número de Prandtl de $Pr = \nu/\alpha = 0.71$ donde ν es la viscosidad cinemática.

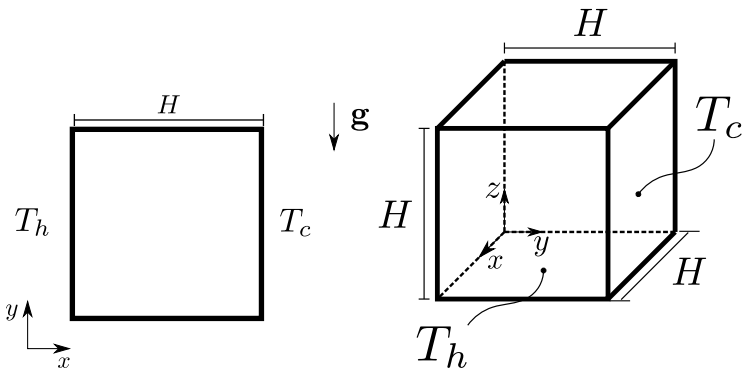


Figura 4.8: Configuración de los problemas 2D y 3D de convección natural en cavidades cerradas. Se impone una diferencia de temperaturas entre dos paredes opuestas, dejando aisladas térmicamente las restantes.

La Figura 4.8 presenta las geometrías, tanto en 2D y 3D, y las condiciones de borde de temperatura para los casos a simular. En los problemas resueltos se consideran condiciones de

borde no-slip para la velocidad en cada pared. Un par de paredes de cada geometría se encuentran a diferentes temperaturas $T_c < T < T_h$, donde T_c y T_h son las temperaturas de la paredes fría y caliente respectivamente (con $\Delta T = T_h - T_c = 1[K]$), mientras que el resto de las paredes se las aisla térmicamente ($\nabla T \cdot \mathbf{n} = 0$).

La malla de triángulos utilizada en las simulaciones 2D posee una discretización de $100x100$ nodos refinando hacia las paredes. El amplio rango de números de Rayleigh se obtiene fijando la diferencia de temperatura y demás parámetros físicos ($T_h = 0.5$[K], $T_c = -0.5$[K], $\nu = 10^{-5}$[m^2/s], $g = 1$[m/s^2]) y geométricos ($H = 1$[m]), dejando como única variable de ajuste al coeficiente de expansión térmica β, con el cual se puede determinar el Ra requerido. En el caso tridimensional, la malla utilizada tiene 81000 tetraedros y alrededor de 18000 nodos, también con refinamiento hacia las paredes.

Resultados en dos dimensiones

Las cantidades bajo análisis en esta sección son:

- $U_{max} = u_{max}(x = 0.5)/\alpha$: El máximo normalizado de la velocidad horizontal en el plano medio vertical de la cavidad (junto a su ubicación).
- $V_{max} = v_{max}(y = 0.5)/\alpha$: El máximo normalizado de la velocidad vertical en el plano medio horizontal de la cavidad (junto a su ubicación).

La Tabla 4.1 muestra los resultados con PFEM-2 para $Ra = 10^3$, 10^4 y 10^6, y se los compara con los numéricos obtenidos en [46] y en [47] quienes utilizan diferentes métodos Eulerianos (volúmenes finitos y diferencias finitas respectivamente) sobre mallas similares a la presentada en la sección anterior. A la vista de los resultados, se puede percibir una excelente concordancia con los datos obtenidos con otros métodos numéricos, lo que prueba la precisión de PFEM-2 para bajos rangos de Ra. La componente horizontal de la velocidad se presenta en la Figura 4.9. Aquí debe notarse que mientras el número de Rayleigh crece, la capa límite se torna más delgada sobre las paredes y los valores máximos de la velocidad aumentan. Por último, la Figura 4.10 presenta los perfiles de temperatura en el estado estacionario para cada uno de los valores de Ra simulados.

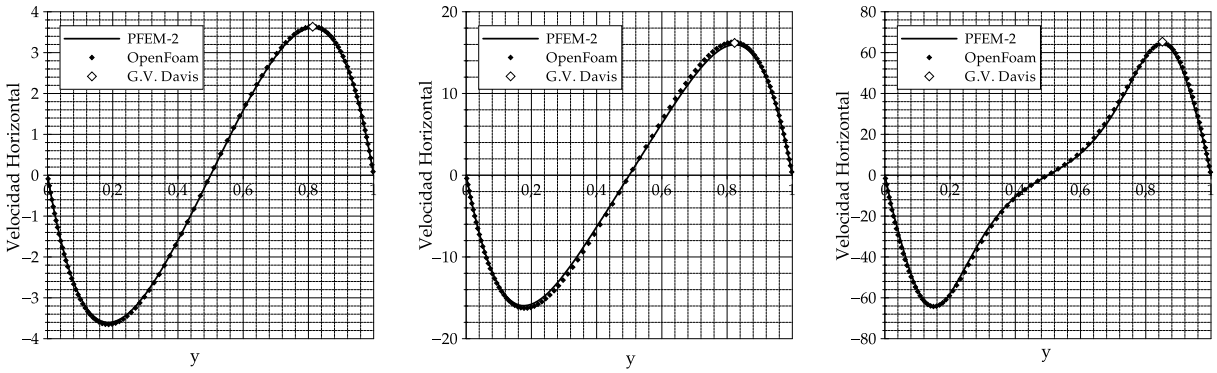
En este ejemplo se resaltan las fortalezas del método PFEM-2 al comparar los tiempos de ejecución necesarios para arribar a la solución estacionaria. La Tabla 4.2 muestra un resumen de los tiempos de cómputo requeridos para resolver $4000[s]$ de tiempo real de $Ra = 1e6$ con las estrategias de PFEM-2 utilizando esquemas de difusión explícita e implícita, y

Ra	Dato	PFEM2	Corzo	G. V. Davis
10^3	U_{max} (x=0.5)	3.605	3.640	3.634
10^3	y_max (x=0.5)	0.814	0.812	0.813
10^3	V_{max} (y=0.5)	3.650	3.700	3.679
10^3	x_max (y=0.5)	0.183	0.177	0.179
10^4	U_{max} (x=0.5)	15.982	16.281	16.182
10^4	y_max (x=0.5)	0.824	0.822	0.823
10^4	V_{max} (y=0.5)	19.378	19.547	19.509
10^4	x_max (y=0.5)	0.116	0.123	0.120
10^6	U_{max} (x=0.5)	64.483	64.558	65.330
10^6	y_max (x=0.5)	0.845	0.851	0.851
10^6	V_max (y=0.5)	218.054	221.572	216.750
10^6	x_max (y=0.5)	0.037	0.067	0.039

Tabla 4.1: Solución numérica para la cavidad térmica cuadrada con PFEM-2 comparando con los datos de referencia de Corzo y deVahl Davis.

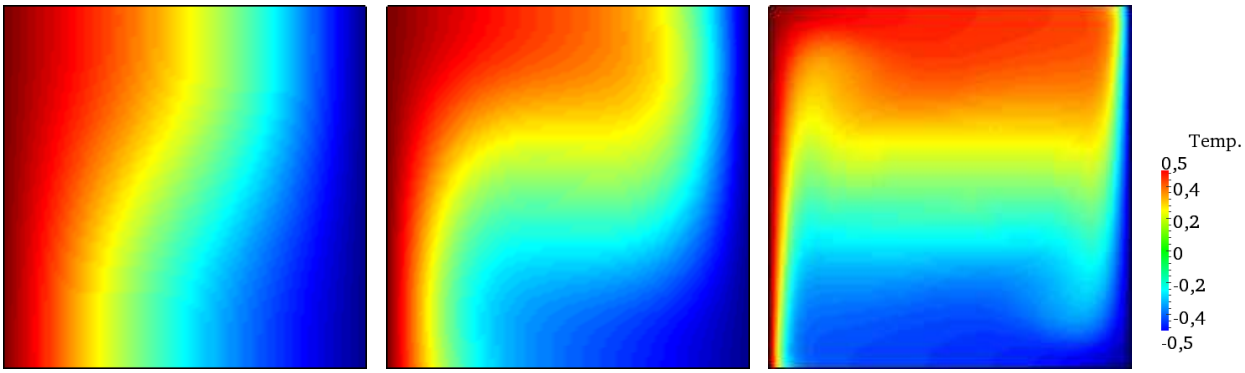
comparando con los resultados obtenidos al simular con OpenFOAM(utilizando el resolvedor `bouyantBoussinesqPimpleFoam`). Para el cómputo se utiliza un procesador Intel i7-2600k con 16Gb de RAM, corriendo con 4 procesos en paralelo.

En cada simulación se busca utilizar el máximo Δt posible, tal que la solución persista estable y precisa. La restricción por el número de Fourier limita la estrategia explícita para la difusión en PFEM-2, por lo que es necesario recurrir a un esquema implícito con el fin de alargar los pasos de tiempo. OpenFOAM no tiene esta limitación por Fourier, pero si la tiene por Courant, y por ello no es posible alcanzar los mismos pasos de tiempo que PFEM-2. Este posibilidad de alargamiento del Δt que posee PFEM-2, junto con la eficiente implementación presentada en esta tesis, permite resolver el mismo problema y con la misma precisión, alrededor de 4.7× más rápido que con OpenFOAM.



(a) (b) (c)

Figura 4.9: Perfiles de velocidad horizontal $U(x = 0.5, y)$ para 4.9a: $Ra = 10^3$, 4.9b: $Ra = 10^4$ y 4.9c: $Ra = 10^6$.



(a) (b) (c)

Figura 4.10: Campo de temperaturas en estado estacionario para : $Ra = 10^3$, : $Ra = 10^4$ y $Ra = 10^6$.

	Difusión	Δt	Tiempo de CPU
PFEM-2	Explícita	$0.5[s]$	$1127.96[s]$
PFEM-2	Implícita	$5[s]$	$148.46[s]$
OpenFOAM	-	$1[s]$	$690.88[s]$

Tabla 4.2: Tiempo total de cómputo y paso de tiempo máximo utilizado con PFEM-2 (difusión explícita e implícita) y utilizando OpenFOAM, para resolver $4000[s]$ de tiempo real de la cavidad térmica cuadrada con $Ra = 1e6$.

Resultados en tres dimensiones

Las cantidades bajo análisis en esta sección son:

- $U_{max} = u_{max}(x = 0.5, y = 0.5)/\alpha$: El máximo normalizado de la velocidad horizontal (componente x) en la línea $x = 0.5$, $y = 0.5$ y su ubicación.
- $W_{max} = w_{max}(x = 0.5, y = 0.5)/\alpha$: El máximo normalizado de la velocidad vertical (componente z) en la línea $x = 0.5$, $y = 0.5$ y su ubicación.

La Tabla 4.3 muestra los resultados PFEM-2 para $Ra = 10^4$, 10^5 y 10^6 comparados con las soluciones numéricas de Wakashima [48] (diferencias finitas de cuarto orden con avance temporal implícito) y Fusegi [49] (diferencias finitas de alto orden). La Figura 4.11 muestra los perfiles de temperatura sobre cortes en los planos medios $y = 0.5$ y $z = 0.5$ de la malla.

Ra	Dato	PFEM-2	Wakashima	Fusegi
10^4	$U_{max}(x = y = 0.5)$	0.1978	0.1989	0.2013
10^4	$z_{max}(x = y = 0.5)$	0.8460	0.8250	0.8167
10^4	$W_{max}(y = z = 0.5)$	0.2190	0.2211	0.2252
10^4	$x_{max}(y = z = 0.5)$	0.1260	0.1253	0.1167
10^5	$U_{max}(x = y = 0.5)$	0.1409	0.1423	0.1468
10^5	$z_{max}(x = y = 0.5)$	0.8460	0.8500	0.8547
10^5	$W_{max}(y = z = 0.5)$	0.2359	0.2407	0.2471
10^5	$x_{max}(y = z = 0.5)$	0.0680	0.0751	0.0647
10^6	$U_{max}(x = y = 0.5)$	0.0766	0.0813	0.0842
10^6	$z_{max}(x = y = 0.5)$	0.8570	0.8500	0.8557
10^6	$W_{max}(y = z = 0.5)$	0.2897	0.2382	0.2588
10^6	$x_{max}(y = z = 0.5)$	0.0280	0.0500	0.0331

Tabla 4.3: Solución numérica para la cavidad térmica cúbica con PFEM-2 comparando con los datos de referencia de Wakashima y Fusegi.

Los resultados muestran que PFEM-2 obtiene gran precisión también en el caso tridimensional, siendo más precisa cuando el número de Rayleigh es menor. Esto es debido a que, para números de Rayleigh mayores, debe incluirse el modelado de la turbulencia. Respecto a los pasos de tiempo utilizados, utilizando el esquema de difusión implícita se mantiene la misma ventaja respecto a lo presentado para el caso bidimensional, consecuentemente se logra una disminución

de los tiempos de cómputo totales para arribar a la solución estacionaria. Respecto a la escalabilidad del código, éste preserva la misma eficiencia que se presentó en secciones anteriores, ya que la inclusión de la nueva ecuación de transporte escalar no tiene un alto impacto sobre el rendimiento de la implementación.

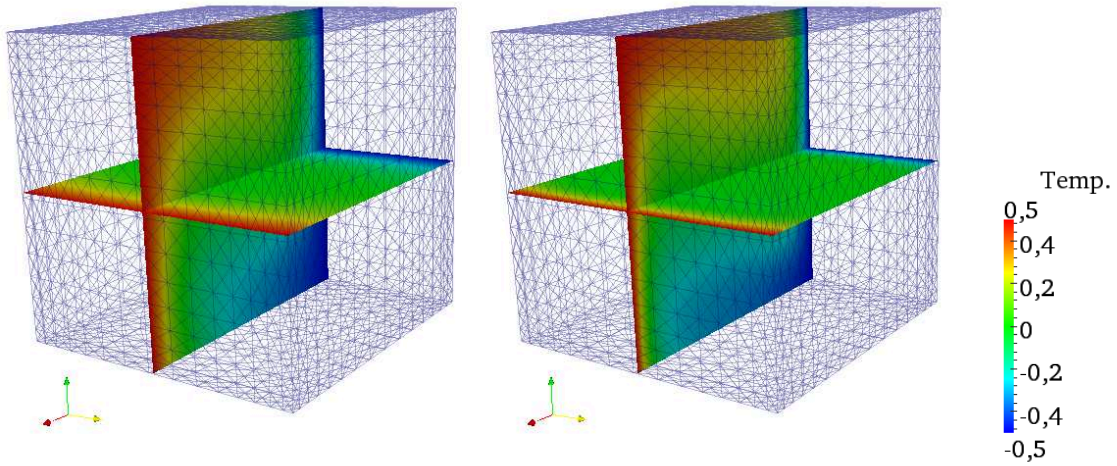


Figura 4.11: Malla tridimensional con cortes en los planos medios $y = 0.5$ y $z = 0.5$ en donde se visualizan los perfiles de temperatura en estados estacionario. Izquierda: $Ra = 10^4$, derecha: $Ra = 10^6$.

Capítulo 5

Conclusiones

5.1. Conclusiones del Trabajo

Esta tesis presenta una revisión y los últimos desarrollos de la metodología PFEM-2 para la solución de problemas de transporte, ya sea escalares o vectoriales. La novedosa estrategia de implementación explícita X-IVAS es desarrollada en profundidad debido a su fuerte influencia en la capacidad de extender los pasos de tiempo, que hace fuerte al método PFEM-2 en comparación a otras estrategias.

En los últimos años se ha dedicado mucho esfuerzo a la mejora del rendimiento del método con el fin de hacerlo competitivo con el resto de estrategias y programas utilizados tradicionalmente en la mecánica computacional. En el presente trabajo se presentaron dos implementaciones, originadas por esta tesis, que hacen uso de arquitecturas paralelas de memoria compartida y de memoria distribuida. En la Tabla 5.1 se presenta un resumen con las características a resaltar de cada una.

La primer implementación, sobre memoria compartida, es utilizada principalmente para determinar cual de los enfoques de comunicación nodo-partícula propuestos en la sección 2.2 presenta las mejores características de acuerdo a criterios de eficiencia computacional y precisión numérica. En este marco, es seleccionada la estrategia Malla Fija Partícula Móvil (MFPM) para su implementación en memoria distribuida. En la implementación sobre esta arquitectura se realizan análisis de escalabilidad y eficiencia de acuerdo a la estrategia de particionamiento seleccionada. En simulaciones donde la etapa X-IVAS domina el cómputo (casos de transporte escalar y de flujo incompresible a bajo número de Fourier en los que no se requiere corrección implícita de la difusión), utilizando un vector de pesos con valores proporcionales al CFL elemental se mejora la escalabilidad de la integración por líneas de corriente, y por lo tanto,

Características	**Memoria Compartida**	**Memoria Distribuida**
Enfoques PFEM-2	MMPM - MFPM - MFPF	MFPM
Distribución de la Malla	-	descomposición del dominio ponderada (Metis)
Distribución de Partículas	dinámica	estática
$Ax = b$ Solvers	Directo (Pardiso)	Iterativo Krylov (PETSc)
$Ax = b$ Precondicionadores	-	PETSc (permite user-defined)
100K elementos - 1M partículas	1.4 GB	1 GB
Máx problema resuelto	1M elementos - 10M partículas	6M elementos - 60M partículas
PFEM-2 Máx Speed-Up	4.3x en 6 procesadores	26x en 32 procesadores
X-IVAS Max Speed-Up	4.9x en 6 procesadores	29x en 32 procesadores
Resolviendo un mismo problema	100 pasos en 3.25[s]	100 pasos en 3.28[s]

Tabla 5.1: Comparación entre las dos implementaciones del método PFEM-2 presentadas en esta tesis.

de la simulación en su conjunto. En problemas de flujo incompresible con mayores números de Fourier, en donde el tiempo total de cómputo es controlado por las etapas implícitas, si se desea incrementar el rendimiento, el vector de pesos debe contener valores proporcionales al número de grados de libertad de cada elemento. Más allá de la ponderación utilizada, otra gran mejora en la eficiencia puede obtenerse si se utilizan arquitecturas dedicadas de interconexión, tales como Infiniband.

Con el fin de demostrar las virtudes del método para resolver una gama más amplia de problemas, se incluyó en las últimas secciones la resolución de casos que requieren modelado de la turbulencia y problemas de acople térmico entre otros. Tanto en estos ejemplos, como en cada uno de los desarrollados en las secciones previas, se presentan también resultados obtenidos con la suite OpenFOAM. De esto se concluye que, en la mayoría de los casos, el método PFEM-2, gracias a su capacidad de utilizar grandes pasos de tiempo, logra obtener rendimientos superiores al mencionado software, ubicándose hoy en día entre los métodos más rápidos para la simulación de fluidos homogéneos.

5.2. Trabajos Futuros

El método PFEM-2 se encuentra en constante desarrollo y existen diversas líneas de trabajo para continuar con lo presentado en la presente tesis, entre ellos se pueden listar:

- En el presente trabajo no se ha realizado un análisis profundo del rendimiento del código en

la solución de los sistemas de ecuaciones. Debe realizarse un análisis de escalabilidad débil del código, haciendo hincapié en la selección de precondicionadores y métodos iterativos para la solución de los sistemas lineales.

- Análisis del error numérico que se comete al utilizar formulaciones Lagrangianas en contraposición de utilizar formulaciones Eulerianas, demostrando de forma teórica que la extensión de los pasos de tiempo perjudica menos los primeros que a los segundos.

- Realizar simulaciones con problemas más grandes, tendiendo a probar las respuestas de la presente implementación al momento de tratar situaciones masivamente paralelas. Para esto es necesario rediseñar la estrategia de generación de mallas, para poder crearlas y refinarlas de forma distribuida. Además en necesario reimplementar la comunicación de partículas entre dominios, pasando desde la actual operación colectiva a operaciones punto a punto entre procesadores vecinos.

- Existe un gran interés en el grupo de investigación en extender el método a problemas bifásicos y de superficie libre. Dotar a PFEM-2 de la capacidad de resolver este tipo de problemas permitirá al método resolver una gama más amplia de problemas, principalmente enfocado a casos industriales que CIMEC resuelve a diario. Preservar o ampliar las ventajas de PFEM-2 sobre otras estrategias es uno de los objetivos. En esta línea se han comenzado desarrollos durante el último tiempo y es el tema elegido por el autor para la continuación de su trabajo, que deberá ser plasmado en la futura tesis de doctorado.

Bibliografía

[1] J. Donea and A. Huerta. *Finite Element Method for Flow Problems*. Wiley, Chichester England, 1983.

[2] R. A. Gingold and J. J. Monaghan. Smoothed Particle Hydrodynamics, theory and application to non-spherical stars. *Royal Astronomical Society*, 181:375–389, 1977.

[3] J.J. Monaghan. An introduction to SPH. *Computational Physics Communications*, 48:89–96, 1988.

[4] H. Tamako S. Koshizuka and Y. Oka. A particle method for incompressible viscous flow with fluid fragmentation. *Computational Fluid Mechanics Journal*, 113:134–147, 1995.

[5] S.R. Idelsohn, E. Oñate, N. Calvo, and F. Del Pin. The meshless finite element method. *Int. J. Num. Meth. Engng.*, 58,6:893–912, 2003.

[6] S.R. Idelsohn, N. Calvo, and E. Oñate. Polyhedrization of an arbitrary 3D point set. *Computer Method in Applied Mechanics and Engineering*, 2003.

[7] S.R. Idelsohn, E. Oñate, and F. Del Pin. 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.

[8] S.R. Idelsohn, E. Oñate, F. Del Pin, and N. Calvo. Lagrangian formulation: the only way to solve some free-surface fluid mechanics problems. In *Fifth World Congress on Computational Mechanics*, pages 1459–1475, 2002.

[9] A. Larese, R. Rossi, E. Oñate, and S.R. Idelsohn. Validation of the particle finite element method (PFEM) for simulation of the free-surface flows. *Engineering Computations*, 25(4):385–425, 2008.

[10] S.R. Idelsohn, E. Oñate, F. Del Pin, and N. Calvo. Fluid–structure interaction using the particle finite element method. *Comput. Meth. Appl. Mech. Engrg.*, 195:2100–2113, 2006.

[11] S.R. Idelsohn, M. Mier-Torrecilla, and E. Oñate. Multi-fluid flows with the Particle Finite Element Method. *Comput. Meth. Appl. Mech. Engrg.*, 198:2750–2767, 2009.

[12] Z. Wieckowsky. The material point method in large strain engineering problems. *Computer methods in applied mechanics and engineering*, 193.39:4417–4438, 2004.

[13] J. Stam. Stable Fluids. *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series:121–128, 1999.

[14] J. Stam. Stable Fluids. Document Unpublished, 2010.

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

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

[17] V. Alexiades, G. Amiez, and P.A. Gremaud. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Communications in Numerical Methods in Engineering*, 12:31 – 42, 1996.

[18] G. K. Despotis and S. Tsangaris. Fractional step method for solution of incompressible Navier-Stokes equations on unstructured triangular meshes. *International Journal for Numerical Methods in Fluids*, 20(11):1273–1288, 1995.

[19] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Adv. in Comput. Math.*, 389:389–396, 1995.

[20] M.A. Bretones, A.R. Ferran, and A. Huerta. La programación orientada al objeto aplicada al cálculo por elementos finitos. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingenieria*, 11:423–449, 1995.

[21] J. M. Gimenez. Desarrollo e implementación del método PFEM-2. *Degree Thesis - Facultad de Ingeniería y Ciencias Hídricas, Universidad Nacional del Litoral*, 2011.

[22] I.F. Sbalzarini, J.H. Walther, M. Bergdorf, S.E. Hieber, E.M. Kotsalis, and P. Koumoutsakos. PPM – A highly efficient parallel particle–mesh library for the simulation of continuum systems. *Journal of Computational Physics*, 215(2):566 – 588, 2006.

[23] P.J. Novara and N.A. Calvo. Implementación de un método Paralelo de triangulación Delaunay euclídeo. In *Mecánica Computacional Vol XXX*, pages 1933–1944.

[24] P.J. Novara and N.A. Calvo. Generación de mallas de tetraedros Delaunay en paralelo a partir de una nube de puntos y una frontera impuesta. In *Mecánica Computacional Vol XXXI*, pages 3075–3084.

[25] A.C. Limache and P.S. Rojas Fredini. LTensor: A high performance Tensor Library based on Index Notation. `http://code.google.com/p/ltensor/`, 2010.

[26] Intel MKL. Intel Math Kernel Library, Linear Solvers Basics. Document Number: 308659-001, 2005.

[27] U. Ghia, K. Ghia, and C. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

[28] S. Mittal and V. Kumar. Flow-Induced vibrations a of a light circular cylinder at Reynolds numbers 1000 to 10000. *Journal of sound and vibration*, 245(5):923–946, 2001.

[29] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998.

[30] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4):237–254, 2006.

[31] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. Curfman McInnes, B. Smith, and H. Zhang. PETSc Web page, 2012. http://www.mcs.anl.gov/petsc.

[32] G. Karypis and V. Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359–392, 1999.

[33] G. Karypis and V. Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix reordering. *Parallel and Distributed Computing*, 48:71–95, 1998.

[34] R. Kacianauskas, A. Maknickas, A. Kaceniauskas, D. Markauskas, and R. Balevicius. Parallel discrete element simulation of poly-dispersed granular material. *Advances in Engineering Software*, 41:52–63, 2010.

[35] B. Kaludercic. Parallelisation of the Lagrangian model in a mixed Eulerian-Lagrangian CFD algorithm. *Parallel Distrib. Comput.*, 64:277–284, 2004.

[36] Torsten Hoefler and Jesper Larsson Traff. Sparse collective operations for mpi. In *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, IPDPS '09, pages 1–8, Washington, DC, USA, 2009. IEEE Computer Society.

[37] Centro de investigaciones en mecánica computacional, 2013. http://www.cimec.org.ar.

[38] E. Jarvinen. OpenFOAM performance on Cray XT4/XT5. Technical report, CSC - IT Center for Science Ltd. , Espoo, Finland, 2009.

[39] M. Culpo. Current Bottlenecks in the Scalability of OpenFOAM on Massively Parallel Clusters. Technical report, Partnership for Advanced Computing in Europe, 2012.

[40] R.E. Shedhal and P.C. Klimas. 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, 1981.

[41] D.N. Srinath and S. Mittal. Optimal aerodynamic design of airfoils in unsteady viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 199:1976–1991, 2010.

[42] L. García Serrano. *Análisis de perfil aerodinámico con PFEM-2*. PhD thesis, Master en Métodos Numéricos para Cálculo y Diseño en Ingeniería, Universidad Politécnica de Catalunia, 2014.

[43] R. Martinuzzi and C. Tropea. The flow around surface-mounted, prismatic obstacles placed in a fully developed channel flow. *Journal of Fluids Engineering*, 115:85 – 92, 1993.

[44] K. B. Shah and J. H. Ferziger. A fluid mechanicians view of wind engineering:large eddy simulation of flow past a cubic obstacle. *Journal of Wind Engineering and Industrial Aerodynamic*, 67&68:211 – 224, 1997.

[45] J. Smagorinsky. General circulation experiments with the primitive equations.

[46] S. Corzo, S. Márquez Damián, D. Ramajo, and N. Nigro. Numerical simulation of natural convection phenomena. *ENIEF*, XXX, 2011.

[47] G. De Vahl Davis. Natural convection of air in a square cavity: a benchmark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.

[48] S. Wakashima and T. Saitoh. Benchmark solutions for natural convection in a cubic cavity using the high-order time-space method. *International Journal of Heat and Mass Transfer*, 47:853–864, 2004.

[49] T. Fusegi, J.M. Hyun, K. Kuwahara, and B. Farouk. A numerical study of three-dimensional natural convection in a differentially heated cubical enclosure. *International Journal of Heat and Mass Transfer*, 34(6):1543 – 1557, 1991.